Kapitel 0
Was ist Kryptologie?

Die wörtliche Übersetzung ist einfach: Es geht um die Wissenschaft vom Geheimen, und natürlich sind damit nicht die geheimen „Erkenntnisse“ irgendwelcher Esoteriker gemeint, sondern es geht darum, Informationen aller Art vor unbefugtem Zugriff und vor Manupulationen zu schützen.

Grundthema der Kryptologie ist also die Sicherheit. Umgekehrt ist Kryptologie allerdings nur *einer* von vielen Sicherheitsaspekten: Zunächst einmal muß sichergestellt werden, daß eine Information trotz physikalisch-technisch unvermeidlicher Fehlerquellen korrekt rekonstruiert werden kann; darum kümmert sich auf der der mathematischen Seite die Kodierungstheorie und ansonsten natürlich vor allem die Informationstechnik.

Das traditionelle Hauptanliegen der *Kryptographie* ist die Verschlüsselung von Nachrichten, so daß diese über unsichere Leitungen, per Funk oder gar durch das Internet von einem Sender zu einem Empfänger geschickt werden können, ohne daß ein Lauscher mit der verschlüsselten Nachricht etwas anfangen kann. Dazu gehört insbesondere, daß er die Nachricht weder lesen noch unbemerkt verfälschen kann.

Auch dieses Ziel kann Kryptographie jedoch nicht allein erreichen: Sie ist machtlos, gegen physische Angriffe auf die Quelle einer Nachricht, und sie kann auch nicht davor schützen, daß sich ein Gegner durch „social engineering“ Kenntnis von geheimen Entschlüsselungsmethoden verschafft. Aufgabe der Kryptographie ist die Bereitstellung guter Algorithmen; gegen deren falsche Anwendung ist sie machtlos. Da eine Kette immer nur so stark ist wie ihr schwächstes Glied, ist eine starke Kryptographie daher zwar eine *notwendige,* keinesfalls aber eine hinreichende Bedingung für Sicherheit.

Eine starke Kryptographie sollte idealerweise *beweisbar stark* sein, d.h. wir sollten in der Lage sein, mathematisch zu beweisen, daß ein Gegner eine Verschlüsselung nicht oder wenigstens nur mit einem quantifizierbaren Mindestaufwand brechen kann. In Einzelfällen ist so etwas in der Tat möglich, allerdings werden wir bald sehen, daß dies nur bei sehr aufwendigen Systemen möglich ist, deren Einsatz sich nur bei sehr hohen Sicherheitsanforderungen lohnt.

Den kryptographischen Alltag dominieren schon lange nicht mehr die Schlapphüte, Militärs und Diplomaten: Kryptographie schutzt heute vor allem die Geheimzahlen von Geldkarten, den Handel über das Internet, den elektronischen Zahlungsverkehr zwischen Banken und ähnliche kommerzielle Interessen, für die Kryptographie ein Kostenfaktor ist, der sich nur lohnt, wenn die Verluste, die ohne Kryptographie entstünden, größer sind als der Aufwand für die Implementierung eines Kryptosystems.

Zur Illustration können wir etwa die beiden Varianten von bargeldlosen Zahlungen mit EC-Karten betrachten: Beim *electronic cash* muß der Kunde eine Geheimzahl angeben, die nur in verschlüsselter Form auf dem Magnetstreifen der Karte zu finden ist. Wie wir im dritten Kapitel sehen werden, kann das dabei verwendete Verschlüsselungsverfahren, der Triple-DES, nach heutigen Standards als sicher gelten – aber natürlich nur, solange die verwendeten Schlüssel geheim bleiben.

Aus diesem Grund hat ein Händler keine Möglichkeit, die Richtigkeit der Eingabe zu überprüfen; er muß die eingegebene Geheimzahl plus der Information auf dem Magnetstreifen zur Bank *bzw.* Clearingstelle übertragen und sich von dort darüber informieren lassen, ob die Geheimzahl zur Karte paßt oder nicht. Da Banken nichts umsonst tun, muß er dafür in der Regel 0,3% des Umsatzes an die Bank zahlen, kann allerdings auch sicher sein, daß die Bank im Falle der Akzeptanz der Karte auch wirklich bezahlt – auch wenn sich nachträglich herausstellen sollte, daß die Karte gestohlen war.

Alternativ kann er das Lastschriftverfahren anwenden, dessen Sicherheit ausschließlich darauf beruht, daß eine gestreßte Kassierin im Schlußverkauf fehlerfrei entscheiden kann, ob die Unterschrift auf der Karte mit der auf dem Lastschriftbeleg übereinstimmt. Hier gehen etwaige Fehler zu seinen Lasten; 2001 etwa entstand den Händlern dadurch ein Verlust von etwa 0,1% des Lastschriftumsatzes (DER SPIEGEL 3/2003, S. 76). Der volkswirtschaftliche Schaden, zu dem auch die Kosten polizeilicher Ermittlungen zählen, war natürlich größer, aber die Tatsache, daß das Lastschriftverfahren sehr viel weiter verbreitet ist als *electronic cash* zeigt deutlich, daß Sicherheit als solche in vielen kommerziellen Anwendungen keinerlei Rolle spielt; interessant sind für die meisten Anwender nur die eigenen Kosten und Risiken.

Die durchaus vorhandenen, aber aufwendigen absolut sicheren Kryptoverfahren sind in der heutigen kryptographischen Praxis daher nur eine Randerscheinung; die meisten praktisch eingesetzten Verfahren können zumindest *im Prinzip* gebrochen werden. Sie sollten aber jeweils so gewählt sein, daß der Aufwand dafür zumindest im Mittel deutlich größer ist als der zu erwartende Nutzen, denn bei kommerziellen Anwendungen wird wohl auch der Gegner meist zunächst anhand einer Kosten/Nutzen-Rechnung entscheiden, ob sich die Attacke lohnt.

Diese Betrachtungen zeigen, daß Kryptographie allein nicht ausreicht für die Beurteilung des Verschlüsselungsteils eines Sicherheitssystems: Auch wenn man sich nur für Verschlüsselung interessiert, muß man doch auch den zweiten Aspekt der Kryptologie betrachten, die *Kryptanalyse.*

Traditionelles Hauptanliegen der Kryptanalyse ist der Angriff auf ein Kryptosystem; ein Lauscher will also einen verschlüsselten Text unbefugt und ohne Kenntnis geheimer Information des Senders und/oder Empfängers entschlüsseln. Nach dem gerade Gesagten läßt sich die (relative) Sicherheit der meisten heute benutzten Kryptosysteme offenbar nur beurteilen kann anhand des *derzeitigen* Stands ihrer Kryptanalyse; Kryptanalyse ist also nicht nur wichtig für Hacker und Spione, sondern essentielle Voraussetzung für jede realistische Beurteilung der Sicherheit eines Verfahrens. Ein Verfahren kann nur dann als *wahrscheinlich* sicher gelten, wenn hinreichend viele Experten hinreichend lange an seiner Kryptanalyse gearbeitet haben, ohne größere Mängel zu finden.

Die Vertraulichkeit von Nachrichtigkeiten ist heute bei weitem nicht mehr das einzige Thema der Kryptologie: Oft ist es mindestens genauso wichtig, daß der Empfänger *sicher* sein kann, daß die Nachricht (etwa ein Überweisungsauftrag) wirklich vom behaupteten Absender kommt, und er muß in vielen Fällen auch in der Lage sein, dies einem neutralen Dritten zu beweisen. Insbesondere muß er dabei natürlich auch nachweisen, daß nicht er selbst die Nachricht verfaßt hat. Zu diesem Zweck stellt die Kryptologie *elektronische Unterschriften* zur Verfügung, die in Deutschland wie auch in vielen anderen Ländern seit einigen Jahren rechtsverbindlich sind.

Auch das Gegenteil der Absenderidentifizierung ist gelegentlich erwünscht: Nicht jeder möchte, daß seine Bank über alle seine Einkäufe Bescheid weiß (und dieses Wissen dann beispielsweise an die werbetreibende Wirtschaft verkauft). Zu diesem Zweck stellt die Kryptologie *elektronisches Bargeld* zur Verfügung, das genau wie gewöhnliches Bargeld nicht zum Kunden zurückverfolgt werden kann, bei dem aber trotzdem sichergestellt ist, daß der Verkäufer sein Geld bekommt.

Schließlich geht es manchmal auch einfach darum, eine Person zu identifizieren *ohne* daß dabei eine Nachricht übermittelt wird. Dieses Problem tritt beispielsweise auf bei kartenbasierten Zugangskontrollsystemen oder auch bei Mobiltelephonen, wo klar sein muß, zu Lasten welcher SIM-Karte ein Gespräch abgerechnet wird.

Wie sich herausstellen wird, sind es oft dieselben mathematischen Verfahren, die zur Lösung mehrerer wenn nicht gar aller dieser Probleme eingesetzt werden können. Welches Verfahren man wofür einsetzt, hängt einerseits ab von der Sicherheitstufe, die man erreichen will, andererseits vom Aufwand, den zu treiben man bereit ist. Wir werden uns daher nicht nur mit mathematischen Algorithmen befassen müssen, sondern auch mit derem Aufwand sowie dem Aufwand, mit dem sie ein Kryptanalytiker aushebeln kann.

Zunächst aber wollen wir, sowohl zur Einstimmung als auch zur Schärfung des Problembewußtseins, einige einfache klassische Verfahren betrachten.

Kapitel 1
Klassische Verfahren der Kryptologie

Das wichtigste vorab: Abgesehen von einer einzigen Ausnahme ist keines der hier vorgestellten Verfahren auch nur ansatzweise sicher. Die meisten lassen sich ohne große Kenntnisse und ohne großen Aufwand von Hand knacken; einem Experten, der mit Computer arbeitet, bereitet (abgesehen von der einzigen Ausnahme) keines nennenswerte Schwierigkeiten.

Daraus sollte man folgern, daß diese Verfahren heutzutage keine Rolle mehr spielen, aber dem ist leider nicht so: Viele Verschlüsselungsalgorithmen für Festplatten oder in Textverarbeitungsprogrammen arbeiten noch heute mit Algorithmen wie dem VIGENÈRE-Verfahren, das wir gleich als völlig unsicher erkennen werden. Der einzige „Fortschritt“ besteht darin, daß nun nicht mehr die 26 Buchstaben des Alphabets die Bausteine einer Nachricht bilden, sondern die 256 möglichen Bytes. Das ist nicht nur für den Anwender eine große Bequemlichkeit, sondern auch für den Kryptanalytiker, denn wie wir im nächsten Kapitel sehen werden, erhöht es die Redundanz der Nachricht deutlich und bietet damit bessere Ansatzpunkte zur unbefugten Entschlüsselung.

Die alten Kryptologen hatten das verstanden: Sie verzichteten bewußt auf die Unterscheidung zwischen Groß- und Kleinbuchstaben und ließen auch alle Satz- und Leerzeichen weg. Für den legitimen Empfänger ist das nur eine kleine Unbequemlichkeit; für den illegitimen Kryptanalytiker bedeutet es zumindest bei kurzen Nachrichten eine deutliche Erschwernis.

Sinn dieses Kapitels ist nicht in erster Linie die Vorstellung alter Verfahren; es soll in erster Linie eine Vorstellung davon vermitteln, mit welch

geringem Aufwand man gelegentlich Verschlüsselungsverfahren von (auf den ersten Blick) gewaltiger Komplexität knacken kann und auf die informationstheoretischen Ansätze des nächsten Kapitels vorbereiten.

§ 1: Die Anfänge

Wo nur eine verschwindend kleine Minderheit lesen kann, ist *jede* Schrift eine Geheimschrift; viele der frühen Hochkulturen wie etwa die chinesische sahen daher keinen Grund für die Entwicklung von Verschlüsselungstechniken.

Auch wenn der Sender und Empfänger eine gemeinsame Sprache haben, die ihrer Umgebung unbekannt ist, gibt es keine Notwendigkeit für eine Verschlüsselung. Dies war selbst im zwanzigsten Jahrhundet noch durchaus praktikabel: Beispielsweise beschäftigte die amerikanische Armee im zweiten Weltkrieg Navajos als *code talkers.* Da indianische Sprachen eine sehr komplexe Grammatik und eine schwierige Phonetik haben, sind sie nur schwer erlernbar und noch schwieriger nachzuahmen: Damals waren nur 28 Nicht-Navajos bekannt, die der Sprache mächtig waren, größtenteils Missionare. Aus diesem Grund hatte die Kommunikation in Navajo eine größere Sicherheit als die immer wieder geknackte maschinelle Kryptographie des zweiten Weltkriegs.

Ähnlich zeigte sich nach dem zweiten Weltkrieg, beim UN-Einsatz im Kongo, daß das irische Kontingent die effektivste Kryptographie hatte: Die Soldaten unterhielten sich einfach auf gälisch.

Ein historisches Beispiel, wie gelegentlich selbst weitbekannte Sprachen als Kryptosystem dienen können, bietet der Burenkrieg: Dort nutzten die Engländer aus, daß nur wenige Buren nennenswerte Bildung hatten und verwendeten Latein als Kryptographie. Die Einnahme der Stadt *Sindh* etwa wurde gemeldet mit der Nachricht *pecavi,* auf englisch *I have sinned.*

Anderswo schützte man Nachrichten dadurch, daß selbst das *Vorhandensein* einer Nachricht verschleiert wurde; von einem berühmten Beispiel etwa berichtet HERODOT in seinen *Historien:*

Ein gewisser HISTIAEUS, der am persischen Hof lebte, wollte seinen Schwiegersohn, den Tyrannen ARISTAGORAS von Miletos, zu einem Aufstand gegen die Perser bewegen. Dazu rasierte er einem zuverlässigen Sklaven die Haare ab, tätowierte seine Botschaft in dessen Kopfhaut und schickte ihn, nachdem die Haare nachgewachsen waren, zu ARISTAGORAS mit dem Auftrag, sich von diesem die Haare schneiden zu lassen.

Heute bezeichnet man dieses Verschleiern der bloßen Existenz einer Nachricht als *Steganographie;* wo sie erfolgreich ist, besteht keine Notwendigkeit mehr, sich um *Kryptographie* zu kümmern.

In vielen Situationen ist Steganographie undurchführbar, da entweder der Gegner weiß, daß Nachrichten ausgetauscht werden, oder aber weil einfach der Umfang der Kommunikation zu groß ist. Typisches Beispiel ist die militärische Kommunikation, für die erstmals die Spartaner um 487 v.Chr. ein kryptographisches Verfahren benutzten, den sogenannten *Skytale:* Ein Lederband wurde um ein Stück Holz gewickelt, die Nachricht in senkrechten Spalten auf das Leder geschrieben und dieses anschließend von einem Soldaten als Gürtel getragen. Der Empfänger mußte ein identisches Stück Holz haben, konnte das Band darum wickeln und sodann die Nachricht lesen.

Auch in Indien war Kryptographie zu politischen und militärischen Zwecken schon früh bekannt; kurz vor 300 v.Chr. schrieb KAUTILYA sein Buch *Artha-sastra,* in dem er (für Diplomaten) nicht nur die Verwendung von Codes erklärt, sondern auch, wie man diese knacken kann – spätestens seit damals gibt es also zusätzlich zur Kryptographie auch die *Kryptanalyse.* Rund 700 Jahre später bezeichnete VATSAYANA in der *Kama sutra* Kryptographie in Schrift und Wort als zwei der 64 Fähigkeiten, über die jede Frau verfügen sollte.

Der älteste heute noch verwendete Code geht auf CAESAR zurück und ist ein deutlicher Rückschritt hinter die Spartaner und Inder: Genau wie die Römer selbst war auch ihre Kryptographie sehr primitiv. Sie bestand im Falle CAESARS einfach darin, das Alphabet zyklisch um drei Stellen nach rechts zu verschieben, d.h.

$$A \to D, \quad B \to E, \quad ,\dots, \quad X \to A, \quad Y \to B, \quad Z \to C\,.$$

Heute bezeichnet man jeden Code, der das Alphabet zyklisch um eine feste Anzahl n von Stellen verschiebt, als CAESAR-*Chiffre;* im Internet bedeutsam ist der Fall $n = 13$, die sogenannte `ROT-13`-Chiffre des Usenet. Sie dient natürlich nicht dazu, Informationen geheimzuhalten: Ihr Gebrauch ist lediglich ein Hinweis darauf, daß der verschlüsselte Text nicht nach jedermanns Geschmack ist und jemand, der ihn trotzdem entschlüsselt, sich nicht über den Inhalt beklagen sollte.

§2: Arten kryptanalytischer Angriffe

In seiner Originalform hat CAESARs Verfahren einen wesentlichen Nachteil: Da alles eindeutig festgelegt ist, kann jeder, der irgendwann die Methode erlernte, auch künftig alle Nachrichten lesen – angesichts der zeitlich sehr variablen Bündnisse im römischen Reich sicherlich nicht ganz unproblematisch.

Auch ein Gegner, der es irgendwie geschafft hat, *eine* Nachricht zu entschlüsseln, hat künftig die Möglichkeit *jede* Nachricht zu entschlüsseln. Im Rom des ersten Jahrhunderts, wo wohl niemand die *Artha-sastra* kannte, war das vielleicht kein sehr großes Problem; für die heutigen Massenanwendungen der Kryptographie im Internet und im Bankenbereich ist eine solche Chiffre aber wertlos: Selbst die Militärs gehen schon seit mindestens einem halben Jahrhundert davon aus, daß das *grundsätzliche* Verschlüsselungsverfahren über kurz oder lang auch dem Gegner bekannt sein wird – auch wenn sie selbstverständlich weiterhin alles tun, um dies zu verhindern. Die Spartaner hatten hier vorgesorgt: Auch wer ihr Verfahren kannte, hatte ohne das jeweils passende Stück Holz Schwierigkeiten, die Nachricht zu lesen.

Abgesehen von der Original CAESAR-Chiffre hat daher praktisch jedes kryptographische Verfahren zwei Komponenten: Einmal die *Verschlüsselungsmethode,* bei der man realistischerweise davon ausgehen muß, daß man sie nicht unbegrenzt geheimhalten kann, und dann noch einen *Schlüssel,* der bei der Anwendung dieser Methode benutzt wird und der häufig gewechselt werden muß. Als *sicher* darf man ein Kryptoverfahren nur dann betrachten, wenn ein Gegner trotz genauer Kenntnis des

Verfahrens nicht in der Lage ist, einen Text ohne Kenntnis des Schlüssels zu dechiffrieren.

Der letzte Satz klingt vernünftig, ist aber unvollständig, da er eine wesentliche Frage ausklammert: *Was darf der Gegner alles wissen, bevor die Sicherheit des Verfahrens* (der Schlüssel) *gefährdet ist?*

Die offensichtliche Antwort scheint zu sein, daß er den chiffrierten Text und das Verfahren kennt. Das ist aber leider viel zu optimistisch: Schon 725 beschrieb ABU YAHMADI, der Autor des ersten arabischen Wörterbuchs, wie er ein altes byzantinisches Kryptogramm löste ausgehend von der Annahme, daß es mit der Formel „Im Namen Gottes“ beginne – für einen Text aus der fraglichen Zeit fast eine Selbstverständlichkeit.

Heute beginnen wir Texte nicht mehr mit der Formel „Im Namen Gottes“, aber wir verwenden Word-Dateien, zip-Dateien *usw.*, die mit *magic bytes* und anderer Vorspann-Information beginnen; einige von uns versenden e-mails, bei denen die ASCII-Kunst oder die Visitenkarte am Ende länger ist als die Nachricht selbst; Militärs beginnen Nachrichten typischerweise mit Worten wie „Wehrmachtskommando XY an . . . “; Banken haben sich für elektronische Überweisungen auf ziemlich rigide Formate geeinigt; jeder Besitzer einer EC-Karte kennt seine Geheimzahl und die sonstige Kontoinformation auf der Karte und kann, so er einen Magnetkartenleser hat, auch die verschlüsselte Version auf der Karte lesen: Falls er mit dieser Information (eventuell auch von vielen EC-Karten statt nur von einer) die zehnstellige Schlüsselzahl des Systems ermitteln könnte, müßten sofort alle Geldautomaten abgeschaltet werden.

Ein Gegner kennt daher in vielen Fällen nicht nur den Chiffretext, sondern er kennt auch zumindest einen Teil des zugehörigen Klartexts. Ein gutes Kryptoverfahren muß sicherstellen, daß er trotzdem nicht in der Lage ist, den Schlüssel zu identifizieren und damit auch den ihm unbekannten Rest zu entschlüsseln.

In manchen Situationen ist der Gegner sogar in einer noch besseren Position: Er kann sich zu von ihm frei gewähltem Klartext den Chiffretext verschaffen. Eine solche Situationen könnte beispielsweise auftreten, wenn ein Diplomat ein Schreiben übergibt, bei dem er ziemlich sicher sein kann, daß es anschließend verschlüsselt weitergegeben wird. Kurz vor Ausbruch des zweiten Weltkriegs übergab sogar einmal ein japanischer Diplomat ein Schreiben an das amerikanische Außenministerium mit der Bitte, es nach Japan zu übermitteln, da er den Beamten der eigenen Chiffrierabteilung mißtraue – ob dahinter ein Versuch einer solchen Attacke lag, ist bis heute unbekannt. (Die Amerikaner übermittelten das Dokument.)

Heute bilden auch Paßwortdateien wenig gesicherter Computersysteme einen Ansatz für solche Attacken: Jeder Benutzer kann sein Paßwort beliebig oft ändern und auf jeden beliebigen Wert setzen; das Ergebnis wird verschlüsselt in einer Paßwortdatei gespeichert, die auf manchen Computern für jeden Benutzer lesbar ist. Natürlich muß das Verschlüsselungsverfahren sicherstellen, daß trotzdem niemand in der Lage ist, die Paßwörter *anderer* Benutzer zu entschlüsseln.

Auch Smartcards bieten Ansätze für Angriffe mit frei wählbarem Klartext: Sichere Kryptosysteme arbeiten mit Schlüssellängen, die weit jenseits dessen liegen, was sich ein Mensch merken kann; sie können daher nur mit Smartcards oder etwas ähnlichem funktionieren. Diese sind natürlich *(?)* zusätzlich durch eine Geheimzahl oder ein Paßwort gesichert, aber viele unbedarfte Benutzer stellen praktisch sicher, daß ihre Karte nie ohne Paßwort gestohlen werden kann. Damit ist der Dieb in der Lage, die Karte zum Lesen verschlüsselter Nachrichten, zur elektronischen Unterschrift mit dem Namen des Bestohlenen und in Zugangskontrollsystemen verwenden – bis der Diebstahl entdeckt wird.

Eine vielseitig einsetzbare Karte wird sicherlich schnell vermißt werden; danach wird der in die Karte kodierte Schlüssel für ungültig erklärt und somit nutzlos. Ein geschickter Dieb muß also die Karte wieder zurückgeben, *bevor* der Verlust bemerkt wird. Die Schlüsselinformation kann er nicht kopieren: Die steht in einem auslesesicheren Register. Er kann aber beliebig viel Eingabetext seiner Wahl von der Karte bearbeiten lassen, und ein gutes Kryptosystem muß sicherstellen. daß er trotzdem nichts über den Schlüssel erfährt und ohne die Karte später nicht mehr in der Lage ist, verschlüsselte Nachrichten an den Besitzer zu dechiffrieren, in dessen Namen zu unterschreiben oder sich als diesen auszugeben.

§3: Kryptanalyse der Caesarchiffre

Es ist klar, daß auch eine CAESAR-Substitution mit unbekannter Verschiebung n keinen Schutz bietet gegen einen Angreifer, der Klartext und zugehörigen Chiffretext kennt: Sobald er nur für einen einzigen Buchstaben sowohl diesen als auch seine Verschlüsselung kennt, kann er n berechnen.

Aber auch ein Angreifer, der allein mit Chiffretext arbeiten muß, hat keine großen Schwierigkeiten: Er kann beispielsweise einfach alle 26 Möglichkeiten $n = 0, \ldots, 25$ durchprobieren; sobald sinnvoller Klartext entsteht, hat er den Schlüssel gefunden. Informationstheoretische Überlegungen zeigen, daß im Allgemeinen schon nach zwei Buchstaben klar ist, ob es sich bei einer Probeentschlüsselung um deutschen Klartext handeln kann oder nicht.

Es geht aber noch einfacher: Bekanntlich kommen die 26 Buchstaben des Alphabets in deutschem Text nicht mit gleicher relativer Häufigkeit vor. Abbildung eins zeigt die Häufigkeitsverteilung der 260 238 Buchstaben aus dem Roman *Dr. Katzenbergers Badereise* von JEAN PAUL. (Wer nachzählen möchte, findet diesen und viele andere Texte im Internet auf den Seiten des Projekts Gutenberg unter `http://gutenberg.aol.de/` und bei mehreren mirror sites.)

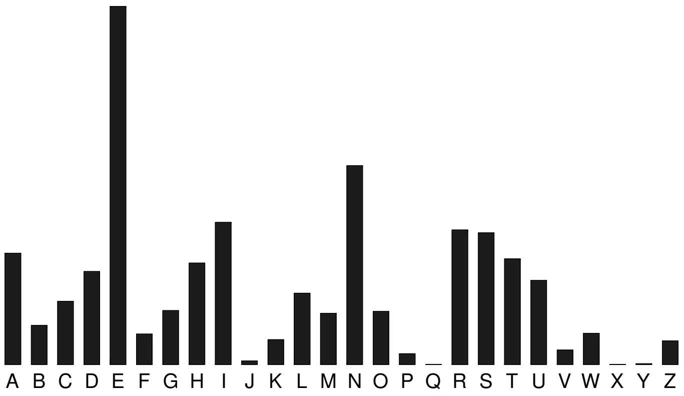


Abb. 1: Buchstabenhäufigkeiten in deutschem Text

In Prozentzahlen ausgedrückt sieht das Ergebnis so aus:

A	B	C	D	E	F	G	H	I
5,75	2,03	3,27	4,81	18,49	1,58	2,79	5,25	7,35

J	K	L	M	N	O	P	Q	R
0,19	1,29	3,69	2,65	10,27	2,75	0,57	0,01	6,95

S	T	U	V	W	X	Y	Z
6,81	5,46	4,35	0,76	1,62	0,02	0,05	1,23

Wie zu erwarten war, ist also E mit großem Abstand der häufigste Buchstabe, während Q, X und Y sehr selten vorkommen.

Wenn ich einen anderen Text als *Dr. Katzenbergers Badereise* ausgezählt hätte, wären die Ergebnisse natürlich anders gewesen: Das Spiegel-Interview mit dem Nobelpreisträger HORST STÖRMER etwa hat 5455 Buchstaben, die sich prozentual folgendermaßen verteilen:

A	B	C	D	E	F	G	H	I
6,00	2,07	2,84	4,16	17,23	1,50	2,97	4,68	8,18

J	K	L	M	N	O	P	Q	R
0,26	1,20	3,81	3,27	10,45	3,04	1,05	0,07	6,93

S	T	U	V	W	X	Y	Z
6,84	5,56	4,42	0,73	1,43	0,02	0,33	1,03

Zumindest in absoluten Prozentzahlen sind die Unterschiede praktisch vernachlässigbar; die *relativen* Unterschiede sind bei den häufigen Buchstaben ebenfalls unbedeutend, bei den seltenen allerdings teilweise recht dramatisch.

Bedenkt man aber, daß 0,02% X bei einem Text aus 5455 Buchstaben gleichbedeutend ist mit nur einem X und 0,07% Q mit vier Q, so sieht man, daß auch diese Abweichungen minimale Zufallsschwankungen sind.

Für sehr kurze Texte spielen solche Zufallsschwankungen eine große Rolle: Falls wir etwa das Kryptogramm

```
DQJUL IILPP RUJHQ JUDXH Q
```

entschlüsseln wollen, zeigt uns Abbildung zwei, daß es dort gleich drei häufigste Buchstaben gibt: J, Q und U kommen je dreimal vor. Entschlüsselt man unter der Annahme, daß J dem E entspricht, ist $n = 5$ und das Entschlüsselungsergebnis `YLEPG DDGKK MPECL EPYSC L` enthält zwar drei E, hat aber sonst nur wenig mit deutschem Text zu tun. Auch die Entschlüsselungsversuche

`REXIZ WWZDD FIXVE XIRLV E`

und

`NATEV SSVZZ BETRA TENHR A`,

die von $E \to Q$ (d.h. $n = 12$) und $E \to U$ (d.h. $n = 16$) ausgehen, liefern kein vernünftiges Ergebnis.

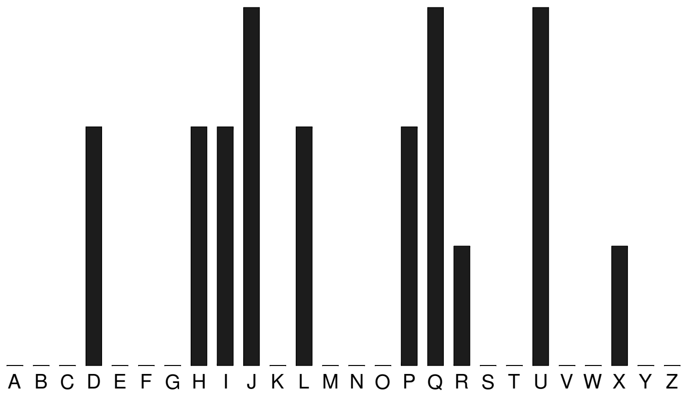


Abb. 2: Buchstabenhäufigkeiten eines kurzen Kryptogramms

Ein klassischer Kryptanalytiker wäre anders vorgegangen: Er wäre sich beim Blick auf Abbildung zwei praktisch sicher gewesen, daß die drei Leerstellen am Ende und die beiden am Anfang wohl nur den fünf seltenen Buchstaben von V bis Z entsprechen können, so daß Abbildung eins um drei Stellen verschoben wurde, d.h. A wurde als D verschlüsselt *usw.,* wie es schon CAESAR praktizierte. Dies führt zum in fast jedem Lehrbuch der klassischen Kryptographie zu findenden Klartext `ANGRIFF IM MORGENGRAUEN`.

Ein moderner Kryptograph möchte sich mit trivialen Kryptogrammen wie dem gerade betrachteten nicht mehr selbst beschäftigen; für solche Dreckarbeit hat er schließlich seine Computer. Denen springt keine Lücke *ins Auge,* und sie haben auch Schwierigkeiten, Lücken *exakt* von Nichtlücken zu unterscheiden. Diese Schwierigkeiten hatte der klassische Kryptanalytiker zwar auch, aber er konnte sich da im Gegensatz zu einem Computer meist auf sein Gefühl verlassen – auch wenn ihn das immer wieder einmal gewaltig in die Irre schickte.

Computer dagegen lieben Zahlen, und auch dazu können ihnen die klassischen Kryptanalytiker verhelfen: Sei p_i die Häufigkeit des i-ten Buchstabens des Alphabets in deutschen Texten, geschätzt etwa durch die Prozentzahlen in obiger Tabelle, und sei q_i die Häufigkeit dieses Buchstabens im zu untersuchenden Kryptogramm. Im obigen Beispiel wäre also

$$q_{10} = q_{17} = q_{21} = \frac{3}{21} = \frac{1}{7}\,,$$

da J, Q und U jeweils dreimal vorkommen;

$$q_4 = q_8 = q_9 = q_{12} = q_{16} = \frac{2}{21}\,,$$

da D, H, I, L und P je zweimal vorkommen,

$$q_{17} = q_{24} = \frac{1}{21}$$

für die beiden einfach vorhandenen Buchstaben R und X, und alle übrigen q_i verschwinden.

Angenommen, die richtige Entschlüsselung besteht aus der Verschiebung um n Buchstaben. Dann sollte

$$q_i \approx p_{i \oplus n} \quad \text{für } i = 1, \ldots, 26$$

sein, wobei

$$i \oplus n = 1 + \big((i + n - 1) \bmod 26\big)$$

die zyklische Verschiebung um n Buchstaben beschreibt; denn q_i gibt dann ja die Häufigkeit an, mit der der $(i \oplus n)$-te Buchstabe im Klartext des Kryptogramms auftaucht.

Als Maß für die Gültigkeit obiger Näherungsformeln können wir den EUKLIDischen Abstand (in $\mathbb{R}^{26}$) zwischen dem Vektor mit Komponenten p_i und dem mit Komponenten $q_{i\oplus n}$ berechnen, also

$$\delta_n = \sum_{i=1}^{26} (p_i - q_{i\oplus n})^2 ,$$

und wir erwarten, daß das richtige n zu einem kleinen δ_n führt.

Rechnerisch etwas einfacher wird es, wenn wir δ_n ausmultiplizieren:

$$\begin{aligned} \delta_n &= \sum_{i=1}^{26} p_i^2 + \sum_{i=1}^{26} q_{i\oplus n}^2 - 2\sum_{i=1}^{26} p_i q_{i\oplus n} \\ &= \sum_{i=1}^{26} p_i^2 + \sum_{i=1}^{26} q_i^2 - 2\sum_{i=1}^{26} p_i q_{i\oplus n} , \end{aligned}$$

denn in der Summe über die q_i tritt jede der 26 Häufigkeiten genau einmal auf. Somit hängt nur der letzte der drei Summanden von n ab, und δ_n wird genau dann klein, wenn

$$r_n = \sum_{i=1}^{26} p_i q_{i\oplus n}$$

groß ist.

Im vorliegenden Fall sind die größten Werte

$$r_{16} = 0{,}0707, \quad r_3 = 0{,}0661, \quad r_{12} = 0{,}0541 \quad \text{und} \quad r_7 = 0{,}0538 .$$

Der erste Versuch mit $n = 16$ führt, wie wir bereits wissen, zu keinem vernünftigen Ergebnis; der zweite mit $n = 3$ entschlüsselt aber das Kryptogramm.

§4: Vigenère-Chiffren

CAESAR-Substitutionen sind so einfach zu entschlüsseln, weil es erstens nur 26 Verschlüsselungsmöglichkeiten gibt und zweitens die Häufigkeitsverteilung der Buchstaben viele Anhaltspunkte gibt. Zur Verbesserung des Verfahrens muß also erstens die Anzahl der Verschlüsselungsmöglichkeiten deutlich erhöht werden, und zweitens sollte man versuchen, die Häufigkeitsverteilung der Buchstaben im Chiffretext möglichst homogen zu machen.

Beide Ziele erreicht die VIGENÈRE-Chiffre, indem sie mehrere CAESAR-Substitutionen miteinander mischt: Schlüssel ist eine (nicht zu kurze) Buchstabenfolge. Diese schreibt man so oft hintereinander, bis eine Buchstabenfolge entsteht, die mindestens genauso lang ist wie der zu verschlüsselnde Text, und man verschlüsselt den i-ten Klartextbuchstaben dadurch, daß man ihn zum i-ten Buchstaben der aus dem Schlüssel entstandenen Buchstabenfolge „addiert“. Bei dieser „Addition“ wird A mit 1, B mit 2 identifiziert *usw.*, und ein Additionsergebnis größer 26 wird durch Subtraktion von 26 wieder in den richtigen Bereich gebracht. Beispielsweise ist also

$$A + A = B, \quad B + B = D, \quad A + Y = Z \quad \text{und} \quad B + Y = A .$$

Verschlüsselung von `Angriff im Morgengrauen` mit dem Schlüssel `Krypto` führt somit zu

```
  ANGRI FFIMM ORGEN GRAUE N
+ KRYPT OKRYP TOKRY PTOKR Y
= LFFHC UQALC IGRWM WLPFW M
```

VIGENÈRE-Chiffren sind benannt nach BLAISSE DE VIGENÈRE, der sie 1586 in seinem Buch *Traictés des chiffres ou secrètes manières d'escrire* beschrieb; sie wurden allerdings bereits im 1518 erschienenen Buch *Polygraphia* des Abbé JEAN TRITHÈME erwähnt und gehen laut VIGENÈRE zurück auf die Hebräer.

L. SACCO, der ehemalige Chef des Chiffrierdienstes der italienischen Armee, schreibt in seinem *Manuel de cryptographie* (Paris, 1951):

> *Vigenère réussit à produire un chiffre plus faible et moins commode que les précédents, qui n'en connut pas moins une fortune imméritée, particulièrement dans les cent dernières années, où il fut adopté par de nombreuses armées, même après la publication d'un moyen de décryptement (1863), indice évident de décadence dans la domaine de la cryptographie.*

Die Vigènere-Chiffre oder ähnliche Verfahren wurden bis Ende des neunzehnten Jahrhunderts unter anderem von der österreichischen, der französischen und der italienischen Armee benutzt; von letzterer sogar bis 1917. In verschiedenen Computerprogrammen ist ähnliches heute noch zu finden.

Um zu verstehen, warum Sacco trotzdem so abfällig darüber spricht, müssen wir ihre Sicherheit etwas genauer betrachten und insbesondere sehen, wann sie mit welchem Aufwand geknackt werden kann.

Für einen Schlüssel mit nur einem Buchstaben haben wir einfach eine Caesar-Substitution, die praktisch keinerlei Schutz bietet – es sei denn, die Nachricht sei nur einen Buchstaben lang. In diesem Fall bietet die Caesar-Chiffre perfekte Sicherheit gegen eine Attacke nur mit Chiffretext, denn genau wie der Schlüsselbuchstabe kann auch der Klartextbuchtabe beliebig sein.

Genauso bietet die Vigenère-Chiffre absolute Sicherheit gegen einen Angriff nur mit Chiffretext, wenn der Schlüssel mindestens genauso lang ist wie die Nachricht und zufällig gewählt wird. Verschlüsseln wir die wohlbekannte Nachricht mit einem zufällig gewählten Schlüssel der Länge 21, so erhalten wir etwa

```
  ANGRI FFIMM ORGEN GRAUE N
+ KCHQR OFVFN FVSLA XRQBV E
= LQOIA ULESA UNZQO EJRWA S
```

Nun könnte möglicherweise ein Gegner irgendwie den Schlüssel erraten und damit auf den Klartext stoßen. Das nützt ihm aber nur wenig, denn es gibt noch viele andere Klartexte, die zu diesem Kryptogramm passen: Wenn er andere Schlüssel ausprobiert, erhält er beispielsweise auch die Entschlüsselung

```
  LQOIA ULESA UNZQO EJRWA S
– JYFUT PJSXN PZTVJ MWWCG J
= BRING EBLUM ENFUE RMUTT I
```

und entsprechend läßt sich auch jeder andere Klartext mit 21 Buchstaben produzieren; er kann aus dem Kryptogramm also keine weitere Information ziehen, als daß der Klartext 21 Buchstaben lang war. Wenn in einem Kryptosystem der Schlüssel mindestens genauso komplex ist wie der Chiffretext, läßt sich durch geschickte Wahl von Schlüssel und Verschlüsselungsverfahren fast jede Botschaft in die Chiffre hineinlesen – was von Personen, die Nachrichten von Außerirdischen bekommen und/oder Verschwörungstheorien beweisen, bewußt oder unbewußt durchaus auch ausgenutzt wird.

Für die Sicherheit eines Vigenère-Verfahrens mit Schlüssellänge = Nachrichtenlänge ist ganz wesentlich, daß der Schlüssel *zufällig* gewählt wird; er darf auf keinen Fall beispielsweise ein Buch sein, denn die „Summe“ (im Vigenère-Sinne) zweier deutscher Klartexte hat noch so viele statistische Gesetzmäßigkeiten, daß die beiden Summanden relativ einfach ermittelt werden können.

Das gerade vorgestellte perfekt sichere Verfahren bezeichnet man in der Kryptographie als *one time pad,* da es früher meist mit Schreibblöcken realisiert wurde: Man erzeugte durch einen Zufallsprozeß Buchstabenfolgen und notierte diese auf zwei Schreibblöcken in gleicher Weise, bis die Blöcke voll waren. Diese beiden Blöcke gingen dann an die beiden Kommunikationspartner, die für ihre i-te Nachricht die Buchstabenfolge auf Blatt i des Blocks als Schlüssel benutzten; danach wurde dieses Blatt vernichtet. Als erstes benutzte diese Technik wohl der amerikanische General Vernam im ersten Weltkrieg; im zweiten Weltkrieg kommunizierte London auf diese Weise mit der französischen *Résistance,* und später sicherten auch Fidel Castro und Che Guevara ihre Kommunikation auf diese Weise.

Als *high tech*-Variante davon entstand im Kalten Krieg das *Rote Telephon* zwischen dem Weißen Haus und dem Kreml: Damals hielten viele (wohl zu Recht) die Gefahr eines Atomkriegs aus Versehen für erheblich größer als die eines absichtlichen. Um ersteren etwas weniger wahrscheinlich zu machen, einigten sich die beiden Großmächte im Juni 1963 in Genf darauf, das sogenannte *Rote Telephon* einzurichten; es funktioniert seit dem 30. August 1963.

Natürlich handelt es sich dabei nicht wirklich um ein Telephon, denn zu keinem Zeitpunkt des kalten Krieges reichten die Sprachkenntnisse eines amerikanischen Präsidenten oder eines Generalsekretärs der KPdSU auch nur für ein direktes Gespräch über das Wetter. Tatsächlich geht es um eine Fernschreibverbindung mit je vier Fernschreibern an beiden Enden: jeweils zwei mit lateinischem und zwei mit kyrillischem Alphabet. Bislang verbrachten sie ihre meiste Zeit damit, stündliche Testnachrichten zu drucken wie amerikanische Baseball-Ergebnisse oder TURGENJEWS *Aufzeichnungen einer Jägers.*

Aus Sicherheitsgründen wurden zwei Leitungen eingerichtet, eine entlang der Route Washington-London-Kopenhagen-Stockholm-Helsinki-Moskau, die andere via Tanger. Natürlich war es unmöglich, diese Leitungen auf ihrer ganzen Länge zu überwachen, so daß niemand ausschließen konnte, daß irgendwo zwischen Moskau und Washington eine vertrauliche Kommunikation abgehört oder – mit potentiell sehr viel katastrophaleren Folgen – eine gefälschte Nachricht eingespielt wurde. Aus diesem Grund mußten alle Nachrichten verschlüsselt werden

Das *Rote Telephon* benutzte stattdessen eine Variante eines alten, absolut sicheren, Verschlüsselungsverfahrens, des sogenannten *one time pads:* Von Zeit zu Zeit tauschten die beiden Seiten per Kurier Magnetbänder mit zufallserzeugten Bitfolgen aus. Jedesmal, wenn eine Nachricht übermittelt werden sollte, übersetzte der Fernschreiber diese in eine Bitfolge, d.h. in einen Vektor $\vec{v}$ aus einem Vektorraum $BbbF_2^N$. Aus den ersten N bislang noch nicht benutzten Bits auf dem Magnetband wurde dazu ein weiterer Vektor $\vec{w} \in \mathbb{F}_2^N$ gebildet, und tatsächlich übertragen wurde die Summe $\vec{s} = \vec{v} + \vec{w}$.

Am anderen Ende der Leitung, wo eine Kopie des Magnetbands vorlag, war $\vec{w}$ bekannt, so daß die Nachricht

$$\vec{v} = \vec{v} + \vec{0} = \vec{v} + (\vec{w} + \vec{w}) = (\vec{v} + \vec{w}) + \vec{w} = \vec{s} + \vec{w}$$

leicht rekonstruiert werden konnte.

Ein Lauscher ohne Magnetband konnte nur die Länge N der Nachricht ermitteln, was bei seitenlangen in Diplomatensprache formulierten Texten so gut wie keine konkrete Information liefert. Auch hat jemand, der irgendeinen Vektor $\vec{s}$ in die Leitung einspielt, so gut wie keine Chance,

daß nach Addition von $\vec{w}$ daraus verständlicher Text wird; die Manipulation wird also mit an Sicherheit grenzender Wahrscheinlichkeit entdeckt.

In alltäglicheren Anwendungen ist der mit dem *one time pad* verbundene Aufwand meist zu hoch; man muß notgedrungen mit Schlüsseln arbeiten, die deutlich kürzer sind als die (Summe der) Nachrichten, die damit verschlüsselt werden. Wir wollen sehen, wie sich das VIGENÈRE-System dann verhält.

Betrachten wir als erstes ein Beispiel das Kryptogramm

```
PDOAA KKMQB LYORJ FTLXM OQGYU XTKCQ LXLVB ATCBU MJEDM SQZJJ
OZPHT AEZZD FFRSK XTYZV MVVZS QTZUO CTGDX ENOGX XGCOI GXHBN
OZXCC COJXJ PBQTV XTDOF ZRZND FHADX LZCQC NPBSL HTDVM ESKSP
YFCDB QHCEV ZDVIA DRKTR PGJDQ RFUUW AKQXP UZQVD AMXLF XGFGC
BTCFT KYRES GSALT VGZWT VSQOP UCALM ZFGMA VNDOZ ZNJOA HVDDQ
CMONK CPPBU ZZZYK PYRAD LZLYV GXNUB IURKX OEBBZ DNAWE UOVKH
TPISF DLIIQ LOXHO PAIZZ CAWEV XYSXU IZVUQ MAICE SKYXL AIZEH
KVNCR KUXYH IFKMK BJVHO TRSXX ZIEZJ
```

Abbildung drei zeigt, daß die Buchstabenhäufigkeiten sehr viel gleichmäßiger verteilt sind als in deutschem Text; dies deutet darauf hin, daß keine *monoalphabetische,* sondern eine *polyalphabetische* Substitution angewandt wurde, d.h. die verschiedenen Buchstaben des Kryptogramms wurden nicht alle mittels ein und derselben Permutation aus den Klartextbuchstaben berechnet.

Es gibt polyalphabetische Substitutionen wie etwa den *one time pad,* die ohne Kenntnis von Verfahren und Schlüssel nicht entschlüsselbar sind. Erfolg wird der Kryptanalyst immer dann haben, wenn er die Buchstaben identifizieren kann, die mit derselben Permutation verschlüsselt wurden, und wenn er für (fast) jede Permutation genügend viele Buchstaben hat, um statistische Methoden anzuwenden.

Am einfachsten ist dies bei *periodischen* Verfahren, d.h. wenn die anzuwendende Permutation so wie beim VIGENÈRE-Verfahren periodisch wechselt. In diesem Fall muß er nur die Periode feststellen, um die mit derselben Permutation verschlüsselten Buchstaben zusammenzufassen.

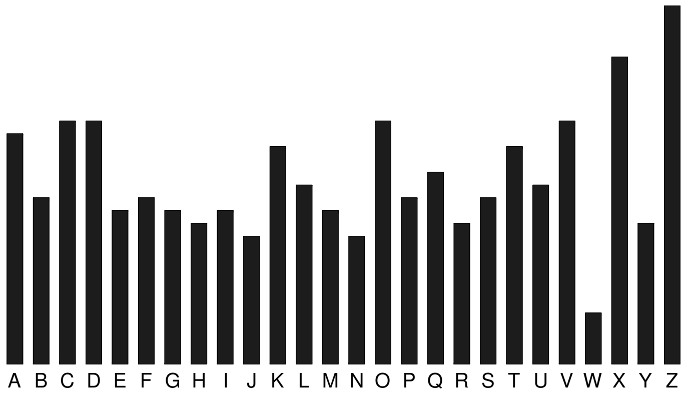


Abb. 3: Buchstabenhäufigkeiten des Beispielkryptogramms

Mit dieser Ermittlung der Periode hatten die Kryptanalytiker im letzten Jahrhundert lange ihre Probleme; den ersten Ansatz fand der preußische Offizier Friedrich W. Kasiki (1805–1881) und veröffentlichte ihn 1863 in einem damals kaum beachteten nur 95 Seiten dicken Buch mit dem Titel *Die Geheimschriften und die Dechiffrierkunst.* Seine Idee war die folgende: Gewisse Wörter und Buchstabenkombinationen wie etwa die bestimmten Artikel sind in fast allen Texten sehr häufig. Wenn nun ein langer Text mit einem polyalphabetischen Verfahren kurzer Periode verschlüsselt wird, ist es sehr wahrscheinlich, daß solche Buchstabenfolgen mehrfach auf die gleiche Weise verschlüsselt werden. Man suche daher im Kryptogramm nach zweimal vorkommenden Buchstabenfolgen (Heute nennt man so etwas ein Kasiki-Paar) und berechne deren Abstand. Die Periode sollte ein Teiler von relativ vielen dieser Abstände sein, wobei Abstände, die zu langen Buchstabenfolgen gehören, natürlich höher zu gewichten sind als solche, die etwa nur zu Buchstabenpaaren gehören: Bei letzteren ist die Wahrscheinlichkeit, daß es sich um eine zufällige Koinzidenz handelt, erheblich größer.

Die Suche nach Kasiki-Paaren ist recht aufwendig, und sie führen im allgemeinen nur dann zu einer Lösung, wenn das Kryptogramm erheblich länger als ist als der verwendete Schlüssel. Eine Alternative fand um 1920 der wohl bedeutendste der klassischen Kryptologen, William Friedman (1891–1969). Er wurde als Wolfe Friedman in Rußland geboren, aber als seine Eltern 1892 nach Amerika emigrierten, änderten sie seinen Vornamen. Er studierte zunächst Landwirtschaft, spezialisierte sich später auf Genetik und bekam 1915 eine Stelle als Genetiker bei dem Textilkaufmann George Fabyan, auf dessen Gut Riverbank in Geneva, Illinois. Dieser unterhielt dort Laboratorien für Akustik, Chemie, Genetik und Kryptologie – letztere mit dem Ziel zu beweisen, daß Bacon der wahre Autor der Shakespeareschen Schriften sei. Dadurch mußte sich Friedman zwangsläufig auch für Kryptologie interessieren und war damit so erfolgreich, daß er bald Leiter der Laboratorien sowohl für Genetik als auch für Kryptologie war.

Mit dem Kriegseintritt der Vereinigten Staaten im April 1917 mußte sich auch die amerikanische Armee für Kryptographie interessieren, und da es außer Riverbank kein amerikanisches Zentrum für Kryptologie gab, wurden nicht nur aufgefangene Kryptogramme dorthin geschickt, sondern auch Armee-Offiziere, die dort bei Friedman in Kryptanalyse ausgebildet werden sollten. Diese Kurse wurden nach Kriegsende fortgesetzt, und 1921 verließ Friedman Riverbank, um anschließend in verschiedenen militärischen Funktionen sowohl Codes zu entwerfen als auch Codes zu knacken. Von ihm stammt das Wort *Kryptanalyse,* das das unbefugte Dechiffrieren vom legitimen unterscheidet.

Seine 1925 perfektionierte Idee zur Bestimmung der Periode ist folgende: Man betrachte für eine natürliche Zahl n die Wahrscheinlichkeit dafür, daß ein Buchstabe mit seinem n-ten Nachfolger übereinstimmt.

Falls n ein Vielfaches der Periode ist, wurden die beiden Buchstaben mit derselben Permutation verschlüsselt; da Summen nicht von der Reihenfolge der Summanden abhängen, ist die Wahrscheinlichkeit also

$$\sum_{i=1}^{26} p_i^2 ,$$

wobei p_i die Häufigkeit des i-ten Buchstabens in typischem Klartext ist.

Falls n dagegen *kein* Vielfaches der Periode ist, stammen ein Buchstabe und sein n-ter Nachfolger bei hinreichend langer Periode fast immer aus verschiedenen Permutationen, man kann sie also in allererster Näherung als zufallsverteilt ansehen. Dann ist die Wahrscheinlichkeit einer Koinzidenz ungefähr gleich

$$\sum_{i=1}^{26}\left(\frac{1}{26}\right)^2 = \frac{1}{26}\,.$$

Betrachtet man die deutsche Sprache auf Grundlage von *Dr. Katzenbergers Badereise,* so ist

$$\sum_{i=1}^{26} p_i^2 \approx 0{,}0789 \quad \text{und} \quad \frac{1}{26} \approx 0{,}0385\,,$$

die beiden Werte unterscheiden sich also deutlich.

In einem relativ kurzen Kryptogramm wird man selbstverständlich andere Werte berechnen, aber trotzdem sollte es im allgemeinen für gleiche Alphabete mehr Koinzidenzen geben als für verschiedene.

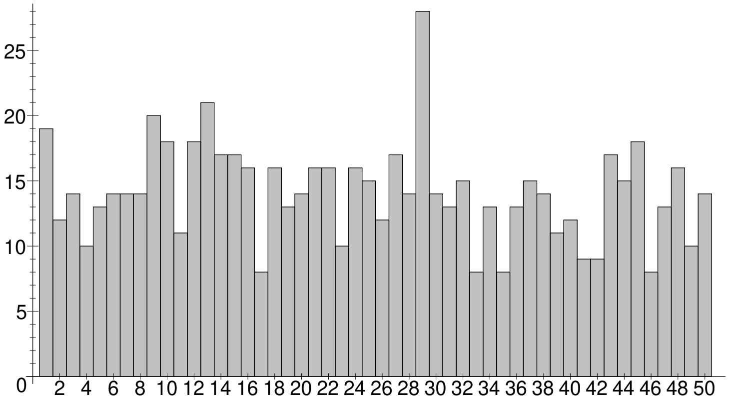


Abb. 4: Koinzidenzen im Beispielkryptogramm für Abstände bis 50

Für das obige Kryptogramm zeigt Abbildung vier die Verteilung der Koinzidenzen bei n Buchstaben Abstand. Das erste Maximum bei $n = 1$ ist natürlich nicht ernstzunehmen; sonst hätten wir eine monoalphabetische Substitution, deren Häufigkeitsverteilung deutlich anders aussieht

als Abbildung drei. Auch die Spitzen bei $n = 9$ und $n = 13$ sind wohl eher zufällig, denn bei $n = 18$ und $n = 26$ sind die Anzahlen eher klein. Unübersehbar ist das absolute Maximum bei $n = 29$; um zu sehen, ob dies das „richtige“ Maximum ist, müssen wir allerdings etwas mehr Werte berechnen. Abbildung fünf zeigt die entsprechenden Abstände bis einschließlich 120, und man sieht doch recht deutliche Ausschläge bei $n = 58$, 67 und 116. Wir wollen daher als erstes versuchen, das Kryptogramm unter der Annahme zu dechiffrieren, daß $n = 29$ ist.

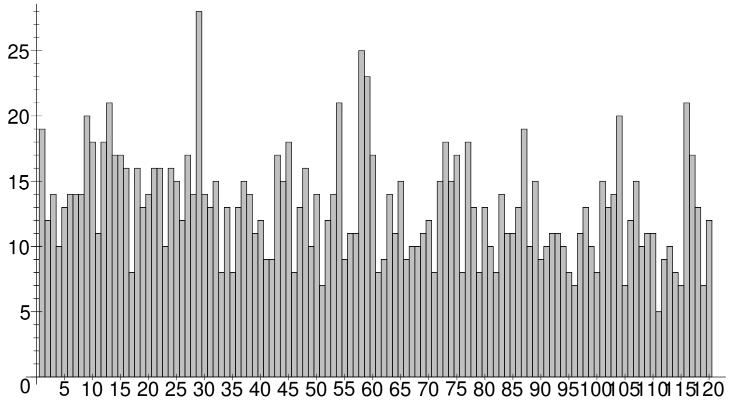


Abb. 5: Koinzidenzen im Beispielkryptogramm für Abstände bis 120

Wir arbeiten also in diesem Ansatz mit der Hypothese, daß zwei Buchstaben genau dann mit derselben Permutation verschlüsselt sind, wenn ihr Abstand ein Vielfaches von 29 ist.

Deshalb teilen wir das Kryptogramm auf in 29 Buchstabenfolgen, die jeweils mit derselben Permutation verschlüsselt sein sollten. Ein Kryptanalytiker würde nun die 29 Häufigkeitsverteilungen dazu betrachten; da sie meisten ungefähr so aussehen wie die von Caesar-Substitutionen, würde er Vigenère als wahrscheinlichste Möglichkeit ansehen.

In erster Näherung kann man Caesar-Chiffren über den pq-Test entschlüsseln; wendet man dies hier an, erhält man folgende Kandidaten für Schlüsselbuchstaben, wobei links jeweils der mit der größten $\sum p_i q_i$ steht, danach die vier mit den nächstkleineren:

```
N 0,085 E 0,057 Z 0,055 O 0,050 A 0,048
A 0,083 E 0,062 N 0,047 Z 0,046 K 0,045
T 0,080 D 0,068 C 0,064 G 0,059 P 0,057
H 0,077 I 0,054 L 0,051 Y 0,049 G 0,049
M 0,077 Z 0,067 Q 0,055 D 0,051 I 0,051
B 0,055 I 0,054 F 0,053 J 0,053 P 0,048
P 0,074 T 0,071 C 0,059 M 0,058 D 0,056
T 0,069 G 0,055 E 0,052 R 0,050 I 0,048
A 0,081 N 0,070 J 0,061 L 0,050 K 0,047
G 0,092 C 0,075 F 0,064 P 0,056 Q 0,048
S 0,079 W 0,066 J 0,060 F 0,055 N 0,054
S 0,061 F 0,060 R 0,053 W 0,052 J 0,047
E 0,080 R 0,056 D 0,054 I 0,052 V 0,049
I 0,070 M 0,062 W 0,056 L 0,052 P 0,051
I 0,089 E 0,059 H 0,053 R 0,053 W 0,051
A 0,080 E 0,079 N 0,074 R 0,055 G 0,052
A 0,067 K 0,061 B 0,054 O 0,050 Z 0,050
R 0,076 N 0,072 W 0,056 B 0,054 I 0,049
K 0,072 B 0,060 E 0,059 V 0,056 U 0,051
R 0,075 E 0,055 N 0,054 Q 0,050 D 0,049
Y 0,069 S 0,059 P 0,058 J 0,054 W 0,054
P 0,099 C 0,061 L 0,053 M 0,051 Y 0,049
T 0,092 X 0,071 G 0,061 K 0,050 H 0,045
O 0,080 K 0,068 B 0,066 F 0,063 S 0,047
L 0,059 P 0,056 W 0,055 M 0,051 X 0,050
O 0,083 K 0,068 B 0,058 G 0,056 N 0,052
G 0,080 C 0,068 K 0,062 D 0,055 R 0,050
I 0,083 E 0,059 Z 0,053 V 0,053 R 0,051
E 0,107 N 0,059 A 0,056 R 0,054 I 0,048
```

Die Buchstabenfolge in der ersten Spalte legt nahe, daß hier kryptographisch unsorgfältig gearbeitet wurde: Der Schlüssel wurde wohl nicht zufällig gewählt, sondern als sinnvoller Teil der deutschen Sprache, wobei wir aber nicht alle Buchstaben auf Anhieb richtig erraten haben.

Wir können nun entweder versuchen, den Schlüssel anhand der Buchstaben aus den folgenden Spalten zu erraten (was hier wohl selbst ohne diese nicht sonderlich schwerfällt), oder aber wir entschlüsseln einfach mit dem wahrscheinlich falschen Schlüssel und korrigieren anhand des entschlüsselten Textes. Wenn wir diesen in Zeilen der Länge 29 aufschreiben, stehen jeweils die Buchstaben, die zum gleichen Alphabet gehören, untereinander, so daß es für jeden Schlüsselbuchstaben mehrere Überprüfungsmöglichkeiten gibt.

Der Entschlüsselungsungsversuch führt auf folgendes Ergebnis: (In der ersten Zeile steht der Schlüssel.)

```
N A T H M B P T A G S S E I I A A R K R Y P T O L O G I E

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

Damit sollte wohl jeder Leser den Klartext rekonstruieren können. Bei einer Schlüssellänge von 29 und einer Textlänge von 480, bei etwa 16,5 Zeichen pro Alphabet also, ist das Vigenère-Verfahren somit schon völlig unsicher. Bei noch mehr Buchstaben pro Alphabet wird das Knacken noch einfacher: Betrachten wir als Beispiel denselben Klartext wie oben, chiffriert mit einem kürzeren Schlüssel:

```
PZWDV AKLNZ ZDMDE MGYNZ VNGSC DVFAD YTFDP PVKOS BFMYT SUDND
JOMAO MJVIQ BMQUQ MZAAV XNAEO UXQFX IDXNM UYHDR AFEHO AQVZN
JPRIQ EBUGC QGRVJ XPLXS IROTX LMMUF ZILPT KKIHM MHWXD NYIIJ
NESVD VTGDX ZZSOA JNJEU ZZLHQ LUXMA CLXJE EZPXQ NXUYJ IIBYW
```

```
ETCFN MSXZH FOPLS FPZFG GCUGR JWHIA OPQEY PTLUM MPHCN BKWAZ
IQGCQ KNZNY MUGGO TCXND ELQYN KTVSR WKCQF ZFBWZ WJLLX IEGGA
FHZYA MRVBP QJNNV QAQQG PYJMM YYYAE WQBCQ GEOZY QLTOW YMQLH
ZWMGQ ZDLXF JJOME SGGSZ SBMTK NJJVY
```

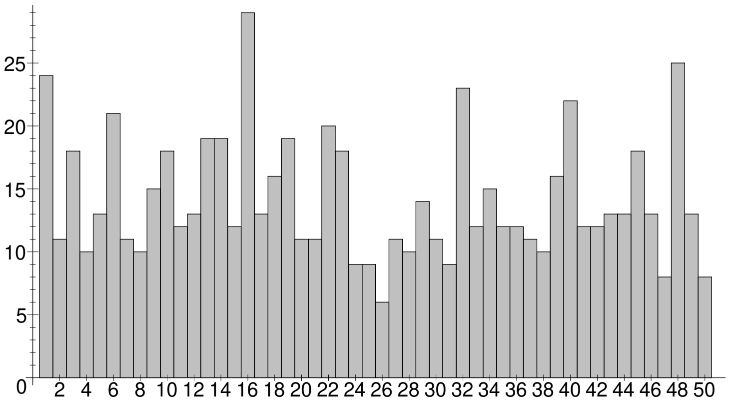


Abb. 6: Koinzidenzen bei kürzerer Schlüssellänge

Für dieses Kryptogramm zeigt Abbildung sechs die Verteilung der Koinzidenzen bei n Buchstaben Abstand. Die Spitze bei $n = 6$ kann ignoriert werden, da es bei $n = 12$ und $n = 24$ keine Maxima gibt; die Maxima bei $n = 16$, 32 und 48 lassen eine Periodenlänge von 16 als wahrscheinlich erscheinen. Es gibt zwar auch eine ziemlich hohe Spitze bei $n = 40$, was vielleicht doch auf eine Periode acht hindeutet, da allerdings weder $n = 8$ noch $n = 24$ zu sonderlich großen Häufigkeiten führen, spricht das Diagramm doch eher für $n = 16$. Der pq-Test schlägt folgende Schlüsselbuchstaben vor:

```
N 0,066 M 0,052 Q 0,050 R 0,048 J 0,046
E 0,079 A 0,056 F 0,055 I 0,050 N 0,050
U 0,079 H 0,058 E 0,057 Q 0,056 D 0,051
E 0,064 D 0,055 P 0,049 F 0,048 T 0,048
R 0,083 V 0,059 E 0,050 I 0,046 Q 0,045
S 0,087 B 0,051 O 0,049 R 0,048 D 0,048
T 0,077 P 0,070 G 0,068 F 0,047 K 0,045
```

```
U 0,067 H 0,052 L 0,052 Y 0,052 V 0,049
D 0,075 Z 0,060 Q 0,056 U 0,049 H 0,046
I 0,072 Z 0,065 M 0,055 E 0,047 N 0,046
E 0,069 R 0,058 I 0,049 P 0,048 F 0,047
N 0,084 R 0,053 J 0,049 W 0,048 A 0,047
G 0,076 K 0,050 Z 0,049 X 0,047 Y 0,046
A 0,078 W 0,073 B 0,063 F 0,051 N 0,050
N 0,090 J 0,053 W 0,052 A 0,047 R 0,042
G 0,091 X 0,057 T 0,052 K 0,048 F 0,047
```

Hier steht gleich in der ersten Spalte der sehr wahrscheinlich aussehende Schlüssel

`NEUERSTUDIENGANG` .

der auch in der Tat zu verständlichem Klartext führt (entnommen aus einer Informationsbroschüre der Fakultät für Mathematik und Informatik für Schüler):

> *Der integrierte Studiengang Mathematik und Informatik bietet Ihnen eine berufsorientierte wissenschaftliche Ausbildung in den Fächern Mathematik und Informatik. Erst nach zwei Jahren entscheiden Sie sich für eine der beiden Ausrichtungen „Mathematik“ und „Informatik“ und damit auch für einen der beiden Abschlüsse „Diplom-Mathematiker“ oder „Diplom-Informatiker“. Während der ersten beiden Studienjahre sind alle Veranstaltungen gemeinsam.*

Informationstheoretische Betrachtungen zeigen, daß schon bei weniger als zwei Buchstaben pro Alphabet der Klartext und der Schlüssel mit hoher Wahrscheinlichkeit eindeutig durch den Chiffretext bestimmt sind. Zu einer auf der Informationstheorie beruhenden Kryptanalyse benötigt man allerdings praktisch *unbeschränkte* Ressourcen, denn die Größen, mit denen hier gerechnet wird, sind definiert als Summen über den Raum *aller* möglicher Schlüssel; im Falle des obigen Schlüssels der Länge 29 wäre das eine Summe über $26^{29} \approx 10^{42}$ Summanden. Kryptanalytiker suchen daher nach weniger aufwendigen Verfahren – und finden Sie auch in vielen Fällen.

Bei VIGENÈRE-Chiffren muß man realistischerweise davon ausgehen, daß sie bereits anfangen unsicher zu werden, wenn der Klartext nur etwa 30% länger ist als der Schlüssel; wirklich sicher sind sie also praktisch nur in Form des *one time pad.* Wenn man nun bedenkt, daß es Programme gibt, die lange Dokumente oder gar eine gesamte Festplatte verschlüsseln in Abhängigkeit von einem nur wenige Zeichen langen Schlüssel mittels eines byte- statt buchstabenbasierten VIGENÈRE-Systems, wundert es nicht, daß so viele Programme auf dem Markt sind, die „vergessene“ Paßwörter rekonstruieren.

§5: Substitutionschiffren

Die Entschlüsselung von CAESAR- und VIGENÈRE-Chiffren wird vor allem dadurch erleichtert, daß es pro Alphabet nur 26 mögliche Schlüssel gibt. Selbstverständlich gibt es keinen Grund, warum man sich auf zyklische Permutationen der Buchstaben des Alphabets beschränken muß; betrachtet man stattdessen beliebige Permutationen, gibt es

$$26! = 403291461126605635584000000$$

Möglichkeiten, also 403 Quadrillionen 291 Trilliarden 461 Trillionen 126 Billiarden 605 Billionen 635 Milliarden und 584 Millionen. Wie schon GIROLOMAO CARDANO (1501–1576, berühmt unter anderem durch die Lösungsformel für die kubische Gleichung, an der er völlig unschuldig ist) bemerkte, ist diese Zahl so groß, daß selbst viele Bücher nicht ausreichten, um alle diese Permutationen zu fassen.

Heutzutage benutzen wir Computer zur Entschlüsselung, und die können deutlich mehr Information verarbeiten als selbst „viele“ Bücher zu fassen vermögen. Ein Computer mit einer Taktfrequenz von 1 GHz führt pro Sekunde eine Milliarde Zykeln aus, pro Stunde also 3,6 Billionen, pro Tag 86,4 Billionen, pro Jahr 31,5 Billiarden. Das Alter unseres Universums wird auf etwa zehn Milliarden Jahre geschätzt; falls ein solcher Computer diese ganze Zeit hindurch gerechnet hätte, hätte er also mittlerweile etwa 300 Quadrillionen Zykeln durchlaufen. Selbst unter der (völlig unrealistischen) Annahme, daß der Computer pro Zykluszeit einen Schlüssel durchprobieren kann, hätte er bis heute noch nicht alle Möglichkeiten erschöpft.

Da Kryptanalytiker nur selten alle möglichen Schlüssel durchprobieren, ist diese Rechnung natürlich ohne jegliche Relevanz für die Sicherheit der Chiffre; in der Tat lassen sich schon relativ kurze Kryptogramme mit Chiffretext allein entschlüsseln.

Dazu reichen nun allerdings nicht mehr die Häufigkeitsverteilungen der einzelnen Buchstaben, denn dazu gibt es doch zu viele Buchstaben mit sehr ähnlicher Häufigkeit. Abbildung sieben zeigt die bereits aus Abbildung eins bekannte Häufigkeitsverteilung bei Anordnung der Buchstaben nach abnehmender Häufigkeit; wie man sieht, sind die Unterschiede zwischen benachbarten Buchstaben oft so minimal, daß eine sichere Identifikation allein aufgrund dieser Unterschiede praktisch ausgeschlossen ist.

Abb. 7: Nochmals die Buchstabenhäufigkeiten in deutschem Text

Ein Buchstabe ist aber relativ gut charakterisiert durch seine Nachbarschaft: Auf ein „C“ folgt praktisch immer ein „H“, ein „D“ vor einem „E“ ist sehr viel häufiger als eines danach *usw.* Abbildung acht zeigt die in der Tat stark schwankenden Häufigkeiten von Buchstabenpaaren; die Achsen sind dabei der Einfachheit halber mit den Rangummern der Buchstabens im Alphabet beschriftet.

Das mag nicht besonders nützlich erscheinen, da schließlich die Nach-

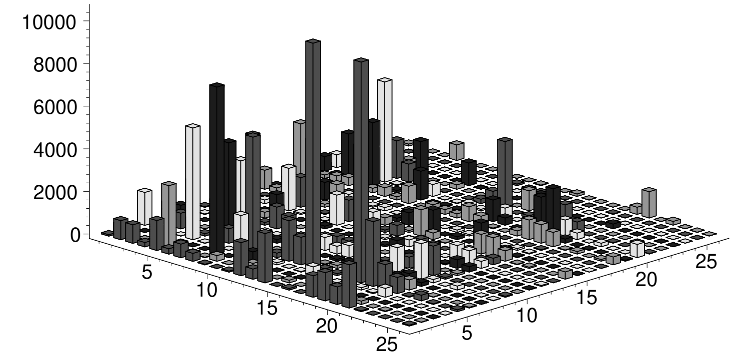


Abb. 8: Häufigkeiten von Buchstabenpaaren in deutschem Text

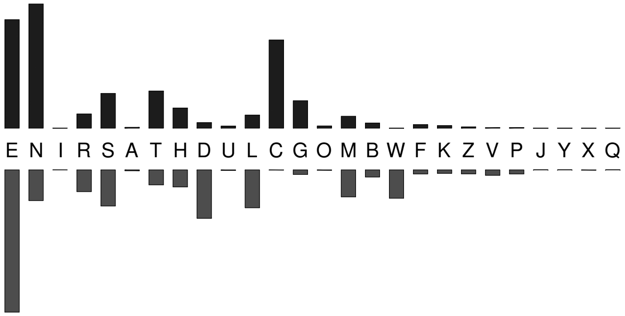
barn eines Buchstaben genausowenig bekannt sind wie der Buchstabe selbst. Bekannt ist aber die Rangordnung der Buchstaben in der Häufigkeitsverteilung des Kryptogramms, und diese sollte sich nur wenig von der der zugehörigen Klartextbuchstaben in einem Referenztext unterscheiden. Trägt man daher für jeden Buchstaben nach oben Balken auf, die angeben, wie oft der häufigste, zweithäufigste usw. Buchstabe *hinter* ihm steht und nach unten entsprechende Balken für die Buchstaben vor ihm, erhält man ein Doppelhistogramm, das sich zumindest für einigermaßen häufige Buchstaben nicht allzusehr unterscheiden sollte von dem des zugehörigen Klartextbuchstabens in einem Referenztext.

Für die praktische Anwendung dieser Idee brauchen wir zunächst die entsprechenden Doppelhistogramme für deutschen Klartext. Im folgenden sind für die 26 Buchstaben des Alphabets, nach Häufigkeit geordnet, die entsprechenden Kontaktdiagramme für *Dr. Katzenbergers Badereise* aufgezeichnet:

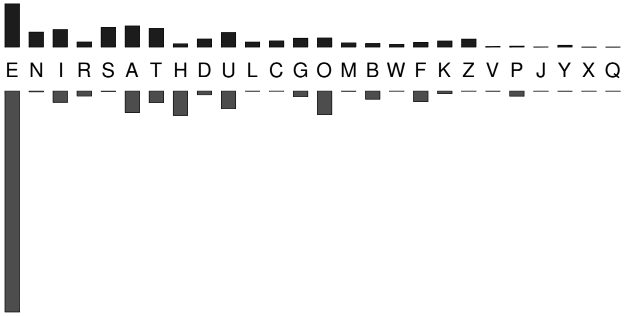
Kontaktdiagramm des Buchstabens E

Kontaktdiagramm des Buchstabens N

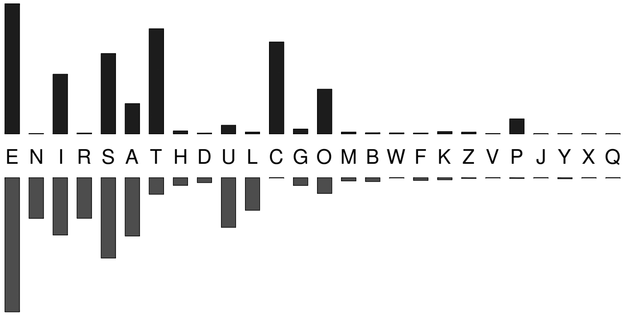
Kontaktdiagramm des Buchstabens I



Kontaktdiagramm des Buchstabens R

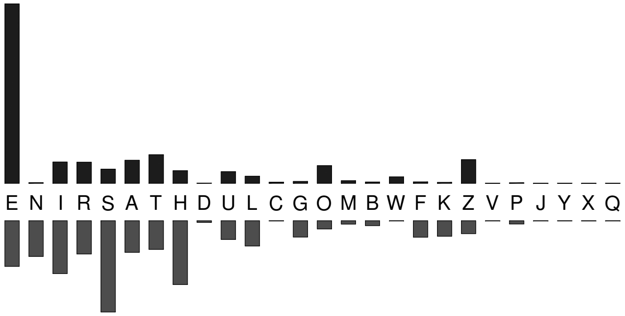


Kontaktdiagramm des Buchstabens S

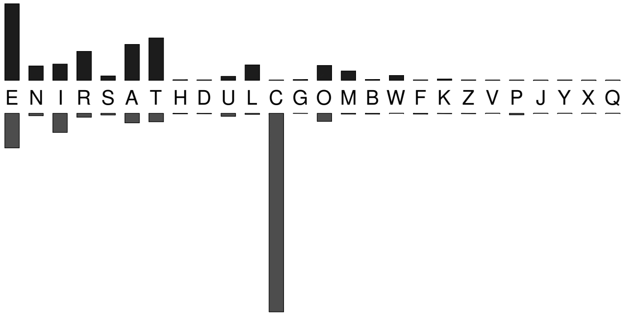


Kontaktdiagramm des Buchstabens A

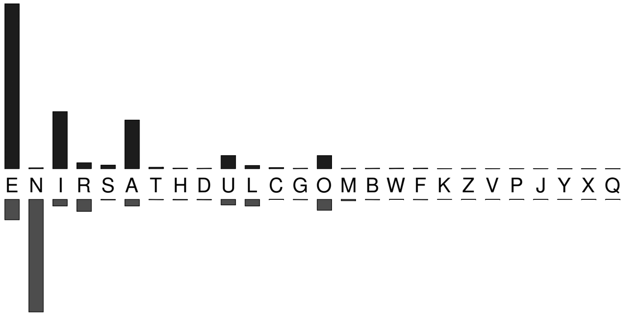
Kontaktdiagramm des Buchstabens T



Kontaktdiagramm des Buchstabens H

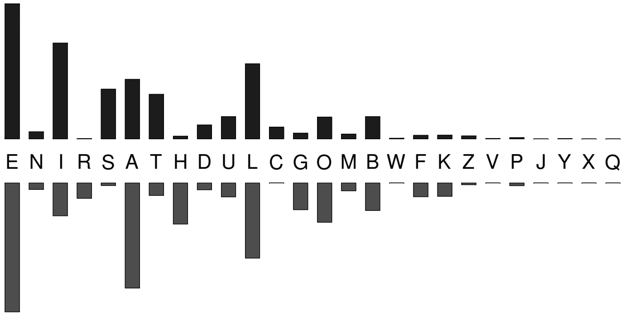


Kontaktdiagramm des Buchstabens D

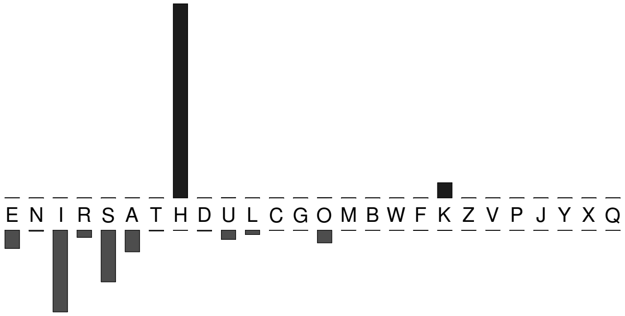


Kontaktdiagramm des Buchstabens U

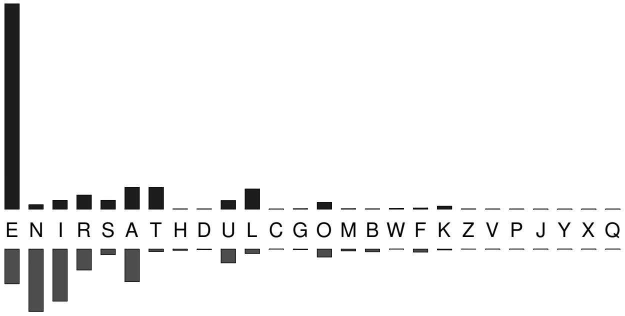
Kontaktdiagramm des Buchstabens L



Kontaktdiagramm des Buchstabens C

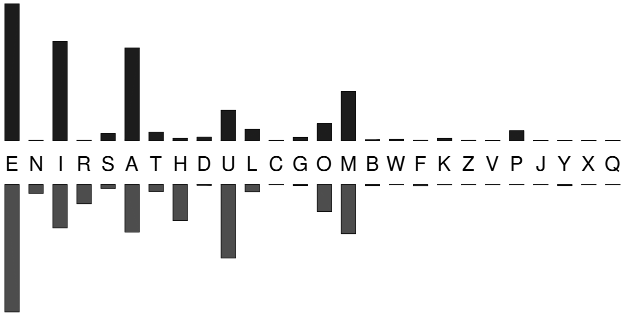


Kontaktdiagramm des Buchstabens G



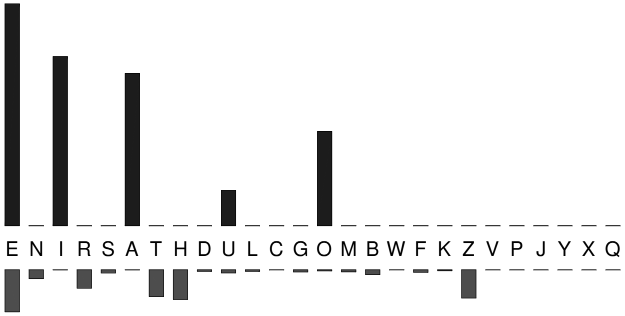
Kontaktdiagramm des Buchstabens O

Kontaktdiagramm des Buchstabens M

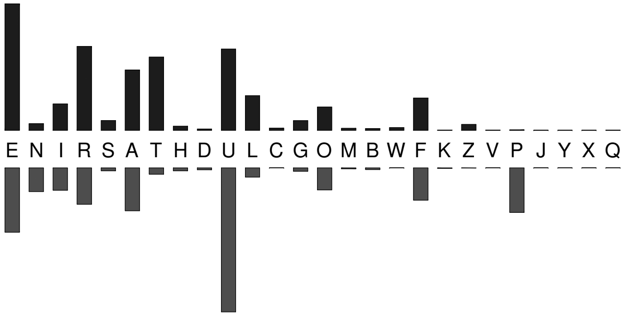


Kontaktdiagramm des Buchstabens B

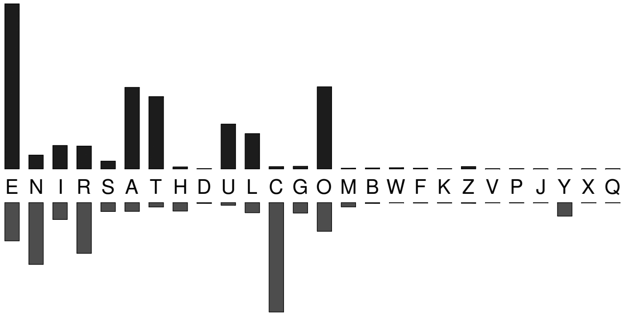
Kontaktdiagramm des Buchstabens W



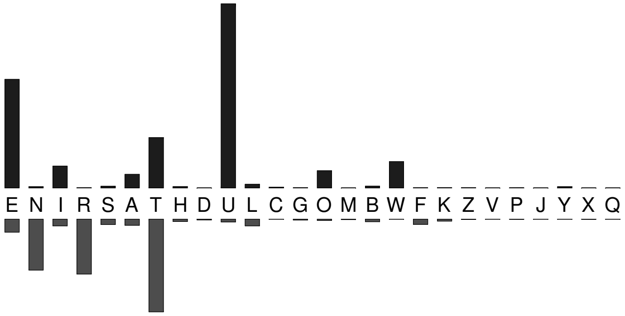
Kontaktdiagramm des Buchstabens F



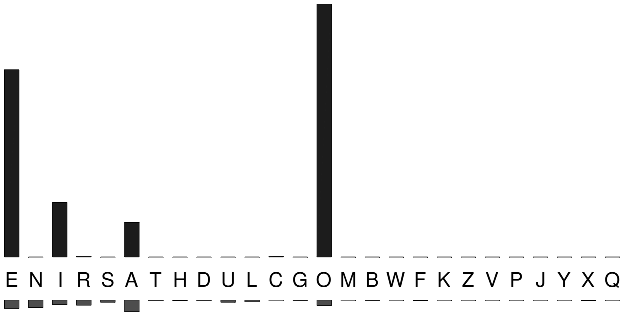
Kontaktdiagramm des Buchstabens K

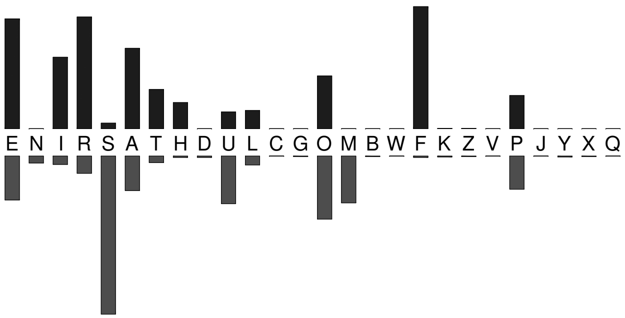


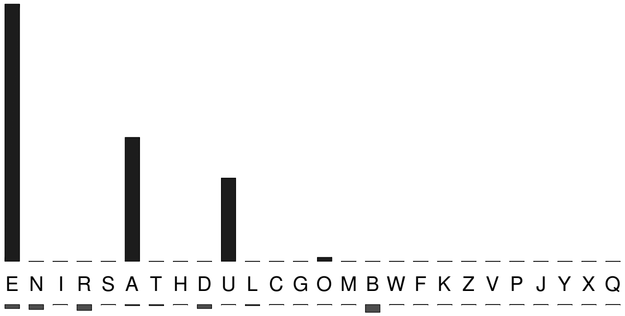
Kontaktdiagramm des Buchstabens Z

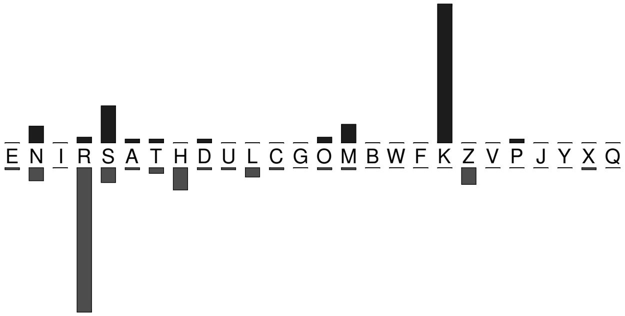


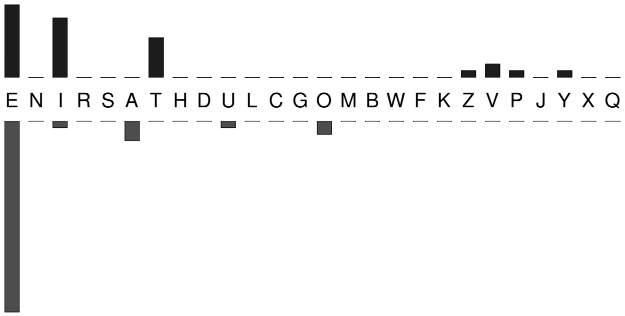
Kontaktdiagramm des Buchstabens V




Kontaktdiagramm des Buchstabens P


Kontaktdiagramm des Buchstabens J


Kontaktdiagramm des Buchstabens Y


Kontaktdiagramm des Buchstabens X


Kontaktdiagramm des Buchstabens Q

Wie man sieht, verhalten sich die Buchstaben in der Tat sehr verschieden: „E“ tritt besonders gerne mit „N“ und “R“ auf, wobei beide eher hinter als vor dem „E“ stehen; „R“ hat in der überwiegenden Anzahl der Fälle ein „E“ vor sich, während die Buchstaben dahinter sehr viel gleichmäßiger verteilt sind; „S“ tritt vor und nach fast allen der häufigen Buchstaben auf, und auch „A“ hat unter diesen nur wenig ausgeprägte Präferenzen; „C“ und „H“ charakterisieren sich gegenseitig wegen der häufigen „CH“ und „SCH“ und so weiter.

Aufgrund seiner Erfahrung mit der deutschen Sprache wird wohl jeder leicht Erklärungen für die genannten Phänomene finden; etwas seltsam erscheint aber, daß beispielsweise „Y“ vorzugsweise auf „R“ folgt und ein „K“ hinter sich stehen hat. Dies ist in der Tat im Bereich der Zufallsschwankungen und nur mit Kenntnis des ausgezählten Textes verständlich: Der Zweck von Dr. Katzenbergers Badereise bestand darin, den Rezensenten seines Buchs zu verprügeln, einen Badearzt namens Strick, der sich St*ryk*ius nennt.

Für seltene Buchstaben sind obige Diagramme also mit Vorsicht zu genießen, wenn auch natürlich der allerseltenste Buchstabe „Q“ in der Tat durch das nachfolgende „U“ charakterisiert ist.

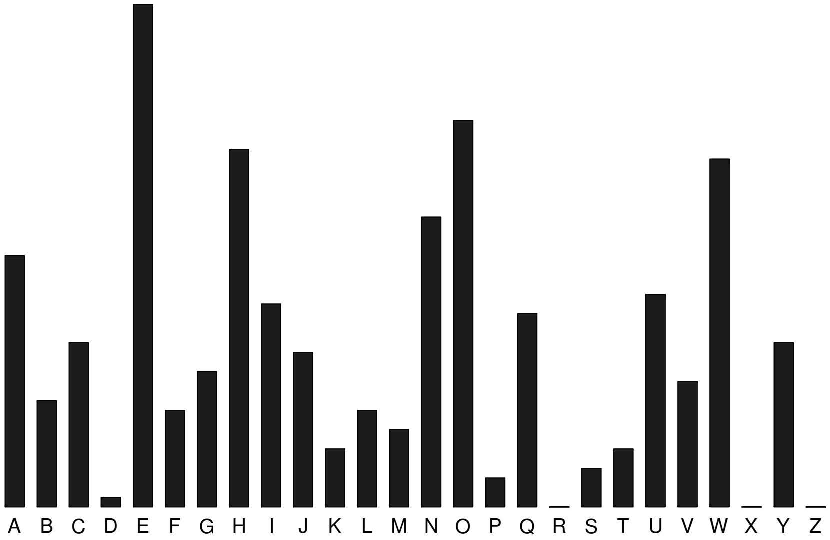
Für die häufigeren Buchstaben aber sind die Diagramme eine gute Charakterisierung; man muß nur bedenken, daß es bei einem kurzen Kryptogramm natürlich Schwankungen nicht nur in der relativen Höhe der Balken gibt, sondern auch bei der Rangfolge: Wie wir bereits bei den CAESAR-Chiffren gesehen haben, ist „E“ nicht immer der häufigste Buchstabe.

Betrachten wir als Beispiel das Kryptogramm

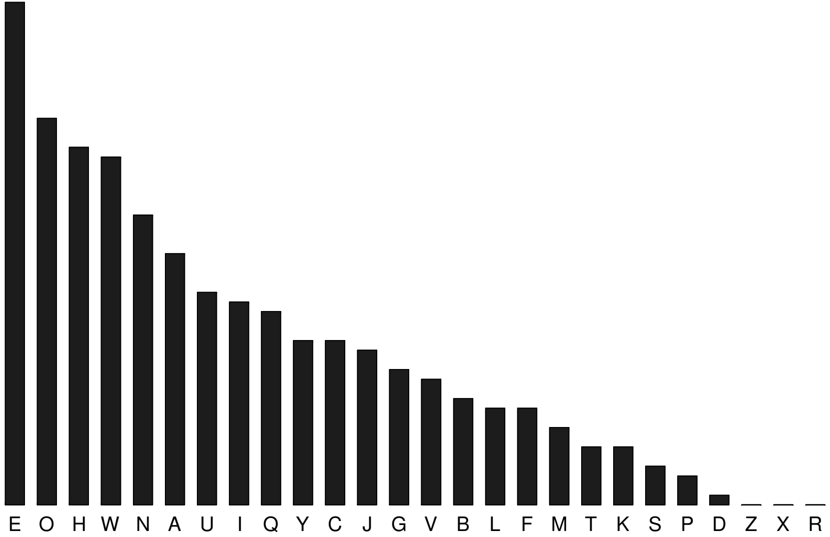
```
KAPUY WOYEU NAWON QANSW HEUIE ONGQE BBHWQ OWOON
EEUIW HEGQH LOVFM EOFWY NGQUE BGALG QNFME OFWYV
EKAOI WHWOQ EUHMA TCEOT WNHLO VNAYE CQEUL HAHIE
OYWLH WQIHC EODLW UWHEO HWEIW HIACV NMWCT LOYUE
CYEGH PUSCA GEUIE VWAUE HECEW EIHEY NEQEO HAOKA
PUOLW YLANN MAOYE OYIEG QHKAO NEEUF LNEEU IWHWQ
IQABN HEEHL OVHCA NNKAI QWUBN ECFHL WNFLI ECFHN
HWBHL WYLAN NHCWO THIWU GQLOV NSCWG QHVEF LYCWO
NHLOV UEGQH IEOGQ IEUVE OOLOV MEOOY WOYLW YLANN
LWNFL CNGQU EBNHE HHKAO IIEOO MAHAV TCEOT MECLO
VELBI LUHWI AUAGQ SBWBB
```

Die Häufigkeitsverteilung der Buchstaben ist in Abbildung neun zu sehen; sie ist offensichtlich nicht einfach eine verschobene Häufigkeitsverteilung für Klartext, wie wir sie bei CAESAR-Substitutionen hatten.

Abbildung zehn zeigt aber, daß zumindest die sortierte Häufigkeitsverteilung doch recht gut der in Abbildung sieben dargestellten entspricht; man kann also davon ausgehen, daß es sich bei Kryptogramm um eine monoalphabetische Substitution handelt.



Buchstabenhäufigkeiten im Kryptogramm



SortierteBuchstabenhäufigkeiten im Kryptogramm

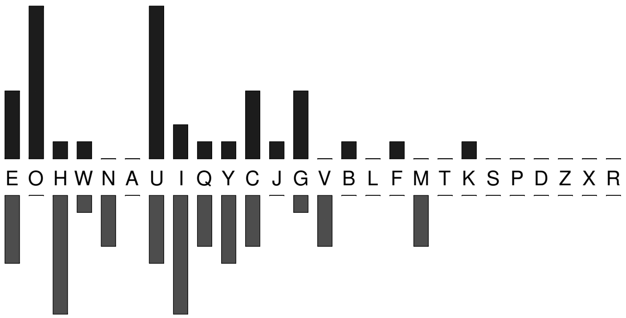
Eine Probeentschlüsselung einfach aufgrund des Vergleichs von Abbildung zehn mit Abbildung sieben führt auf folgenden Text:

```
ZAPTU RNUET SARNS DASVR IETHE NSGDE MMIRD NRNNS
EETHR IEGDI CNOWF ENWRU SGDTE MGACG DSWFE NWRUO
EZANH RIRND ETIFA KLENK RSICN OSAUE LDETB IAIHE
NURBI RDHIL ENJCR TRIEN IREHR IHALO SFRLK CNUTE
LUEGI PTVLA GETHE ORATE IELER EHIEU SEDEN IANZA
PTNBR UBASS FANUE NUHEG DIZAN SEETW CSEET HRIRD
HDAMS IEEIC NOILA SSZAH DRTMS ELWIB RSWCH ELWIS
IRMIB RUBAS SILRN KIHRT GDCNO SVLRG DIOEW CULRN
SICNO TEGDI HENGD HETOE NNCNO FENNU RNUBR UBASS
BRSWC LSGDT EMSIE IIZAN HHENN FAIAO KLENK FELCN
OECMH CTIRH ATAGD VMRMM
```

Das ist ganz offensichtlich kein deutscher Klartext, und auch mit viel Sprachgefühl fällt es schwer, hier Kandidaten für deutsche Wörter zu sehen. Wir müssen also, zumindest für die häufigeren Buchstaben, deren Vorgänger und Nachfolger betrachten.

Der häufigste Buchstabe ist, wie wir das auch von deutschem Klartext gewohnt sind, das „E“; betrachten wir sein Kontaktdiagramm:

Kontaktdiagramm des Buchstabens E



Schon ein flüchtiger Blick zeigt, daß sich dieses Diagramm in wesentlichen Punkten von dem des Klartext- „E“ unterscheidet, denn dieses

kommt nur selten verdoppelt vor, und hat in erster Linie sehr häufige Buchstaben hinter sich stehen.

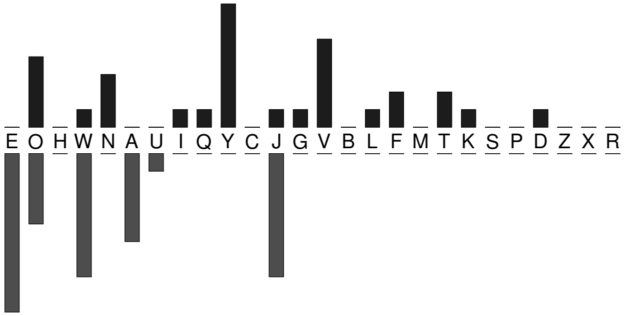
Auch um ein „N“ kann es sich nicht handeln, denn dazu gibt es zu viele Balken nach unten. „I“ ist unmöglich, da Doppel-I praktisch nicht vorkommt, „R“ mit seinem sehr charakteristischen Kontaktdiagramm ist es auch nicht, ebensowenig „S“, das sowohl vor als auch nach sich eher etwas häufigere Buchstaben bevorzugt.

Die Übereinstimmung mit „A“ ist ebenfalls alles andere als optimal, aber immerhin kommt „A“ etwas häufiger als die anderen Vokale verdoppelt vor und hat sehr häufig den zweithäufigsten Buchstaben sowie Buchstaben mittlerer Häufigkeit nach sich bei wenig ausgeprägten Präferenzen für die Vorgänger.

Die Diagramme der nächsten einigermaßen häufigen Buchstaben in deutschem Klartext passen wieder deutlich schlechter. Da der häufigste Buchstabe des Kryptogramms wohl schon unter den zehn häufigsten Buchstaben der deutschen Sprache zu finden sein sollte und mit relativ hoher Wahrscheinlichkeit ein Vokal sein dürfte, spricht alles in allem gesehen doch vieles für „A“. Zumindest falls der Chiffrebuchstabe „O“ für „N“ stehen sollte, paßt das Diagramm einigermaßen zum „A“.

Die (vorläufig) endgültige Entscheidung sollten wir also erst fällen, wenn wir auch das Kontaktdiagramm des zweithäufigsten Chiffrebuchstabens „O“ betrachtet haben.

Kontaktdiagramm des Buchstabens O



Dieses Diagramm entspricht in der Tat recht gut dem des Klartextbuchstabens „N“; es spricht also vieles für die Entschlüsselung

$$\mathrm{E} \to \mathrm{A}, \quad \mathrm{O} \to \mathrm{N}\,.$$

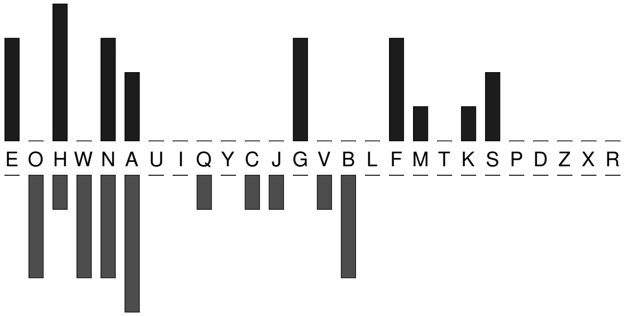
Kontaktdiagramm des Buchstabens H

Der dritthäufigste Kryptogrammbuchstabe „H“ tritt sehr häufig verdoppelt auf, kann also kein „E“, „I“ oder „R“ sein. „S“ und „T“ sind Möglichkeiten, wobei „S“ auf den ersten Blick optisch günstiger aussieht. Bedenkt man aber, daß der erste Balken bei den einen Diagrammen für „E“ und bei den anderen für „A“ stehen, spricht doch vieles eher für „T“: „S“ steht häufiger nach „A“ als davor, bei „T“ ist es eher umgekehrt; auch die relativ scharf ausgeprägten oberen Balken beim „S“ passen nicht richtig. Gegen „T“ spricht vor allem der auffällige „E“-Balken, aber da wir bereits beim dritthäufigsten Buchstaben des Kryptogramms sind und noch immer kein „E“ gefunden haben, ist dieses hier wohl doch etwas unterrepräsentiert.

Kontaktdiagramm des Buchstabens W

Das „W“ im Kryptogramm hat gerne ein Klartext-„N“ nach sich und ein Klartext-„T“ sowohl vor als auch nach sich, tritt praktisch nie verdoppelt auf und hat einen Buchstaben mittlerer Häufigkeit, der oft danach steht; all dies spricht sehr für ein „I“.

Kontaktdiagramm des Buchstabens N



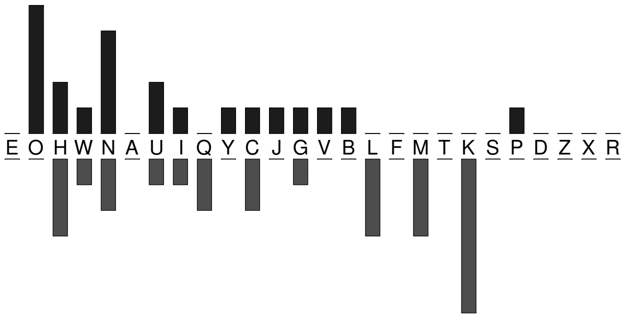
Der Klartextbuchstabe zu „N“ wird oft von „A“, „T“ und sich selbst gefolgt und steht oft nach „N“; damit dürfte fast klar sein, daß es sich um ein „S“ handeln muß.

Damit haben wir mit

$$E \to A, \quad O \to N, \quad H \to T \quad \text{und} \quad N \to S$$

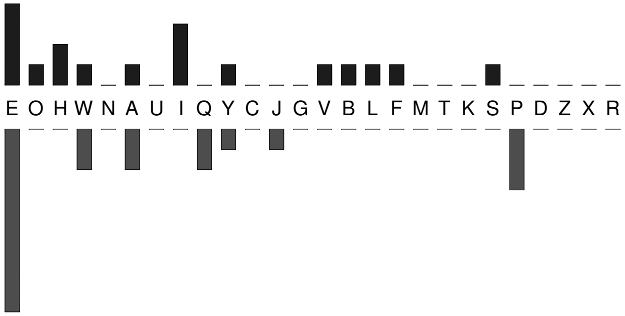
die fünf häufigsten Buchstaben des Kryptogramms entschlüsselt und können zumindest für die mit den jeweils korrekten Balken der Klartextdiagramme vergleichen.

Kontaktdiagramm des Buchstabens A



Das Diagramm zum Chiffretext-„A“ gibt uns gleich zwei Identifizierungen: Ein Buchstabe, der weder mit sich selbst noch mit dem Klartext-„A“ zusammen auftritt, dafür aber häufig ein „N“ oder „S“ nach sich führt, „T“ zu beiden Seiten und einen eher seltenen Buchstaben oft vor sich, kann wohl nur „O“ sein, und das häufig davorstehende „K“ ist somit ein „V“.

Kontaktdiagramm des Buchstabens U



„U“ wird entschlüsselt durch einen Buchstaben, der die bereits bekannten Vokale gerne sowohl vor als auch hinter sich hat, sich selbst allerdings praktisch nie, die Konsonanten „N“, „S“ und „T“ stehen häufiger dahinter, aber praktisch nie davor, und eine ganze Reihe eher seltener Buchstaben steht immer wieder einmal dahinter – hier fällt eine klare Identifikation schwer.

Wir können aber schauen, ob wir mit den bereits gefundenen acht Klartextbuchstaben schon Wörter im Kryptogramm erkennen können:

```
KAPUY WOYEU NAWON QANSW HEUIE ONGQE BBHWQ OWOON
VO    IN A  SOINS  OS I TA  A NS  A   TI  NINNS
EEUIW HEGQH JOVFM EOFWY NGQUE BGAJG QNFME OFWYV
AA  I TA  T  N    AN I  S   A   O    S  A N I
EKAOI WHWOQ EUHMA TCEOT WNHJO VNAYE CQEUL HAHIE
AVON  ITIN  A T O   AN  IST N  SO A   A   TOT A
OYWLH WQIHC EODJW UWHEO HWEIW HIACV NMWCT JOYUE
N I T I  T  AN  I  ITAN TIA I T O   S I    N  A
CYEGH PUSCA GEUIE VWAUE HECEW EIHEY NEQEO HAOKA
A T     O  A  A  IO A TA AI A TA  SA AN TONVO
PUOLW YLANN MAOYE OYIEG QHKAO NEEUF JNEEU IWHWQ
N I   OSS  ON A N  A   TVON SAA    SAA   ITI
IQABN HEEHJ OVHCA NNKAI QWUBN ECFHL WNFJI ECFHN
O S TAAT  N T O SSVO   I  S A  T  IS    A  TS
HWBHL WYLAN NHCWO THIWU GQJOV NSCWG QHVEF JYCWO
TI T  I  OS ST IN  T I     N  S  I    T A     IN
NHJOV UEGQH IEOGQ IEUVE OOJOV MEOOY WOYLW YLANN
ST N   A  T  AN    A  A NN N   ANN  IN  I   OSS
LWNFJ CNGQU EBNHE HHKAO IIEOO MAHAV TCEOT MECJO
IS    S    A STA TTVON   ANN  OTO     AN   A  N
VEJBI JUHWI AUAGQ SBWBB
A      TI  O O     I
```

Gleich in der ersten Zeile springt „a-so ins“ ins Auge, also steht wohl „U“ für „L“. Damit wir das nächste Wort zu „-os-ital“, d.h. „Q“ steht für „H“ und „S“ für „P“.

Nachdem wir zusätzlich zum schon länger bekannten „S“ auch das „H“ kennen, können wir nach „SCH“ suchen; ein Kandidat dafür ist die fünfte Gruppe der zweiten Zeile, was „G“ als Verschlüsselung von „C“ ergibt.

Damit wird der Text schon stellenweise verständlich: Das zweite Wort muß wohl „ging“ sein, vor dem „avon“ zu Beginn der dritten Zeile kann wohl nur ein „D“ stehen und so weiter. Der Rest ist ein reines Puzzlespiel und führt zum Klartext

> *Voyl ging also ins Hospital. Man schafft ihn in'n Saal mit achtundzwanzig Schlafcouchs, ...zwanzig davon mit Inhalt, wo krank ist und sogar halbtot. Man gibt ihm Tranquilitantia mit Mordswirkung (Largactyl, Procalmadiol, Atarax). Am Tag sah Anton Voyl 'n Big Boss, wo'n Gang macht von Saal zu Saal; mit ihm Hofstaat und Tross vom Hilfsarzt bis zum Arztstift; Big Boss trinkt Milch und spricht dazu, grinst und lacht manchmal ... Dann und wann ging Big Boss bis zur Schlafstatt von'm Mann, wo todkrank war und aufm ultimo Loch pfiff ...*

Der Text stammt aus der deutschen Übersetzung des Romans *La disparaition* von GEORGE PEREC *(Edition Denoël, Paris,* 1969*).* Dieser französische Autor zahlreicher Romane, hatte nach einem feuchtfröhlichen Abend gewettet, daß er einen Roman schreiben könne, ohne den Buchstaben „E“ zu verwenden; er gewann. Die deutsche Übersetzung von EUGEN HELMLÉ mit dem Titel *Anton Voyls Fortgang (Zweitausendeins,* ^{3}1998*)* hält sich auch an die Vorgabe, verwendet aber Umlaute – daher die Auslassungspunkte im obigen Text.

Wie man sieht, lassen sich die einzelnen Buchstaben also selbst dann noch recht gut anhand ihrer Umgebung identifizieren, wenn ihre Rangfolge nicht mehr ganz der für deutsche Texte üblichen entspricht.

Eine hier nicht behandelte Möglichkeit, sich Buchstaben zu verschaffen, liegt auch im Erraten von Wörtern aus der Botschaft. Da man beim Abhören meist weiß, wonach man sucht, gibt es oft Wörter, die mit hoher Wahrscheinlichkeit im Text vorkommen sollten. Falls ein solches Wort einen oder gar mehrere Buchstaben mehrfach enthält, kann man im Chiffretext nach demselben Muster suchen. Ein ideales Wort wäre beispielsweise *Substitution:* Es muß von jeder monoalphabetischen Substitution in eine Folge der Art

$$XYpXZWZYZWqr$$

transformiert werden, und eine Folge dieser Art kann praktisch nur für das Wort *Substitution* stehen. Damit sind sieben Buchstaben identifiziert.

§6: Polyalphabetische Substitutionen

Als nächste Stufe der Komplexität bietet sich natürlich an, aufeinanderfolgende Buchstaben mit verschiedenen Permutationen zu verschlüsseln, d.h. also eine Art von VIGENÈRE-Verfahren mit allgemeinen monoalphabetischen Substitutionen anstelle der doch sehr speziellen CAESAR-Substitutionen.

Leider steigt dadurch, sofern die Anzahl der Permutationen nicht in die Größenordnung der Nachrichtenlänge kommt, die Sicherheit nur unwesentlich an, denn die Verfahren aus dem vorigen Paragraphen funktionieren auch für diesen Fall: Man stellt zunächst, genau wie bei den VIGENÈRE-Chiffren, die Periode fest; danach weiß man, welche Buchstaben mit derselben Permutation verschlüsselt wurden. Jede dieser Buchstabenmengen wird nach Häufigkeit geordnet, und man kann wieder Kontaktdiagramme aufstellen. Dabei muß nur beachtet werden, daß die rechten und linken Nachbarn nicht aus derselben Gruppe kommen, sondern aus je einer anderen festen Gruppe, so daß Balken nach oben und nach unten nicht mehr für denselben Buchstaben stehen müssen. Abgesehen von dieser kleinen Erschwernis ändert sich nichts, und ein erfahrener Kryptanalytiker kann ab etwa 30–40 Buchstaben pro Alphabet praktisch sicher sein, daß er die Nachricht entschlüsseln kann.

Polyalphabetische Chiffren mit sehr großen Periodenlängen erzeugten die sogenannten *Rotormaschinen,* die in den Zwanzigerjahren aufkamen und die Kryptographie des zweiten Weltkriegs dominierten. Ein Rotor ist eine Scheibe, die auf jeder Seite in gleichmäßigen Abständen 26 Kontakte hat. Jeder Kontakt auf der Vorderseite ist mit einem der Kontakte auf der Rückseite verbunden, wobei nur darauf zu achten ist, daß diese Verbindungen spiegelsymmetrisch angebracht werden, so daß die so realisierte Permutation ein Produkt von dreizehn Transpositionen ist.

Zum Einsatz der Maschine wählt man, je nach Maschine, drei bis fünf von mehreren verfügbaren Rotoren aus und setzt diese in einer festzulegenden Reihenfolge und Anfangsstellung in die Maschine ein. Der erste Buchstabe wird dann von einem festen Kontakt auf der einen Seite der Maschine durch die Rotoren in dieser Stellung an einen der 26 festen Kontakte auf der anderen Seite der Maschine übertragen; das Chiffrierergebnis ist der diesem Kontakt zugeordnete Buchstabe. Danach bewegen sich die Rotoren automatisch in einer nicht ganz regelmäßigen Weise weiter, so daß für den nächsten Buchstaben eine andere Permutation realisiert wird und so weiter. Zur Entschlüsselung wird der Chiffretext einfach in eine identisch aufgebaute Maschine mit gleicher Anfangsstellung aber umgekehrter Rotorreihenfolge gegeben.

(Unter http://library.thinkquest.org/28005/flashed/timemachine/courseofhistory/enigmaapp.shtml kann man ausprobieren, wie das bei er deutschen Enigma-Maschine funktionierte.)

Die mit solchen Maschinen erzielbaren Perioden liegen im Bereich über 100 000; trotzdem wurden sie schon im zweiten Weltkrieg häufig geknackt.

Ein Grund dafür ist, daß während der Gültigkeitsperiode eines Schlüssels alle Nachrichten mit derselben Anfangsstellung verschlüsselt werden müssen (oder aber die Anfangsstellung auf eine kryptographisch unsichere Weise übermittelt werden muß, was sich gerade bei der deutschen Enigma als noch schlimmer herausstellte) so daß damit zumindest für die ersten Buchstaben relativ viele Chiffrebuchstaben zusammenkommen. Das weitere Problem, daraus die Anfangsstellung der Maschine zu ermitteln, lösten die Engländer im zweiten Weltkrieg mit speziell gebauten Maschinen, den sogenannten „Bomben“, die mechanisch alle möglichen Anfangsstellungen durchprobierten. Die Verdrahtung der einzelnen Rotoren war schon vor dem Krieg von den Polen geknackt worden.

Seit etwa 1970 werden Rotormaschinen nicht mehr eingesetzt – außer in UNIX. Das dortige `crypt(1)`-Kommando simuliert eine Rotormaschine mit *einem* Rotor, der 256 Ein- und Ausgänge hat. Sehr sicher ist das nicht, und auch UNIX selbst verwendet beispielsweise für Paßwörter das auf einem „gesalzenen“ DES beruhende deutlich bessere `crypt(3)`-Unterprogramm.

§7: Literaturhinweise

Das (dickleibige) Standardwerk zur alten Kryptographie mit Schwerpunkt auf der geschichtlichen Darstellung ist

DAVID KAHN: *The Codebreakers – the comprehensive history of secret communication from ancient time to the internet,* Scribner, New York, ^{2}1996

Abgesehen von einem Anhang über public key Kryptographie ist das Buch weitgehend identisch mit der ersten Auflage von 1967.

Deutlich kürzer und billiger, mit Schwerpunkt auf der Darstellung der Verfahren, ist

HELEN FOUCHÉ GAINES: *Cryptanalysis – a study of ciphers and their solution,* Dover, New York, 1956 *(Nachdruck einer Originalausgabe von 1939)*

Rotormaschinen und ihrer Kryptanalyse gewidmet ist das Buch

CIPHER A. DEAVOURS, LOUIS KRUH: *Machine cryptography and modern cryptanalysis,* Artech House, Dedham MA, 1985

Das Wort „modern“ bezeichnet hier den Zeitraum von etwa 1920–1970.

Ein wirklich modernes Lehrbuch mit Kapitel über klassische Kryptographie sowie über statistische und informationstheoretische Ansätze zur Kryptanalyse ist

JAN C.A. VAN DER LUBBE: *Basic Methods of cryptography,* Cambridge University Press, 1998

Kapitel 2
Informationstheoretische Ansätze

Im vorigen Kapitel haben wir anhand einiger Beispiele gesehen, wie die statistischen Eigenschaften eines Kryptogramms zu Informationen über den Klartext und/oder den Schlüssel führen können. Dabei wurde klar, daß gewisse Verfahren trotz auf den ersten Blick gewaltiger Komplexität bereits für relativ kurze Nachrichten unsicher sind.

Bei der Wahl eines Kryptosystems interessiert vor allem die umgekehrte Frage: Wie *sicher* ist das System? Es hieße, den Gegner möglicherweise gewaltig zu unterschätzen, wenn man dabei nur die Sicherheit gegenüber den Angriffsmöglichkeiten betrachtet, die man selbst kennt: Ein ernstzunehmender Gegner wird häufig über Methoden verfügen, die nicht in der offenen Literatur dokumentiert sind.

In diesem Kapitel wollen wir daher untersuchen, wieviel Information über Schlüssel und Klartext in einem Kryptogramm enthalten ist – unabhängig davon, ob ein Gegner in der Lage ist, diese Information mit realistischem Aufwand nutzbar zu machen.

Man kann das auch so formulieren, daß wir den Gegner deutlich *über*schätzen, indem wir annehmen, daß ihm unbegrenzte Resourcen zur Verfügung stehen; er kann also *jeden* endlichen Algorithmus ausführen, unabhängig von physikalischen und sonstigen Einschränkungen. Insbesondere könnte er also eine Substitutionschiffre einfach dadurch lösen, daß er alle 26! Möglichkeiten ausprobiert; sofern nur in einem Fall sinnvoller Klartext entsteht, hat er die Lösung gefunden. Es ist klar, daß von einem realen Gegner keine größere Gefahr ausgeht, als von diesem idealisierten Gegner; ein Kryptosystem ist also sicher, wenn es gegen einen solchen Gegner sicher ist.

Um diese Sicherheit abschätzen zu können, müssen wir zunächst klären, was wir unter der in einem Klartext oder Kryptogramm enthaltene Information verstehen wollen; damit beschäftigt sich der nächste Paragraph.

§1: Die Entropie einer Quelle

Gegeben sei eine Nachricht aus N Buchstaben $x_1, \ldots, x_N$, die einem Alphabet A entstammen. Das Alphabet A kann dabei beispielsweise wie im vorigen Kapitel aus den 26 Buchstaben von „A“ bis „Z“ bestehen, es könnte sich aber auch um die 256 möglichen ASCII-Codes handeln oder um die Codewörter eines der alten Telegrammcodes für Geschäftsleute.

Wieviel Information enthält diese Nachricht für den Empfänger? Diese Information hängt offensichtlich ab von seinem Wissen über die Eigenschaften der Quelle: Bei einer Quelle, die deutschen Klartext sendet, wird er beispielsweise nach einem „C“ ein „H“ erwarten, so daß dieses, wenn es wirklich kommt, seine Information über die Nachricht nicht wesentlich erhöht; ein „I“ dagegen wäre unerwartet und damit informativ.

In den nächsten drei Abschnitten wollen wir dies genauer untersuchen für drei verschiedene statistische Modelle für die deutsche Sprache oder irgendwelchen anderen Klartext.

Als *Informationsgehalt* der Nachricht bezeichnen wir dabei die minimale Anzahl von Bits, die der Empfänger braucht, um die Nachricht zu speichern. Diese Anzahl hängt natürlich davon ab, welches Vorwissen er über die Nachricht hat; je mehr er darüber weiß, desto effizienter kann er sie abspeichern.

a) Ohne Vorwissen über die Quelle

Falls der Empfänger nichts über die statistische Verteilung der Buchstaben weiß, hat jeder Buchstabe dieselbe Wahrscheinlichkeit und enthält somit dieselbe Information. Es gibt daher $|A|^N$ verschiedene Möglichkeiten, von denen jede gleich wahrscheinlich ist.

Um einen einzigen Buchstaben binär zu speichern, brauchen wir $\lceil \log_2 |A| \rceil$ bit, wobei $\lceil x \rceil$ für die kleinste ganze Zahl $\geq x$ steht. Im Falle eines Alphabets aus 26 Buchstaben wären dies wegen

$$\log_2 26 \approx 4{,}70044$$

als fünf Bit.

Das ist aber noch nicht optimal: Wenn wir etwa die Buchstaben der Nachricht in Siebenergruppen zusammenfassen, brauchen wir pro Gruppe nicht 35, sondern nur 33 Bit, denn

$$7 \log_2 26 \approx 32{,}90308\,,$$

so daß pro Buchstabe nur noch

$$\frac{33}{7} \approx 4{,}71428$$

Bit benötigt werden. Mit noch größeren Gruppen kämen wir noch näher an $\log_2 26$ heran, und da es in diesem Kapitel nur ums Prinzip geht, nicht um Praktikabilität, wollen wir diese Zahl als Informationsgehalt pro Buchstabe betrachten.

Allgemein bezeichnen wir also den Zweierlogarithmus $N \log_2 |A|$ von $|A|^N$ als Informationsgehalt der Nachricht und $\log_2 |A|$ als den Informationsgehalt pro Buchstabe.

b) Mit Vorwissen über die Häufigkeitsverteilung

Falls der Empfänger die Häufigkeitsverteilung der Buchstaben kennt, weiß er bereits, wie oft jeder Buchstabe vorkommt: Ist p_i die Häufigkeit des i-ten Buchstaben, so kommt dieser $p_i N$-mal vor. Damit sind also alle Buchstaben der Nachricht bekannt; es geht „nur“ noch um ihre Reihenfolge. Im Prinzip gibt es $N!$ Möglichkeiten, die Reihenfolge von N Buchstaben festzulegen; hiervon sind allerdings alle diejenigen uninteressant, die nur Buchstaben miteinander vertauschen, die ohnehin schon gleich sind. Für den i-ten Buchstaben, der $p_i N$-mal vorkommt, sind dies $(p_i N)!$ Stück; tatsächlich gibt es also nur

$$\frac{N!}{(p_1 N)! \cdots (p_m N)!}, \qquad m = |A|$$

Möglichkeiten, die aus Sicht des Empfängers wieder allesamt gleich wahrscheinlich sind. Der Informationsgehalt der Nachricht ist deshalb gleich dem Zweierlogarithmus dieser Zahl.

Um diesen Logarithmus weiter auszurechnen, erinnern wir uns an die STIRLINGsche Formel, wonach für eine natürlich Zahl n gilt

$$\ln n! = n \ln n - n + \frac{\ln n}{2} + O(1)\,,$$

wobei $O(1)$ bedeuten soll, daß dieser Term für $n \to \infty$ beschränkt bleibt. Da Logarithmen zu verschiedenen Basen proportional zueinander sind, ist auch

$$\log_2 n! = n \log_2 n - \frac{n}{\ln 2} + \frac{\log_2 n}{2} + O(1)\,,$$

und der Informationsgehalt der Nachricht ist

$$\begin{aligned}
&N \log_2 N - (p_1 N) \log_2 (p_1 N) - \cdots - (p_m N) \log_2 (p_m N) \\
&- \frac{N}{\ln 2} + \frac{p_1 N}{\ln 2} + \cdots + \frac{p_m N}{\ln 2} \\
&+ \frac{\log_2 N}{2} - \frac{\log_2 (p_1 N)}{2} - \cdots - \frac{\log_2 (p_m N)}{2} + O(1) \\
&= N \log_2 N (1 - p_1 - \cdots - p_r) - N (p_1 \log_2 p_1 + \cdots + p_m \log_2 p_m) \\
&- \frac{N(1 - p_1 - \cdots - p_m)}{\ln 2} \\
&+ \frac{1-m}{2} \log_2 N + O(1) \\
&= -N \sum_{i=1}^{m} p_i \log_2 p_i + \frac{1-m}{2} \log_2 N + O(1)\,,
\end{aligned}$$

denn

$$1 - \sum_{i=1}^{m} p_i = 0\,.$$

Für $N \to \infty$ ist die Information pro Buchstabe also gleich

$$-\sum_{i=1}^{m} p_i \log_2 p_i\,.$$

Diese Zahl bezeichnen wir als die *Entropie* der Quelle. Sie sollte im allgemeinen kleiner sein als die oben berechneten $\log_2 m$ Bit. Für die 26 Buchstaben der deutschen Sprache mit ihren Häufigkeiten gemäß der Auszählung von *Dr. Katzenbergers Badereise* ist in der Tat

$$-\sum_{i=1}^{26} p_i \log_2 p_i \approx 4{,}04088 < \log_2 26 \approx 4{,}70044\,,$$

und auch allgemein haben wir das folgende

Lemma: Für m Zahlen $p_1, \dots, p_m \in [0,1]$ mit $\sum_{i=1}^m p_i = 1$ gilt stets

$$0 \le -\sum_{i=1}^{m} p_i \log_2 p_i \le \log m\,;$$

dabei steht rechts genau dann ein Gleichheitszeichen, wenn alle p_i gleich $1/m$ sind.

Der *Beweis* erfolgt durch vollständige Induktion nach m.

Der Fall $m = 1$ ist trivial, denn dann muß $p_1 = 1$ sein, und die Behauptung gilt, da sowohl $\log_2 p_1$ also auch $\log_2 m$ verschwinden.

Für $m > 1$ können wir uns sofort auf den Fall $m = 1$ berufen, falls eine der Zahlen p_i gleich eins sein sollt. Interessant ist daher nur der Fall, daß mindestens ein p_i von eins verschieden ist, und dies sei o.B.d.A. das letzte, also p_m.

Wir betrachten die Summe $-\sum_{i=1}^m p_i \log_2 p_i$ zunächst nur unter der Nebenbedingung $p_m = q < 1$. Dann ist

$$\sum_{i=1}^{m-1} p_i = 1 - q\,,$$

mit

$$p_i^* = \frac{p_i}{1-q} \quad \text{ist also} \quad \sum_{i=1}^{m-1} p_i^* = 1\,.$$

Darau können wir die Induktionsannahme anwenden:

$$0 \le -\sum p_i^* \log_2 p_i^* \le \log_2(m-1)$$

mit Gleichheitszeichen rechts genau dann, wenn alle $p_i^* = \frac{1}{1-m}$ sind. Dies müssen wir mit $\sum_{i=1}^m p_i \log_2 p_i$ in Verbindung bringen; setzen wir also die Definition von p_i^* ein:

$$\begin{aligned}
\sum_{i=1}^{m-1} p_i^* \log_2 p_i^* &= \frac{1}{1-q}\sum_{i=1}^{m-1} p_i\big((\log_2 p_i - \log_2(1-q))\big)\\
&= \frac{1}{1-q}\sum_{i=1}^{m-1} p_i \log_2 p_i - \frac{1}{1-q}\sum_{i=1}^{m-1} p_i \log_2(1-q)\\
&= \frac{1}{1-q}\sum_{i=1}^{m-1} p_i \log_2 p_i - \frac{1}{1-q}(1-q)\log_2(1-q)\\
&= \frac{1}{1-q}\sum_{i=1}^{m-1} p_i \log_2 p_i - \log_2(1-q)\,.
\end{aligned}$$

Somit ist die hier interessante Summe

$$-\sum_{i=1}^{n} p_i \log p_i = -(1-q)\sum_{i=1}^{m-1} p_i^* \log_2 p_i^* - (1-q)\log_2(1-q) - q\log_2 q\,.$$

Die negative Summe über die p_i^* wird nach Induktionsannahme genau dann maximal, wenn alle $p_i^* = \frac{1}{m-1}$ sind; unter der Nebenbedinung $p_m = q$ ist der gesamte Ausdruck also höchstens gleich

$$(1-q)\log_2(m-1) - (1-q)\log_2(1-q) - q\log_2 q\,.$$

Das müssen wir nun noch noch in Abhängigkeit von q maximieren; die Ableitung nach q ist

$$\begin{aligned}
&-\log_2(m-1) + \log_2(1-q) + \frac{1-q}{1-q} - \log_2 q - \frac{q}{q}\\
= &-\log_2(m-1) + \log_2(1-q) - \log_2 q = \log_2 \frac{1-q}{(m-1)q}\,,
\end{aligned}$$

und dies wird genau dann gleich null, wenn der Bruch eins ist, d.h. für

$$1 - q = (m-1)q \qquad \text{oder} \qquad q = \frac{1}{m}\,.$$

In diesem Fall sind auch alle $p_i = 1/m$, und damit ist das Lemma bewiesen.

■

3) Mit Vorwissen über die Kontakthäufigkeiten

Wie wir bei der Kryptoanalyse von allgemeinen monoalphabetischen Substitutionen gesehen haben, ergeben die Kontakthäufigkeiten ein deutlich klareres Bild als die bloßen Buchstabenhäufigkeiten; auch dies läßt sich informationstheoretisch nachvollziehen:

Wir bezeichnen die Wahrscheinlichkeit dafür, daß der Buchstabe y auf den Buchstaben x folgt mit $p(x|y)$ und die Wahrscheinlichkeit für das Auftreten von x mit $p(x)$. Die Nachricht $x_1x_2\dots x_N$ hat dann die Wahrscheinlichkeit

$$p(x_1\dots x_N) = p(x_1)p(x_2|x_1)\cdots p(x_N|x_{N-1})\,.$$

Wir betrachten diese Nachricht formal als *einen* Buchstaben $\mathbf{x}$ aus dem Alphabet A^N und wissen dann aus dem vorigen Abschnitt, daß zur Übertragung eines solchen Buchstaben im Mittel

$$H_N = -\sum_{\mathbf{x}\in A^N} p(\mathbf{x})\log_2 p(\mathbf{x})$$

Bit notwendig sind. Dabei ist H_1 gerade die im vorigen Abschnitt berechnete Entropie für das Alphabet A.

Versuchen wir, H_N aus H_{N-1} zu berechnen! Dazu schreiben wir

$$\mathbf{x} = \mathbf{y}x_N \quad \text{mit} \quad \mathbf{y}\in A^{N-1}\,;$$

dann ist

$$p(\mathbf{x}) = p(x_1\dots x_N) = p(\mathbf{y})p(x_N|x_{N-1})$$

und

$$\log_2 p(\mathbf{x}) = \log_2 p(\mathbf{y}) + \log_2 p(x_N|x_{N-1})\,,$$

also

$$p(\mathbf{x})\log_2 p(\mathbf{x}) = p(\mathbf{y})p(x_N|x_{N-1})\big(\log_2 p(\mathbf{y}) + \log_2 p(x_N|x_{N-1})\big)$$

und

$$\begin{aligned} H_N &= -\sum_{\mathbf{x}\in A^N} p(\mathbf{x})\log_2 p(\mathbf{x}) \\ &= -\sum_{\mathbf{x}\in A^N} p(\mathbf{y})\log_2 p(\mathbf{y})\cdot p(x_N|x_{N-1}) - \sum_{\mathbf{x}\in A^N} p(\mathbf{x})\log_2 p(x_N|x_{N-1})\,. \end{aligned}$$

Dabei ist

$$\begin{aligned} &-\sum_{\mathbf{x}\in A^N} p(\mathbf{y})\log_2 p(\mathbf{y})\cdot p(x_N|x_{N-1}) \\ &= -\sum_{\mathbf{y}\in A^N} p(\mathbf{y})\log_2 p(\mathbf{y}) \sum_{x_N\in A} p(x_N|x_{N-1})\,. \end{aligned}$$

Für jedes $x_{N-1}\in A$ und damit auch für jedes $\mathbf{y}\in A^{N-1}$ ist

$$\sum_{x_N\in A} p(x_N|x_{N-1}) = 1\,,$$

denn irgendein Buchstabe muß schließlich auf x_{N-1} folgen. Also ist

$$-\sum_{\mathbf{x}\in A^N} p(\mathbf{y})\log_2 p(\mathbf{y})\cdot p(x_N|x_{N-1}) = H_{N-1}$$

und

$$H_N = H_{N-1} + \Delta_N \quad \text{mit} \quad \Delta_N = -\sum_{\mathbf{x}\in A^N} p(\mathbf{x})\log_2 p(x_N|x_{N-1})\,.$$

Für $N=2$ ist also

$$\Delta_2 = -\sum_{x_1x_2\in A^2} p(x_1x_2)\log_2 p(x_2|x_1)\,.$$

Für $N>2$ ist nach Definition

$$p(x_1\dots x_N) = p(x_1)p(x_2|x_1)\cdots p(x_N|x_{N-1})\,,$$

und

$$\begin{aligned} &\sum_{x_1\in A} p(x_1)p(x_2|x_1)\cdots p(x_N|x_{N-1}) \\ &= \sum_{x_1\in A} p(x_1x_2)p(x_3|x_2)\cdots p(x_N|x_{N-1}) \\ &= p(x_2)p(x_3|x_2)\cdots p(x_N|x_{N-1})\,, \end{aligned}$$

denn

$$\sum_{x_1\in A} p(x_1x_2) = p(x_2)\,.$$

Also ist $\Delta_N = \Delta_{N-1}$, d.h. $\Delta_N = \Delta_2$ für alle $N\ge 2$.

Damit ist gezeigt:

Lemma: $H_N = H_1 + (N-1)H_\infty$, wobei H_1 die Entropie ist und

$$H_\infty = -\sum_{x_1x_2\in A^2} p(x_1x_2)\log_2 p(x_2|x_1)\,.$$

■

Asymptotisch, für $N \to \infty$, ist also der Informationsgehalt pro Buchstabe gleich H_∞.

Auf der Basis von JEAN PAULS Roman *Dr. Katzenbergers Badereise* erhält man

$$H_2 \approx 7{,}43853 \text{ Bit}\,.$$

Da wir

$$H_1 \approx 4{,}04088 \text{ Bit}$$

bereits kennen, liefert das Lemma

$$H_\infty \approx 3{,}39765 \text{ Bit/Buchstabe}\,.$$

Genauso lassen sich H_1, H_2 und H_∞ auch für andere Arten von Daten bestimmen: Betrachten wir etwa Dr. Katzenbergers Badereise als ASCII-Text, also mit Groß- und Kleinschreibung, Leerzeichen und Satzzeichen, so ergeben sich die Werte

$$H_1 \approx 4{,}423089, \quad H_2 \approx 7{,}797414 \quad \text{und} \quad H_\infty \approx 3{,}374325\,,$$

beim gleichen Text im Format von MS WORD ist

$$H_1 \approx 5{,}077718, \quad H_2 \approx 8{,}248848 \quad \text{und} \quad H_\infty \approx 3{,}171130\,,$$

je formatierter der Text ist, desto kleiner wird also H_∞.

Große Werte von H_∞ erhält man für komprimierte Dateien; auszählen einer `.zip`-Datei etwa ergab

$$H_1 \approx approx 7{,}998676, \quad H_2 \approx 15{,}925112 \quad \text{und} \quad H_\infty \approx 7{,}926437\,.$$

Programmdateien führen auf ähnliche Werte wie Texte: Für eine Programmdatei unter Windows ergab sich

$$H_1 \approx 5{,}278473, \quad H_2 \approx 8{,}791534 \quad \text{und} \quad H_\infty \approx 3{,}513061\,,$$

für eine unter UNIX

$$H_1 \approx 6{,}550956, \quad H_2 \approx 11{,}462319 \quad \text{und} \quad H_\infty 4{,}911363\,.$$

Schließlich kann man auch Trigrammhäufigkeiten und so weiter im statistischen Modell berücksichtigen; hierfür habe ich nur die Entropie H_n pro n-Gramm und, als ein n-tel davon, die pro Buchstabe ausgerechnet, ausgezählt auf der Basis von 26 Buchstaben ohne Leerzeichen *usw.:*

n	H_n	H_n/n
1	4,04088	4,04088
2	7,43853	3,71926
3	10,37095	3,45698
4	12,81915	3,20479

Man schätzt, daß bei voller Ausnutzung aller statistischer Strukturen der Sprache schließlich noch etwa 30% der $\log_2 26$ Bit/Buchstabe notwendig sind. Zwar ist eine Sprache mit all ihrer Orthographie und Grammatik viel zu komplex, als daß man eine solche Zahl exakt berechnen könnte, man kann sie aber doch experimentell schätzen, indem man muttersprachlichen Lesern Lückentexte vorlegt und schaut, bis zu welchem Prozentsatz lesbarer Buchstaben sie das Original noch rekonstruieren können.

§2: Kryptoanalyse durch den Bayesschen Gegner

a) Die Grundidee

Gegeben seien ein Alphabet A (über das wir im Augenblick keine weitere Annahmen machen) und ein Kryptosystem

$$\{T_s\colon A \to A \mid s \in S\}\,.$$

Dabei sind die Abbildungen $T_s\colon A \to A$ Verschlüsselungsverfahren, die von einem Schlüssel $s \in S$ abhängen.

Bei der Einschätzung der Sicherheit dieses Systems gehen wir, aus den im vorigen Kapitel genannten Gründen, davon aus, daß der Gegner das Kryptosystem kennt; wir beurteilen die als nur auf der Grundlage des Schlüssels.

Ein Gegner muß zur Entschlüsselung einer Nachricht also „nur“ noch den Schlüssel finden; ein gutes Kryptosystem versucht, ihm dieses so schwer wie möglich zu machen.

Wie eingangs erwähnt, betrachten wir in diesem Paragraphen einen „idealisierten“ Gegner, der über unbegrenzte Rechenmöglichkeiten verfügt, so daß er jeden endlichen Algorithmus, unabhängig von der Zahl der notwendigen Rechenschritte, in kurzer Zeit durchführen kann; diesen Gegner bezeichnet man in der Kryptologie als den BAYESschen Gegner.

Sein Name ist abgeleitet von dem englischen Theologen THOMAS BAYES (1702–1761), der als erster das Entscheidungsprinzip formulierte, wonach unter mehreren möglichen Hypothesen diejenige als wahr anzusehen sei, die vor dem Hintergrund seines vorhandenen Wissens die größte Wahrscheinlichkeit hat; auf die Kryptanalyse angewandt bedeutet dies, daß man sich unter allen möglichen Schlüsseln für den entscheidet, bei dem Klartext entsteht, der gemäß dem jeweiligen statistischen Modell für Klartexte die größte Wahrscheinlichkeit hat.

b) Einführendes Beispiel

Um zu sehen, wie das funktioniert, beobachten wir den BAYESschen Gegner bei der Entschlüsselung einer CAESAR-Chiffre – bei komplizierteren Chiffren macht sich zu schnell bemerkbar, daß er deutlich schneller rechnen kann als wir.

Angenommen, er empfängt das schon im letzten Kapitel betrachtete Kryptogramm QJULIILPPRUJHQJUDXH Q .

Unter der Annahme, daß es sich hier um eine CAESAR-Chiffre handelt, ist es nicht nur für den BAYESschen Gegner, sondern auch für uns leicht möglich, alle 26 möglichen Klartexte sowie deren jeweiligen Wahrscheinlichkeiten zu berechnen. Zu jedem dieser Klartexte berechnen wir dann dessen Wahrscheinlichkeit in der deutschen Sprache.

Um zu sehen, wie viele Chiffretextbuchstaben zur Entschlüsselung notwendig sind, betrachten wir nicht das gesamte Kryptogramm auf einmal, sondern wir betrachten in der i-ten Zeile der folgenden Tabelle

nur die ersten i Buchstaben. Für diese ist jeweils die wahrscheinlichstse Entschlüsselung angegeben, zunächst auf der Basis der Buchstabenwahrscheinlichkeiten, dann auf Basis der Kontaktwahrscheinlichkeiten. Die angegebenen Wahrscheinlichkeiten sind der Übersichtlichkeit halber nicht die absoluten Klartextwahrscheinlichkeiten der jeweiligen Entschlüsselungen, sondern die relativen: Wir dividieren also durch die Summe der Wahrscheinlichkeiten der 26 möglichen Klartexte. Das Ergebnis ist in Prozent angegeben:

E	18,498	E	18,498
RE	25,379	ER	54,251
NAT	25,832	ERK	40,153
NATE	71,640	NATE	77,103
ANGRI	50,664	ANGRI	58,662
NATEV S	58,718	ANGRI F	95,260
NATEV SS	78,445	ANGRI FF	99,281
NATEV SSV	33,482	ANGRI FFI	99,311
ANGRI FFIM	42,358	ANGRI FFIM	99,974
ANGRI FFIMM	31,116	ANGRI FFIMM	99,997
FSLWN KKNRR T	35,311	ANGRI FFIMM O	100,000
ANGRI FFIMM OR	43,708	ANGRI FFIMM OR	100,000
ANGRI FFIMM ORG	33,840	ANGRI FFIMM ORG	100,000
ANGRI FFIMM ORGE	65,818	ANGRI FFIMM ORGE	100,000
ANGRI FFIMM ORGEN	84,217	ANGRI FFIMM ORGEN	100,000
ANGRI FFIMM ORGEN G	73,487	ANGRI FFIMM ORGEN G	100,000
ANGRI FFIMM ORGEN GR	51,676	ANGRI FFIMM ORGEN GR	100,000
NATEV SSVZZ BETRA TEN	62,256	ANGRI FFIMM ORGEN GRA	100,000
NATEV SSVZZ BETRA TENH	66,592	ANGRI FFIM MORGENG RAU	100,000
ANGRI FFIMM ORGEN GRAUE	57,125	ANGRI FFIMM ORGEN GRAUE	100,000
ANGRI FFIMM ORGEN GRAUE N	70,387	ANGRI FFIMM ORGEN GRAUE N	100,000

Mit Buchstabenhäufigkeiten allein schwankt der BAYESsche Gegner also noch ziemlich lange zwischen zwei Schlüsseln, beachtet er auch Kontakthäufigkeiten, hat er schon nach fünf Buchstaben die richtige Entschlüsselnung und ab dem sechsten Buchstaben weiß er das mit einer Wahrscheinlichkeit von über 95%.

Als nächstes Kryptogramm betrachten wir PWDUYFSFQDXJ .

Hier ergeben sich folgende wahrscheinlichste Entschlüsselungen:

E	18,498	E	18,498
EL	20,611	EL	27,369
ELS	36,252	ELS	61,022
ZGNE	20,524	MTAR	27,712
ZGNEI	51,027	MTARV	32,544
NUBSW D	25,059	DKRIM T	46,246
ELSJN UH	42,422	KRYPT AN	63,235
ELSJN UHU	44,039	KRYPT ANA	72,856
DKRIM TGTE	81,297	KRYPT ANAL	53,404
DKRIM TGTER	85,742	DKRIM TGTER	99,705
DKRIM TGTER L	87,118	DKRIM TGTER L	98,089
ZGNEI PCPAN HT	75,437	KRYPT ANALY SE	99,797

Hier gelingt die Entschlüsselung also nur mittels Kontakthäufigkeiten, und auch da nur dank des guten Doktors STRYKIUS, der dafür sorgt, daß das Digramm „RY“ in *Dr. Katzenbergers Badereise* mit positiver Wahrscheinlichkeit auftritt: Bei vielen anderen Referenztexten tritt diese Buchstabenkombination nie auf, so daß die Wahrscheinlichkeit eines jeden Worts, daß sie enthält auf null gesetzt würde.

Man muß freilich beachten, daß der ideale BAYESsche Gegner nicht mit der Auszählung eines einzelnen Texts arbeitet, sondern *die* korrekte Wahrscheinlichkeitsverteilung kennt – wie immer diese auch definiert sein mag.

Es ist natürlich kein Zufall, das der BAYESsche Gegner einbuchstabige Nachrichten immer als „E“ entschlüsselt: Für einbuchstabige Texte gibt es nach beiden Wahrscheinlichkeitsmodellen keine bessere Alternative. Entsprechend einfach werden auch Wörter erkannt, die mit „E“ beginnen und viele häufige Buchstaben und Buchstabenkombinationen enthalten wie beispielsweise im Kryptogramm JSIQNHMPJNY, wo beide Modelle sofort den richtigen Schlüssel finden:

E	18,498	E	18,498
EN	43,447	EN	66,315
END	49,997	END	96,907
ENDL	60,346	ENDL	93,935
ENDLI	89,043	ENDLI	99,306
ENDLI C	89,309	ENDLI C	99,995
ENDLI CH	96,787	ENDLI CH	100,000
ENDLI CHK	92,988	ENDLI CHK	100,000
ENDLI CHKE	98,564	ENDLI CHKE	100,000
ENDLI CHKEI	99,529	ENDLI CHKEI	100,000
ENDLI CHKEI T	99,939	ENDLI CHKEI T	100,000

Auch nicht zu exotische Substantive mit Artikel sind meist problemlos: IJWYNXHM führt zu

E	18,498	E	18,498
DE	22,057	DE	38,733
DER	38,490	DER	87,654
DERT	51,481	DERT	96,107
DERTI	75,471	DERTI	99,393
DERTI S	88,877	DERTI S	99,717
DERTI SC	87,934	DERTI SC	99,999
DERTI SCH	90,031	DERTI SCH	100,000

und PUQEFGQTXQ zu

E	18,498	E	18,498
IN	23,056	CH	40,135
DIE	41,643	DIE	43,465
DIES	64,018	DIES	70,969
DIEST	69,681	DIEST	90,466
DIEST U	53,199	DIEST U	76,378
DIEST UE	83,884	DIEST UE	91,854
DIEST UEH	87,958	DIEST UEH	97,642
DIEST UEHL	96,798	DIEST UEHL	99,523
DIEST UEHLE	99,372	DIEST UEHLE	99,978

Nur der sächliche Artikel ist etwas problematischer: EBTIBVT wird entschlüsselt als

E	18,498	E	18,498
HE	29,769	HE	28,458
EBT	20,051	DAS	47,768

EBTI	34,481	DASH	32,720
HEWLE	45,218	DASHA	75,568
DASHA U	47,297	DASHA U	98,212
DASHA US	57,475	DASHA US	99,609

Hier gibt es also Schwierigkeiten mit den Buchstabenhäufigkeiten allein, da der Text aus eher nicht so häufigen Buchstaben zusammengesetzt ist. Dieses Problem tritt auch bei anderen Texten auf wie etwa HAQQNAA mit

E	18,498	E	18,498
LE	20,611	UN	25,577
LEU	29,391	UND	69,721
LEUU	24,785	UNDD	88,852
LEUUR	44,254	UNDDA	97,663
LEUUR E	68,174	UNDDA N	99,279
LEUUR EE	81,555	UNDDA NN	99,890

und selbst bei bei so zentralen deutschen Wörtern wieYRVGX HYGHE mit

E	18,498	E	18,498
LE	20,611	UN	25,577
HAE	33,485	LEI	39,068
LEIT	54,851	LEIT	71,632
UNRCT	43,649	LEITK	41,025
UNRCT D	51,689	VOSDU E	56,410
UNRCT DU	59,272	LEITK UL	71,093
UNRCT DUC	47,724	LEITK ULT	98,296
UNRCT DUCD	48,308	LEITK ULTU	91,777
UNRCT DUCDA	50,558	LEITK ULTUR	97,113

oder FNHZNTRA mit

E	18,498	E	18,498
EM	15,918	EM	21,504
LTN	19,493	CKE	37,812
AICU	20,640	SAUM	70,182
AICUI	28,589	SAUMA	97,483

AICUI O	29,941	SAUMA G	98,658
SAUMA GE	52,561	SAUMA GE	99,995
SAUMA GEN	78,116	SAUMA GEN	100,000

Zusammenfassend können wir festhalten, daß dem BAYESschen Gegner zumindest mit Kontakthäufigkeiten bereits wenige Buchstaben zur korrekten Entschlüsselung reichen. Beachtet man noch, daß sich weder ein BAYESsche Gegner noch ein realer Kryptanalytiker auf Kontakthäufigkeiten beschränken muß, sondern auch Sprachkenntnisse einsetzt, die sich auf Trigramme, Worte und so weiter beziehen, kommen sie mit noch weniger Buchstaben aus, als obige Beispiele nahelegen. Während des ersten Weltkriegs aktive Militärkryptographen gehen davon aus, daß man nur knapp zwei Buchstaben pro Alphabet (das heißt also pro CAESAR-Chiffre) benötigt, um eine VIGENÈRE-Chiffre zu brechen.

Als Kuriositär am Rande sei noch erwähnt, daß der BAYESsche Gegner auch ohne Chiffretext bereits wahrscheinlichste „Entschlüsselungen“ findet: Falls er nur mit Buchstabenhäufigkeiten arbeitet, ist das natürlich eine Folge von lauter „E“s, schon bei Kontakthäufigkeiten ergeben sich aber interessantetere Ergebnisse, die hier bis zur Nachrichtenlänge acht aufgeführt sind:

n	Text	Wahrscheinlichkeit
1	e	0.184945672750
2	en	0,042051665485
3	ich	0,011662006378
4	ende	0,003537155688
5	eiche	0,000825895113
6	endich	0,000201549003
7	endende	0,000067649431
8	eichende	0,000015795555

c) Der allgemeine Ansatz

Der BAYESsche Gegner braucht als erstes Wissen oder zumindest Annahmen über den Inhalt der Nachrichten: In den obigen Beispielen handelte es sich um deutschen Klartext, verschlüsselt auf der Grundlage von 26 Buchstaben ohne Zwischenräume und Satzzeichen; das ist natürlich nicht mehr der typische Fall für heutige Kryptanalyse: Hier geht es eher um Dokumente von Textverarbeitungs-, Tabellenkalkulations- oder Datenbankprogrammen oder um ausführbare Programme für ein bekanntes Betriebssystem oder ähnliches.

In jedem dieser Fälle verschafft er sich als erstes durch Auszählen eine Wahrscheinlichkeitsverteilung für die möglichen Klar„texte“ der Länge N und ordnet dadurch jedem solchen Klartext W eine

Klartextwahrscheinlichkeit $p_K(W)$

zu; diese kann beispielsweise so wie im letzten Paragraphen über Buchstabenhäufigkeiten oder über Kontakthäufigkeiten definiert sein, aber auch viele andere Ansätze sind möglich.

Außerdem ordnet er jedem Schlüssel s aus dem Schlüsselraum S eine

Schlüsselwahrscheinlichkeit $p_S(s)$

zu. Bei einem gut gemanagten Kryptosystem sollten alle Schlüssel mit gleicher Wahrscheinlichkeit auftreten, so daß

$$p_S(s) = \frac{1}{\#S} \quad \text{für alle } s \in S$$

ist, aber in der Praxis kann der Gegner oft große Vorteile aus der Tatsache ziehen, daß dies nicht der Fall ist. (Wie zufällig sind Ihre Paßwörter?)

Falls nun ein Chiffretext $C \in A^N$ aufgefangen wird, muß der BAYESsche Gegner berechnen, wie wahrscheinlich jeder Schlüssel $s \in S$ vor dem Hintergrund dieses Chiffretexts ist. Dazu müssen bedingte Wahrscheinlichkeiten berechnet werden, also muß man zunächst die Wahrscheinlichkeit des Chiffretexts $C \in A^N$ kennen.

C entsteht aus einem Klartext $W \in A^N$ durch Verschlüsselung mit einem Schlüssel $s \in S$; da der Schlüssel vom Klartext unabhängig sein sollte, ist die Wahrscheinlichkeit für das Zusammentreffen von Klartext W und Schlüssel s gleich

$$p_K(W) \cdot p_S(s)\,.$$

Die Wahrscheinlichkeit, einen bestimmten Chiffretext C zu empfangen, ist also *a priori* gleich

$$p_{Ch}(C) = \sum_{\substack{(W,s) \in A^N \times S \\ T_s(W) = C}} p_K(W) \cdot p_S(s)\,.$$

Die Wahrscheinlichkeit, daß C auftritt und mit Schlüssel $s \in S$ verschlüsselt wurde, ist entsprechend gleich

$$p_{Ch,S}(C,s) = \sum_{\substack{W \in A^N \\ T_s(W) = C}} p_K(W) \cdot p_S(s)\,,$$

wobei die Summe hier im allgemeinen wegen der Injektivität von T_s nur einen Summanden hat. (T_s muß nicht injektiv auf A^N sein; es muß nur injektiv auf der Teilmenge der tatsächlich vorkommenden Nachrichten sein.)

Die eigentlich interessante Wahrscheinlichkeit, die Wahrscheinlichkeit, daß ein gegebener Chiffretext C durch Verschlüsselung mit $s \in S$ entstanden ist, berechnet sich nun als

$$p_{S|Ch}(s|C) = \frac{p_{Ch,S}(C,s)}{p_{Ch}(C)}\,.$$

Der BAYESsche Gegner entscheidet sich bei seinem Entschlüsselungsversuch für einen Schlüssel, der diese bedingte Wahrscheinlichkeit maximal macht. Um einen solchen Schlüssel zu bestimmen, muß er $p_{S|Ch}(s|C)$ möglicherweise für alle Schlüssel $s \in S$ berechnen, jedoch ist dies angesichts seiner unbegrenzten Rechenfähigkeit kein Problem.

Solange es sich nur um eine Nachricht dreht, wird der BAYESsche Gegner nicht in erster Linie am Schlüssel interessiert sein, sondern am wahrscheinlichsten Klartext. Hier ist entsprechend die Wahrscheinlichkeit für das Zusammentreffen von Klartext W und Chiffretext C gleich

$$p_{Ch,K}(C,W) = \sum_{\substack{s \in S \\ T_s(W) = C}} p_K(W) \cdot p_S(s)$$

und die bedingte Wahrscheinlichkeit für Klartext W, nachdem Chiffretext C aufgefangen wurde, ist

$$p_{K|Ch}(W|C) = \frac{p_{Ch,K}(C,W)}{p_{Ch}(C)}\,.$$

Definition: Eine BAYESsche Entscheidungsfunktion ist eine Familie von Abbildungen

$$\delta_N\colon A^N \to A^N$$

mit der Eigenschaft, daß für jeden Chiffretext $C \in A^N$ gilt:

$$p_{K|Ch}(\delta(C)|C) = \max_{W\in A^N} p_{K|Ch}(W|C)\,.$$

Weniger formal ausgedrückt: Für jeden Chiffretext $C \in A^N$ ist $\delta(C)$ ein Klartext mit höchstmöglicher Wahrscheinlichkeit – idealerweise *der* Klartext mit höchstmöglicher Wahrscheinlichkeit, allerdings könnte es auch mehrere Klartexte mit gleicher Wahrscheinlichkeit geben, so daß $\delta(C)$ im allgemeinen durch die obige Definition nicht eindeutig bestimmt ist. Trotzdem ist klar, daß man nichts besseres tun kann, als mit einer solchen BAYESschen Entscheidungsfunktion zu arbeiten; wenn dies zu nichts führt, ist ein Kryptosystem sicher.

d) Perfekte Sicherheit

Am allersichersten ist ein Kryptosystem, wenn der BAYESsche Gegner aus dem aufgefangenen Chiffretext C *überhaupt keine* Information gewinnen kann, die über seine ursprünglich gewählte Wahrscheinlichkeitsfunktion p_K hinausgeht, wenn also die bedingten Wahrscheinlichkeiten, die er in Kenntnis von C errechnet, gleich den ursprünglichen Wahrscheinlichkeiten sind:

Definition: Ein Kryptosystem $\{T_s \mid s \in S\}$ über dem Alphabet A hat *perfekte Sicherheit* für Nachrichten der Länge N, wenn für jeden Chiffretext $C \in A^N$ und jeden Klartext $W \in A^N$ gilt:

$$p_{K|Ch}(W|C) = p_K(W)\,.$$

Wir kennen bereits ein System mit perfekter Sicherheit, den *one time pad* aus Kapitel eins. Leider ist er für große Kommunikationsnetzwerke, etwa im Bankenbereich, nicht praktikabel, da jeder Teilnehmer mit jedem anderen riesige Schlüssel austauschen müßte. Auch ist die Erzeugung *wirklich* zufälliger Buchstaben- oder Zahlenfolgen nicht ganz einfach; die typischen Zufallsgeneratoren der gängigen Betriebssysteme und Compiler sind kryptographisch extrem schwach und können für solche Zwecke nicht verwendet werden.

Ideal wäre deshalb perfekte Sicherheit mit deutlich kleineren Schlüssellängen, aber dies ist nach einem Satz von SHANNON leider nur sehr eingeschränkt möglich:

Satz: Falls ein Kryptosystem $\{T_s \mid s \in S\}$ perfekte Sicherheit für Nachrichten der Länge N hat, ist die Anzahl der Schlüssel $s \in S$ mindestens gleich der Anzahl der möglichen (d.h. mit Wahrscheinlichkeit $p_K(W) > 0$ auftretenden) Klartexte der Länge N.

Beweis: K^+ sei die Menge aller Klartexte aus A^N, die mit positiver Wahrscheinlichkeit auftreten, und C^+ sei die Menge der mit positiver Wahrscheinlichkeit auftretenden Chiffretexte. Für jedes feste $W \in K^+$ und jedes $C \in C^+$ muß es dann einen Schlüsses $s \in S$ geben, so daß $T_s(W) = C$ ist; denn sonst wäre $p_{K|Ch}(W|C) = 0$, aber $p_K(W) > 0$; der Gegner könnte also aus dem Auftreten von C Information gewinnen. Damit muß es mindestens so viele Schlüssel geben, wie es Chiffretexte in C^+ gibt. Für jeden Schlüssel $s \in S$ ist aber T_s eine injektive Abbildung von K^+ nach C^+, d.h. es muß erst recht so viele Schlüssel geben, wie es mögliche Klartexte gibt. ■

e) Die Mehrdeutigkeit eines Schlüssels

In der Praxis ist es nicht unbedingt notwendig, daß der Gegner aus einem aufgefangenen Chiffretext *überhaupt keine* Information ziehen kann; es reicht, wenn er *nicht genügend* Information bekommt.

Als nächstes wollen wir daher abschätzen, wieviel Information der (besonders gefährliche) BAYESsche Gegner aus einem Chiffretext ziehen kann. Besonders wichtig ist der Schutz des Schlüssels s, denn sobald dieser bekannt ist, können bis zum nächsten Schlüsselwechsel alle Nachrichten entschlüsselt werden.

Die Information, die der Gegner zur Rekonstruktion des Schlüssels gewinnen muß, läßt sich nach den Vorarbeiten des letzten Paragraphen leicht quantifizieren: Es ist die Entropie

$$H_S = -\sum_{s\in S} p_S(s)\log_2 p_S(s)\,.$$

Das Lemma über die Maximalität der Entropie bei Gleichverteilung zeigt also noch einmal die eigentlich auch so selbstverständliche Tatsache, daß ein Kryptosystem umso sicherer ist, je mehr die Wahrscheinlichkeitsverteilung der Schlüssel einer Gleichverteilung entspricht.

Der BAYESsche Gegner kennt die Entropie

$$H_{Ch} = -\sum_{C\,\in\, A^N} p_{Ch}(C)\log_2 p_{Ch}(C)$$

der Menge aller Chiffretexte der Länge N; nachdem er eine Nachricht C aufgefangen hat, hat er im Mittel diese Information gewonnen.

Um daraus Informationen über den Schlüssel zu erhalten, berechnet er zunächst die Entropie der Menge aller Paare (C,s):

$$\begin{aligned}
H_{Ch,S} &= -\sum_{(C,s)\,\in\, A^N\times S} p_{Ch,S}(C,s)\log_2 p_{Ch,S}(C,s)\\
&= -\sum_{(C,s)\,\in\, A^N\times S} p_{Ch,S}(C,s)\bigl(\log_2 p_{Ch}(C) + \log_2 p_{S|Ch}(s|C)\bigr)\\
&= -\sum_{(C,s)\,\in\, A^N\times S} p_{Ch,S}(C,s)\log_2 p_{Ch}(C)\\
&\qquad -\sum_{(C,s)\,\in\, A^N\times S} p_{Ch,S}(C,s)\log_2 p_{S|Ch}(s|C)\\
&= -\sum_{C\,\in\, A^N} p_{Ch}(C)\log_2 p_{Ch}(C)\\
&\qquad -\sum_{(C,s)\,\in\, A^N\times S} p_{Ch,S}(C,s)\log_2 p_{S|Ch}(s|C)\\
&= H_{Ch} - \sum_{(C,s)\,\in\, A^N\times S} p_{Ch,S}(C,s)\log_2 p_{S|Ch}(s|C)\,.
\end{aligned}$$

Die letzte Summe in der letzten Zeile bezeichnen wir als *bedingte Entropie* oder *Mehrdeutigkeit*

$$H_{S|Ch} \underset{\text{def}}{=} -\sum_{(C,s)\,\in\, A^N\times S} p_{Ch,S}(C,s)\log_2 p_{S|Ch}(s|C)$$

des Schlüssels bei vorliegendem Chiffretext; diese Information muß sich der Gegner verschaffen, um den Schlüssel zu bestimmen. Sobald $H_{S|Ch} = 0$ ist, kennt er den Schlüssel, denn da in der obigen Summe alle Summanden kleiner oder gleich null sind, kann $H_{S|Ch}$ nur dann gleich null sein, wenn jeder einzelne Summand verschwindet, wenn also für jeden Schlüssel $s \in S$ entweder $p(C,s) = 0$ ist, womit der Schlüssel mit Sicherheit ausgeschlossen ist, oder $\log_2 p(s|C) = 0$, d.h. $p(s|C) = 1$, womit der Schlüssel s mit Sicherheit richtig ist.

Um die Information abzuschätzen, die der Gegner maximal aus einem Chiffretext gewinnen kann, müssen wir also $H_{S|Ch}$ berechnen. Für ein beliebiges Kryptosystem und beliebige Wahrscheinlichkeitsverteilung der Klartexte können wir offensichtlich nichts konkreteres hinschreiben als die definierende Summe. Die für die gesamte Informationstheorie grundlegende Idee von CLAUDE SHANNON war es, spezifischere Aussagen nicht über ein spezielles, sondern über das *durchschnittliche* Kryptosystem zu machen; diese Aussagen sollten dann *im wesentlichen* für die meisten Kryptosysteme gelten.

f) Randomisierung

Um von Durchschnitten reden zu können, müssen wir zunächst einige Annahmen machen über das Verhalten der Klartexte. Wir betrachten wieder Klartexte über einem Alphabet A aus n Buchstaben; für $x \in A$ sei $p(x)$ die Wahrscheinlichkeit, mit der der Buchstabe x auftritt, und für $(x,y) \in A^2$ sei $p(x|y)$ die bedingte Wahrscheinlichkeit dafür, daß er auf ein y folgt.

Wir hatten bislang immer angenommen, daß etwa die Buchstabenhäufigkeiten, die wir durch Auszählen eines hinreichend langen Texts erhalten, in hinreichend guter Näherung mit denen übereinstimmen, die wir in einem gegebenen Klartext vorfinden. Dies müssen wir für die folgenden Überlegungen etwas präziser fassen:

Definition: Für einen gegebenen Klartext $W = w_1 \dots w_N \in A^N$ sei

$$m_x(W) = \#\{i \le N \mid w_i = x\}$$

und

$$m_{xy}(W) = \#\{i \le N-1 \mid w_i = x \text{ und } w_{i+1} = y\}\,.$$

$m_x(W)$ und $m_{xy}(W)$ zählen also, wie oft der Buchstabe x *bzw.* das Buchstabenpaar xy in W vorkommen. Idealerweise sollte für einen hinreichend langen Text W und zwei Buchstaben $x, y \in A$ gelten

$$p(x) \approx \frac{m_x(W)}{N} \quad \text{und} \quad p(x|y) \approx \frac{m_{yx}(W)}{m_y(W)}\,.$$

Exakt kann das natürlich schon aus zahlentheoretischen Gründen nicht gelten, aber wir wollen verlangen, daß die Abweichung für große N mit sehr kleiner Wahrscheinlichkeit größer als eine vorgebbare Schranke δ ist:

Definition: Eine Quelle für Klartexte heißt *ergodisch,* wenn es zu zwei beliebig vorgebbaren reellen Zahlen $\delta, \varepsilon > 0$ stets eine natürliche Zahl N_0 gibt, so daß für alle $N \ge N_0$ und alle $x \in A$ für Texte $W \in A^N$ gilt

$$p\left(\left|\frac{m_x(W)}{N} - p(x)\right| > \delta\right) < \varepsilon\,.$$

Unsere Ansätze zur Kryptoanalyse klassischer Systeme beruhten somit stets darauf, daß wir annahne, daß die deutsche Sprache eine ergodische Quelle für Klartexte ist; der Erfolg bei unseren Entschlüsselungsversuchen spricht dafür, daß diese Annahme nicht garzu falsch sein sollte. In der Tat zeigt man in der Stochastik, daß jede Quelle, bei der es zu jedem Buchstabenpaar (x, y) einen Text gibt, in dem y irgendwo hinter x steht, ergodisch ist. Da wohl niemand bezweifelt, daß es solche Texte zu jedem Buchstabenpaar gibt (für (q, y) etwa **q**uer durch **Zy**pern), kann so die Ergodizität auch bewiesen werden.

Wir wollen im folgenden Aussagen machen über die *durchschnittliche* ergodische Quelle. Es ist klar, daß wir keine Chance haben, alle ergodischen Quellen zu bestimmen und dann einen Mittelwert zu bilden; der

folgende Satz von SHANNON zeigt aber, daß wir durch Zusammenfassen der Buchstaben zu hinreichend langen Wörtern erreichen können, daß die Wahrscheinlichkeitsverteilung sehr einfach ist: Die Wörter zerfallen in zwei Klassen S und O, so daß Wörter aus S (wie *selten*) praktisch nie vorkommen, während die restlichen Wörter (aus O wie *oft*) alle praktisch dieselbe Wahrscheinlichkeit haben. Aus technischen Gründen arbeiten wir nicht mit Wahrscheinlichkeiten, sondern mit der Entropie: Die buchstabenweise Entropie der Quelle ist, wenn wir wie in §1, *3)* von Kontakthäufigkeiten ausgehen, gleich

$$H = -\sum_{x \in A}\sum_{y \in A} p(x)p(y|x)\log_2 p(y|x)\,,$$

und wir fordern, daß sich diese Entropie gleichmäßig auf die $W \in O$ verteilen soll:

Satz: Für eine ergodische Quelle gibt es zu zwei beliebig vorgebbaren reellen Zahlen $\varepsilon, \eta > 0$ stets ein $N_0 \in \mathbb{N}$, so daß A^N für $N \ge N_0$ als Vereinigung zweier Teilmengen O und S geschrieben werden kann mit den Eigenschaften

1.) $p(W \in S) < \varepsilon$

2.) $\left|\frac{-\log_2 p(W)}{N} - H\right| < \eta$ für alle $W \in O$.

Zum *Beweis* wählen wir zunächst willkürlich eine natürliche Zahl N und eine reelle Zahl $\delta > 0$ und setzen dann

$$O \underset{\text{def}}{=} \left\{ W \in A^N \;\middle|\; \begin{array}{l} p(W) > 0 \text{ und für alle } x, y \in A \text{ ist} \\ \left|m_{xy}(W) - Np(x)p(y|x)\right| < N\delta \end{array} \right\}$$

und $S = A^N \setminus O$. Für $W \in O$ können wir dann $m_{xy}(W)$ schreiben als

$$m_{xy}(W) = Np(x)p(y|x) + N\delta_{xy} \quad \text{mit} \quad \left|\delta_{xy}\right| < \delta$$

und erhalten damit für $W = w_1 \dots w_N$

$$\begin{aligned} p(W) &= p(w_1)\prod_{i=2}^{N} p(w_i|w_{i-1}) \\ &= p(w_1)\prod_{x\in A}\prod_{y\in A} p(y|x)^{m_{xy}(W)} \\ &= p(w_1)\prod_{x\in A}\prod_{y\in A} p(y|x)^{Np(x)p(y|x)+N\delta_{xy}} \; . \end{aligned}$$

Logarithmieren führt auf

$$\begin{aligned} -\log_2 p(w) &= -\log_2 p(w_1) - N\sum_{x\in A}\sum_{y\in A} p(x)p(y|x)\log_2 p(y|x) \\ &\qquad - N\sum_{x\in A}\sum_{y\in A} \delta_{xy}\log_2 p(y|x) \\ &= -\log p(w_1) + NH - N\sum_{x\in A}\sum_{y\in A} \delta_{xy}\log_2 p(y|x) \end{aligned}$$

und somit zur Ungleichung

$$\begin{aligned} \left|\frac{-\log_2 p(W)}{N} - H\right| &= \left|\frac{-\log_2 p(W)}{N} - \sum_{x\in A}\sum_{y\in A}\delta_{xy}\log_2 p(y|x)\right| \\ &< \frac{-\log_2 p(W)}{N} + \delta\sum_{x\in A}\sum_{y\in A} -\log_2 p(y|x) \; . \end{aligned}$$

Wählt man N hinreichend groß und δ hinreichend klein, so läßt sich dieser Ausdruck offensichtlich kleiner als jedes vorgegebene $\eta > 0$ machen, so daß die Eigenschaft 2.) erfüllt ist.

Für die Eigenschaft 1.) müssen wir nachrechnen, mit welcher Wahrscheinlichkeit eine Nachricht $W \in A^N$ in S liegt. Sie liegt genau dann in S, wenn sie nicht in O liegt, für mindestens ein Buchstabenpaar (x, y) muß also

$$\left|m_{xy}(W) - Np(x)p(y|x)\right| \geq N\delta$$

sein. Die Wahrscheinlichkeit hierfür ist sicherlich nicht größer als die Summe über alle Paare (x, y) für die entsprechenden Wahrscheinlichkeiten, d.h.

$$p(W \in S) \leq \sum_{x\in A}\sum_{y\in A} p\left(\left|m_{xy}(W) - Np(x)p(y|x)\right| \geq N\delta\right) \; .$$

Wegen der Ergodizität der Quelle können wir für jedes $\widetilde{\varepsilon} > 0$ ein $N_1 \in \mathbb{N}$ finden, so daß für $N \geq N_1$ gilt

$$p\left(\left|\frac{m_x(W)}{N} - p(x)\right| > \frac{\delta}{2}\right) < \widetilde{\varepsilon}$$

oder, was dasselbe ist,

$$p\left(|m_x(W) - Np(x)| > \frac{\delta}{2}N\right) < \widetilde{\varepsilon} \, .$$

Für große N wird, wieder wegen der Ergodizität der Quelle, auch die Häufigkeit $m_x(W)$ groß; falls wir N so groß wählen, daß wir die Wahrscheinlichkeit des Ereignisses $m_x(W) < N_1$ vernachlässigen können, ist also auch

$$p\left(\left|\frac{m_{xy}(W)}{m_x(W)} - p(y|x)\right| > \frac{\delta}{2}\right) < \widetilde{\varepsilon} \, .$$

Die komplementäre Wahrscheinlichkeit dafür, daß

$$|m_x(W) - Np(x)| < \frac{\delta}{2}N$$

bzw.

$$\left|m_{xy}(W) - m_x(W)p(y|x)\right| < \frac{\delta}{2}m_x(W) \leq \frac{\delta}{2}N$$

ist, beträgt jeweils mindestens $1 - \widetilde{\varepsilon}$; die Wahrscheinlichkeit dafür, daß beides eintritt, ist also mindestens

$$(1 - \widetilde{\varepsilon})^2 > 1 - 2\widetilde{\varepsilon} \, .$$

Falls die erste der beiden Ungleichungen erfüllt ist, gilt erst recht

$$|m_x(W)p(y|x) - Np(x)p(y|x)| < \frac{\delta}{2}Np(y|x) \leq \frac{\delta}{2}N$$

und damit nach der Dreiecksungleichung zusammen mit der zweiten Ungleichung

$$\left|m_{xy}(W) - Np(x)p(y|x)\right| < \frac{\delta}{2}N + \frac{\delta}{2}N = N\delta\,.$$

Diese Ungleichung ist somit mindestens mit Wahrscheinlichkeit $1-2\widetilde{\varepsilon}$ erfüllt; die komplementäre Wahrscheinlichkeit ist

$$p\big(\left|m_{xy}(W) - Np(x)p(y|x)\right| > N\delta\big) < 2\widetilde{\varepsilon}\,.$$

Dies können wir in die oben abgeleitete Formel

$$p(W\in S) \le \sum_{x\in A}\sum_{y\in A} p\left(\left|m_{xy}(W) - Np(x)p(y|x)\right| \ge N\delta\right)$$

einsetzen und erhalten, da es n^2 Buchstabenpaare gibt,

$$p(W\in S) < 2n^2\widetilde{\varepsilon}\,.$$

Da n eine Konstante ist, müssen wir nun nur $\widetilde{\varepsilon}$ hinreichend klein wählen und erhalten dann, daß diese Wahrscheinlichkeit höchstens gleich ε ist, wie behauptet.

■

Die Mengen O und S aus dem gerade bewiesenen Satz lassen sich für eine gegebene Quelle natürlich nur schwer konstruieren, denn im allgemeinen muß N doch schon sehr groß gewählt werden, damit dieser Satz gilt. Wir können allerdings trotzdem Aussagen über die Größe von O und S machen: Für kleine Werte von ε und η machen wir keinen großen Fehler, wenn wir davon ausgehen, daß alle Nachrichten aus O mit *exakt* derselben Wahrscheinlichkeit vorkommen und daß Nachrichten aus S *nie* vorkommen. Dann ist für $W\in A^N$ also

$$p(W) = \begin{cases} 1/\#O & \text{falls } W\in O\\ 0 & \text{falls } W\in S\end{cases}$$

und die Entropie der Quelle ist

$$-\sum_{W\in A^N} p(W)\log_2 p(W) = \sum_{W\in O}\frac{1}{\#O}\log_2\#O = \log_2\#O\,.$$

Die Entropie pro Buchstabe des Alphabets A ist damit gleich

$$\frac{\log_2\#O}{N}\,.$$

Diese Entropie pro Buchstabe können wir berechnen; für die Quellen aus §1 haben wir dies dort bereits getan: Geht man nur von den Buchstabenhäufigkeiten aus, ist sie gleich 4,04088 Bit, und bei Berücksichtigung der Kontakthäufigkeiten gleich

$$\frac{4{,}04088 + 3{,}39765(N-1)}{N}\ \text{Bit}\,.$$

Also ist im ersten Fall

$$\#O \approx 2^{4{,}04088N}$$

und im zweiten

$$\#O \approx 2^{4{,}04088+3{,}39765(N-1)}\,.$$

In A^N gibt es insgesamt

$$26^N \approx 2^{4{,}7044N}$$

Nachrichten; das Verhältnis $\#O$ zu $\#A^N$ läßt sich daher schreiben als

$$\frac{\#O}{\#A^N} = 2^{-R_N}\,,$$

wobei im ersten Fall

$$R_N \approx (4{,}7044 - 4{,}04088)N = 0{,}66352N$$

ist und im zweiten

$$R_N \approx 4{,}7044N - 4{,}04088 + 3{,}39765(N-1) = 1{,}30675N - 0{,}64323\,.$$

Damit können wir uns daran machen, die Mehrdeutigkeit eines Schlüssels für diese (und viele andere) Quellen abzuschätzen: Wir ersetzen die jeweilige Quelle durch ein „durchschnittliche“ ergodische Quelle derselben Entropie und rechnen mit dieser in der Hoffnung, daß wir dadurch keinen allzu großen Fehler machen. Außerdem nehmen wir an, daß N hinreichend groß sei, so daß wir den obigen Satz anwenden können.

Nach Definition ist die Mehrdeutigkeit des Schlüssels nach Empfang einer Chiffretext-Nachricht C der Länge N gleich

$$H_{S|Ch} \underset{\text{def}}{=} -\sum_{(C,s)\,\in\, A^N\times S} p_{Ch,S}(C,s)\log_2 p_{S|Ch}(s|C)$$

Wenn wir davon ausgehen, daß jeder Schlüssel mit derselben Wahrscheinlichkeit auftritt, ist für jeden Schlüssel $s \in S$

$$p_S(s) = \frac{1}{\#S} .$$

und für einen Klartext $W \in A^N$ ist

$$p_K(W) = \begin{cases} 1/\#O & \text{falls } W \in O \\ 0 & \text{sonst} \end{cases} ;$$

eine Nachricht $C \in A^N$ kann also nur dann als Chiffretext auftreten, wenn es (mindestens) einen Schlüssel $s \in S$ gibt, so daß der zugehörige Klartext $W = T_s^{-1}(C)$ in O liegt. Wenn wir die Anzahl aller solcher Schlüssel mit $N_S(C)$ bezeichnen, ist also die Wahrscheinlichkeit des Chiffretexts C gleich

$$p_{Ch}(C) = \frac{N_S(C)}{\#S} \cdot \frac{1}{\#O} = \frac{N_S(C)}{\#S \cdot \#O} ;$$

die Wahrscheinlichkeit für Chiffretext C verschlüsselt mit $s \in S$ ist

$$p_{Ch,S}(C, s) = \begin{cases} \frac{1}{\#O \cdot \#S} & \text{falls } T_s^{-1}(C) \in O \\ 0 & \text{sonst} \end{cases} ,$$

und damit ist die bedingte Wahrscheinlichkeit für den Schlüssel s nach Empfang des Chiffretexts C

$$p_{S|Ch}(s|C) = \frac{p_{Ch,S}(C, s)}{p_{Ch}(C)} = \frac{1}{N_S(C)} .$$

Somit ist

$$\begin{aligned} H_{S|Ch} &= -\sum_{(C,s) \in A^N \times S} p_{Ch,S}(C, s) \log_2 p_{S|Ch}(s|C) \\ &= -\sum_{C \in A^N} \sum_{\substack{s \in S \\ T_s^{-1}(C) \in O}} \frac{1}{\#O \cdot \#S} (-\log_2 N_S(C)) \\ &= \sum_{C \in A^N} \frac{N_S(C)}{\#O \cdot \#S} \log_2 N_S(C) . \end{aligned}$$

Diese Summe können wir nicht weiter ausrechnen, da wir die Zahlen $N_S(C)$ nicht kennen. Nun haben wir aber angenommen, daß wir ein

durchschnittliches Kryptosystem haben, d.h. wir interessieren uns für den Mittelwert von $H_{S|Ch}$ über *alle* Kryptosysteme, und darüber können wir Aussagen machen:

Für einen zufällig gewählen Text $C \in A^N$ und einen zufällig gewählten Schlüssel $s \in S$ ist $T_s^{-1}(C) \in O$ mit Wahrscheinlichkeit

$$\frac{\#O}{\#A^N} = \frac{\#O}{n^N} = 2^{-R_N} ,$$

wobei die reelle Zahl R_N wie oben so gewählt wird, daß die rechtsstehende Gleichung gilt. Entsprechend ist die Wahrscheinlichkeit dafür, daß $T_s^{-1}(C)$ nicht in O liegt, gleich

$$1 - 2^{-R_N} .$$

Die Wahrscheinlichkeit dafür, daß $N_S(C)$ gleich einer festen Zahl m ist, können wir interpretieren als die Wahrscheinlichkeit dafür, daß für m Schlüssel s gilt $T_s^{-1}(C) \in O$, während für die restlichen $\#S - m$ Schlüssel gilt $T_s^{-1}(C) \notin O$. Falls die m Schlüssel vorgegeben sind, ist diese Wahrscheinlichkeit gleich

$$\left(2^{-R_N}\right)^m \left(1 - 2^{-R_N}\right)^{\#S-m} ,$$

und da es $\binom{\#S}{m}$ Möglichkeiten gibt, aus $\#S$ Schlüsseln m auszuwählen, ist die Wahrscheinlichkeit dafür, daß $N_S(C) = m$ ist, gleich

$$\binom{\#S}{m} \left(2^{-R_N}\right)^m \left(1 - 2^{-R_N}\right)^{\#S-m} ,$$

die Anzahl der Chiffretexte C mit $N_S(C) = m$ ist also gleich

$$\binom{\#S}{m} \left(2^{-R_N}\right)^m \left(1 - 2^{-R_N}\right)^{\#S-m} \#A^N .$$

Der durchschnittliche Wert von $H_{S|Ch}$ berechnet sich somit zu

$$\begin{aligned}
&\sum_{m=0}^{\#S} \binom{\#S}{m} \left(2^{-R_N}\right)^m \left(1-2^{-R_N}\right)^{\#S-m} \#A^N \frac{m}{\#O\#S} \log_2 m \\
&= \frac{\#A^N}{\#O}\,\frac{1}{\#S} \sum_{m=0}^{\#S} \binom{\#S}{m} \left(2^{-R_N}\right)^m \left(1-2^{-R_N}\right)^{\#S-m} m \log_2 m \\
&= \frac{2^{R_N}}{\#S} \sum_{m=0}^{\#S} \binom{\#S}{m} \left(2^{-R_N}\right)^m \left(1-2^{-R_N}\right)^{\#S-m} m \log_2 m\,.
\end{aligned}$$

Falls wir $\#S = 26$ setzen und von den beiden oben betrachteten Wahrscheinlichkeitsmodellen ausgehen, wenn wir also ein *zufälliges* Kryptosystem betrachten, das dieselben Parameter hat wie das System der 26 Caesar-Substitutionen, läßt sich dies leicht berechnen; das Ergebnis ist für die beiden oben betrachteten Wahrscheinlichkeitsmodelle in den beiden folgenden Abbildungen dargestellt. Wie man sieht, ist die Mehrdeutigkeit des Schlüssels ab etwa zehn bis zwölf *bzw.* sechs bis acht Buchstaben Chiffretext praktisch gleich null, was auch ungefähr unseren Experimenten mit der Caesar-Chiffre entspricht.

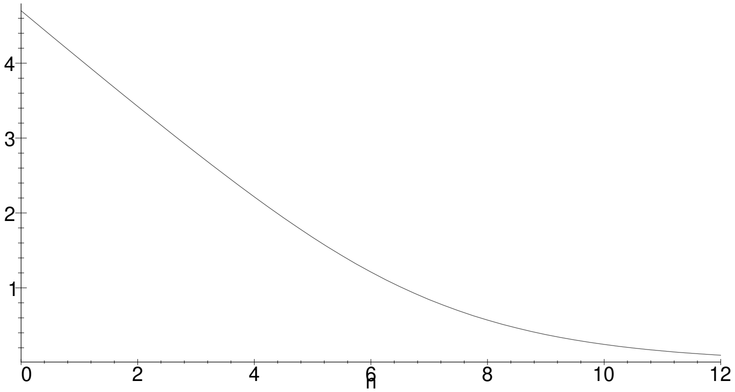


Abb. 1: Mehrdeutigkeit des Schlüssels via Buchstabenhäufigkeiten

Wie man den Kurven ansieht, nimmt der Mehrdeutigkeit des Schlüssels anfänglich etwa linear ab mit n. Wer die Binomialverteilung kennt, sieht auch leicht, warum dies so ist: Falls der Erwartungswert $2^{-R_N}\#S$ von $N_S(C)$ hinreichend groß ist, sind alle einigermaßen wahrscheinliche Werte in seiner näheren Umgebung, man kann daher ohne allzu großen Fehler $N_S(C)$ durch diesen Wert ersetzen und erhält

$$\begin{aligned}
H_{S|Ch} &= \sum_{C\in A^N} \frac{N_S(C)}{\#O\cdot\#S} \log_2 N_S(C) \\
&\approx \#A^N \cdot \frac{2^{-R_N}\#S}{\#O\cdot\#S}\left(-R_N + \log_2 \#S\right) \\
&= \log_2 \#S - R_N\,.
\end{aligned}$$

Anfänglich gewinnt man also aus N Buchstaben Chiffretext etwa R_N Bit des Schlüssels. Falls R_N wie in unseren Beispielen eine lineare Funktion von N ist, nimmt die Mehrdeutigkeit des Schlüssels also anfänglich linear ab.

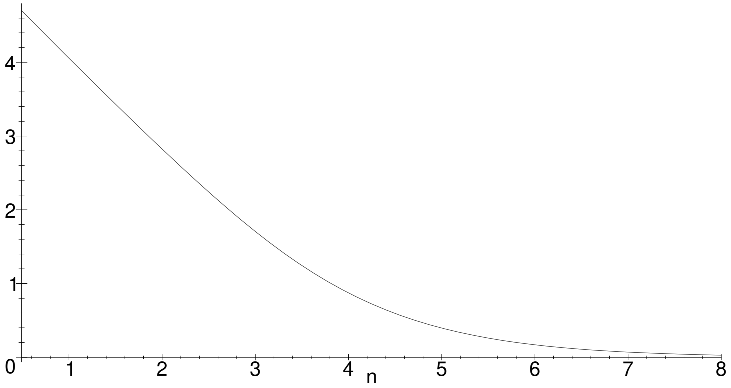


Abb. 2: Mehrdeutigkeit des Schlüssels via Kontakthäufigkeiten

Allgemein sieht man, daß der Gegner die Redundanz R_N der Nachrichtenquelle ausnutzt; je kleiner diese ist, desto weniger Information kann er gewinnen.

§3: Literatur

Eine sehr ausführliche Darstellung der statistischen und informationstheoretischen Ansätze findet man in

ALAN G. KONHEIM: *Cryptography: a primer,* Wiley, New York, 1981

Ein universelles Lehrbuch der Kryptologie mit einem Kapitel über diese Ansätze ist das bereits zitierte Buch

JAN C.A. VAN DER LUBBE: *Basic Methods of cryptography,* Cambridge University Press, 1998

Kapitel 3
Blockchiffren und ihre Kryptanalyse I: DES

§1: Grundlagen

Bei den sogenannten klassischen oder symmetrischen Kryptosystemen gibt es einen zwischen Sender und Empfänger vereinbarten Schlüssel, der sowohl zur Ver- als auch zur Entschlüsselung benutzt wird. Je nach Vorgehensweise unterscheidet man zwei Arten von Chiffren:

- Stromchiffren *und*
- Blockchiffren.

Bei *Stromchiffren* wird ein Text oder sonstiger Datenstrom buchstaben- oder byteweise oder in welchen Einheiten auch immer die Daten ankommen chiffriert, wobei der Schlüssel natürlich von Buchstabe zu Buchstabe wechselt. *Blockchiffren* fassen die Daten zu Blöcken einer festen Länge zusammen und verschlüsseln diese Blöcke als ganzes.

Stromchiffren werden häufig angewandt zur Übermittlung von Satellitendaten, Satellitenfernsehen und ähnlichen kontinuierlichen Datenströmen; Blockchiffren haben ihre Bedeutung im Bankenbereich, sowohl im elektronischen Zahlungsverkehr wie auch zur Identifikation der Benutzer an Geldautomaten, zur Sicherung der Kommunikation zwischen Computern und in ähnlichen Anwendungen.

a) Die Sicherheit einer Blockchiffre

Typischerweise faßt eine Blockchiffre die zu übermittelnden Daten zusammen zu Blöcken von 64 Bit (bei alten Systemen) oder 128 Bit (bei neueren). Jeder Block wird in derselben (von einem Schlüssel abhängigen) Weise durch einen anderen Block derselben Länge ersetzt; ist $\mathcal{B}$

die Menge aller Blöcke der vorgegebenen Länge, so besteht das Kryptosystem also aus bijektiven Abbildungen

$$T_s: B \to B; \qquad s \in S\,.$$

Jedes T_s ist damit ein Element der symmetrischen Gruppe $S_{\#B}$, im Gegensatz zu den monoalphabetischen Buchstabensubstitutionen können wir allerdings nicht mehr alle Permutation zulassen: Falls etwa B aus Blöcken der Länge 64 Bit besteht, hat B immerhin schon 2^{64} Elemente, es gibt also $2^{64}!$ Permutationen. Nach STIRLING ist in erster Näherung

$$\log_2(2^{64}!) \approx 2^{64} \log_2 2^{64} = 64 \cdot 2^{64} = 2^{70}\,,$$

zur Festlegung einer Permutation braucht man also ungefähr 2^{70} Bit oder 2^{67} Byte oder 2^{57} kB oder 2^{47} MB oder 2^{37} GB oder 2^{27} TB oder . . ., jedenfalls viel zu viel.

Aus diesem Grund muß die Abbildungsvorschrift konkreter definiert werden; allerdings sollten die Substitutionen wenn möglich immer noch wie zufallsverteilt in der symmetrischen Gruppe liegen. Beispielsweise sollten sie keine Untergruppe bilden, denn die Kenntnis von Erzeugenden und Relationen dieser Untergruppe könnte einem Gegner Ansatzpunkte zur Entschlüsselung geben.

Ansonsten hängt die Sicherheit einer jeden Blockchiffre stark vom Inhalt der Klartexte und der Schlüssellänge ab: Für deutschen Klartext mit ASCII-Kodierung der Buchstaben etwa enthält ein Block von 64 Bit acht Buchstaben; geht man von einer Redundanz der deutschen Sprache in der Größenordunung von etwa 70% aus, enthält er also eine Information von

$$8 \times 0{,}3 \times \log_2 26 \approx 11{,}28\,\text{Bit}\,;$$

tatsächlich übermittelt werden 64 Bit Chiffretext, so daß in erster Näherung ein Block etwa 52 Bit Schlüsselinformation liefert. Bei einer Schlüssellänge von 64 Bit reichen dem BAYESschen Gegner also im allgemeinen bereits zwei Blocke zur Dekodierung, bei einer Schlüssellänge von 128 Bit braucht er drei.

Sicherer ist ein System, das komprimierten Text übermittelt, da hier die Redundanz bereits deutlich reduziert ist; man muß allerdings beachten,

daß bei vielen gängigen Komprimierungsprogrammen eine ganze Reihe von Anfangsbytes auf feste Werte gesetzt werden, so daß der BAYESsche Gegner mit bekanntem Klartext arbeiten kann.

Für die praktischen Beurteilung einer Blockchiffre hat sich heutzutage folgendes Kriterium durchgesetzt:

- Die Schlüssellänge muß so groß sein, daß systematisches Durchprobieren für die zu erwarteten Gegner unrealistisch ist
- Das Verfahren muß so gewählt werden, daß es keine Attacke gibt, die nennenswert schneller ist als systematisches Durchprobieren

Beide Kriterien sind eher vage formuliert: Das erste hängt davon ab, daß man abschätzen kann, über welche Hardware ein Gegner verfügt, was zumindest bei Gegnern, die für Regierungen arbeiten, kaum möglich ist; beim zweiten müßte man, um wirklich sicher zu sein, die untere Schranken für die Laufzeit eines *beliebigen* Algorithmus finden, der die Chiffre entschlüsseln kann, was weit jenseits der heutigen Möglichkeiten der Komplexitätstheorie liegt.

In der Praxis sollte man sich auf Systeme beschränken, die bereits seit einigen Jahren untersucht sind und dabei keine Schwächen gegenüber kryptanalytischen Attacken zeigten; das Restrisiko, daß jemand nicht dokumentierte bessere Verfahren kennt, muß man versuchen (soweit das möglich ist)durch zusätzliche Sicherheitsfaktoren bei der Schlüssellänge und der Komplexität des Algorithmus zu kompensieren.

b) Beispiel: Hill-Chiffren

Ein möglicher Ansatz für eine Blockchiffre wäre etwa, daß man die Substitution der Blöcke durch lineare Gleichungen angibt: Für $\mathbf{x} = (x_1, \dots, x_N)$ sei die Verschlüsselung $\mathbf{y} = (y_1, \dots, y_N)$ mit

$$y_i = \sum_{j=1}^{N} a_{ij} x_j + b_i, \qquad a_{ij}, b_j \in \mathbb{F}_2\,,$$

wobei N natürlich die Blocklänge bezeichnet und $\mathbb{F}_2 = \{0, 1\}$ den Körper mit zwei Elementen mit Rechenoperationen modulo zwei.

Der Schlüssel hätte dann die Länge $N(N + 1)$, was handhabbar ist; man muß aber darauf achten, daß die Matrix $A = (a_{ij})$ invertierbar ist, da es sonst keine eindeutige Entschlüsselung gibt. Solche Chiffren heißen HILL-Chiffren nach LESTER HILL, der sie allerdings nicht über $\mathbb{F}_2^N$ betrachtete, sondern über $(\mathbb{Z}/26)^N$, wo sie eine Verallgemeinerung der CAESAR-Chiffren darstellen.

Unsicher sind sie sowohl über $\mathbb{F}_2$ als auch über $\mathbb{Z}/26$, sofern der Gegner über hinreichend viel korrespondierenden Klartext und Chiffretext verfügt und den GAUSS-Algorithmus zur Lösung linearer Gleichungssysteme kennt: Mit $N + 1$ linear unabhängigen Klartextblöcken kann er über einem Körper die Spalten von A sowie auch b berechnen; bei einem Ring wie $\mathbb{Z}/26$ kann es noch Probleme mit gemeinsamen Teilern geben, die aber den Aufwand auch nicht wesentlich größer machen. Auch ohne Kenntnis von Klartext (die man in der Praxis nie ausschließen kann) gibt es Verfahren zur Kryptoanalyse.

c) Diffusion und Konfusion

Nach CLAUDE SHANNON sollte ein Kryptosystem auf zwei Techniken beruhen: **Diffusion** und **Konfusion**.

Diffusion soll die Redundanz des Klartexts möglichst weiträumig über den Chiffretext verteilen. Dies könnte beispielsweise durch eine Permutation der Klartextbuchstaben erreicht werden (was die deutsche Armee im ersten Weltkrieg als hauptsächliches Kryptoverfahren benutzte); allerdings ist diese Permutation allein kryptographisch ziemlich unsicher: Da man beispielsweise weiß, daß auf ein „C“ in deutschen Klartext praktisch immer ein „H“ folgt oder auf ein „Q“ ein „U“, kann man bei hinreichend viel Chiffretext solche Permutationen leicht rückgängig machen. Bei guten Kryptoverfahren werden sie daher nur angewandt als eine von mehreren Komponenten. Diffusion wird auch erreicht, wenn man dafür sorgt, daß jedes Klartextbit möglichst viele Bit des Chiffretexts beeinflußt.

Konfusion soll dafür sorgen, daß der Zusammenhang zwischen Schlüssel und Chiffretext möglichst undurchsichtig ist; der Kryptanalytiker soll aus dem Chiffretext nur wenig Information über die Schlüsselverteilung

gewinnen können. Der *one time pad* ist ein Beispiel dafür, wie Konfusion zu perfekter Sicherheit führen kann; die anderen Beispiele aus dem ersten Kapitel zeigen, daß reine Substitution bei kürzeren Schlüssellängen nicht sonderlich sicher ist.

Moderne Kryptosysteme beruhen darauf, daß Diffusion und Konfusion *in Kombination* zu deutlich größerer Sicherheit führen als jede der beiden Strategien für sich allein.

§ 2: Feistelnetzwerke und der Aufbau des DES

a) Feistelnetzwerke

Bis vor kurzem waren fast alle gebräuchlichen Blockchiffren sogenannte FEISTEL-Netzwerke, die in *Runden* arbeiten: In jeder Runde wird nur ein Teil des Blocks und ein Teil des Schlüssels verwendet, aber nachdem alle der (typischerweise sechzehn) Runden abgearbeitet sind, ist sichergestellt, daß die Ausgabe von allen Schlüsselbits und allen Klartextbits abhängt.

HORST FEISTEL entwickelte ab etwa 1960 bei IBM die ersten Blockchiffren, insbesondere auch das Lucifer-System, auf dessen Grundlage DES entwickelt wurde.

Sei $2N$ die Blocklänge des zu betrachtenden Systems. Dann gibt es eine Funktion

$$f\colon \mathbb{F}_2^k \times \mathbb{F}_2^N \to \mathbb{F}_2^N ,$$

die sogenannte FEISTEL-Funktion, die aus k Schlüsselbits und N Nachrichtenbits wieder N Bits produziert. Diese wird folgendermaßen angewandt: Vor Beginn der i-ten Runde wird die Nachricht aufgeteilt in eine linke Hälfte L und eine rechte Hälfte R. Die i-te Runde ersetzt den Block (L, R) durch

$$\bigl(f(s_i, R) \oplus L, R\bigr) ,$$

wobei $\oplus$ die Addition im Vektorraum $\mathbb{F}_2^N$ bezeichnet und s_i den Schlüssel der i-ten Runde. Nach jeder außer der letzten Runde werden anschließend die linke und rechte Hälfte miteinander vertauscht – ansonsten würde ja die rechte Hälfte den gesamten Algorithmus unverändert durchlaufen.

Die FEISTEL-Funktion dient somit zur *Konfusion;* die Auswahl der Schlüsselbits für die jeweiligen Runden sowie auch die Vertauschung von linker und rechter Seite in jeder Runde zur Diffusion.

b) Aufbau des DES

Als Beispiel betrachten wir den 1977 in USA als Standard eingeführten DES (Data Encryption Standard), der unter anderem auch unserem EC-Kartensystem zugrunde liegt. Er arbeitet mit einem Schlüssel der Länge 56 Bit (der mit acht Prüfbits auf 64 Bit verlängert wird, indem man an je sieben Schlüsselbits ein Paritätsbit anhängt) auf Blöcken der Länge 64.

Zunächst werden die Datenwörter

$$(x_1, x_2, \ldots, x_{64})$$

einer Anfangspermutation unterzogen, d.h. das Wort wird ersetzt durch

$$(x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(64)})\,,$$

wobei die Folge der Zahlen $\pi(1), \ldots, \pi(64)$ der folgenden Tabelle entnommen wird:

58	50	42	34	26	18	10	2	60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6	64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1	59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5	63	55	47	39	31	23	15	7

Das Wort geht also auf

$$(x_{58}, x_{50}, x_{42}, \ldots, x_{23}, x_{15}, x_7)\,.$$

Danach beginnen die sechzehn Runden.

Die Anfangspermutation hat keine kryptographische Funktion: Da sie nicht vom Schlüssel abhängt und allgemein bekannt ist, kann sie jeder Kryptanalytiker leicht rückgängig machen. Ihr Sinn bestand anscheinend in erster Linie darin, Software-Attacken zu erschweren, denn Permutation sind aufwendig zu programmieren. (Bei Hardware-Implementierungen sind Permutationen natürlich sehr einfach und schnell durch Leitungskreuzungen zu realisieren.)

Zur Definition der FEISTEL-Funktion f dienen acht sogenannte S-Boxen (S = Substitution), die als Wertetabellen einem Eingabewort aus sechs Bit einen vier Bit langen Funktionswert zuordnen; sie beschreiben also Abbildungen von $\mathbb{F}_2^6$ nach $\mathbb{F}_2^4$.

Diese Wertetabellen sind folgendermaßen angeordnet: Das Eingabewort mit seinen sechs Bit wird geschrieben als (a, m, e), wobei a das Anfangsbit, e das Endbit und m die aus vier Bit bestehende Mitte ist. Diese wird als Zahl zwischen null und fünfzehn aufgefaßt, genau wie auch die Ausgabe der S-Box. Die S-Box wird angegeben durch vier Zeilen, die mit den verschiedenen Möglichkeiten für das Paar (a, e) indiziert sind, und die für die sechszehn Werte von m die Ausgabewerte enthalten:

Box 1

$a\ e$	m = 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0 0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0 1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
1 0	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
1 1	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Box 2

$a\ e$	m = 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0 0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
0 1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
1 0	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
1 1	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

Box 3

$a\ e$	m = 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0 0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
0 1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
1 0	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1 1	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

Box 4

a e	m = 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0 0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
0 1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
1 0	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
1 1	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

Box 5

a e	m = 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0 0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
0 1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
1 0	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
1 1	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

Box 6

a e	m = 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0 0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
0 1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
1 0	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
1 1	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

Box 7

a e	m = 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0 0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
0 1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1 0	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
1 1	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

Box 8

a e	m = 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0 0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
0 1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
1 0	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
1 1	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Diese S-Boxen werden zusammengesetzt zu einer Funktion

$$F\colon \mathbb{F}_2^{48} \to \mathbb{F}_2^{32}$$

in der offensichtlichen Weise: Ein Vektor der Länge 48 wird aufgeteilt in acht Vektoren der Länge sechs; auf den ersten davon wird die erste S-Box angewandt, auf den zweiten die zweite *usw;* dabei entstehen acht Vektoren der Länge vier, die zum Ergebnisvektor der Länge 32 zusammengesetzt werden.

Nach diesem Konfusionsschritt folgt noch ein Diffusionsschritt: Die Komponenten des Vektors werden untereinander permutiert mittels einer Permutation aus S_{32}, die durch folgende Wertetabelle geben ist:

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

Die Funktion F wird folgendermaßen eingesetzt: Zunächst wird die rechte Hälfte R der Eingabe der jeweiligen Runde auf 48 Bit vergrößert, indem man einen Vektor $(x_1, \dots, x_{32})$ ersetzt durch

$$(x_{\tau(1)}, \dots, x_{\tau(48)}) ,$$

wobei die Werte von τ der folgenden Tabelle entnommen werden:

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

Das Ergebnis dieser Aufblähung wird zum Schlüssel der i-ten Runde addiert (in $\mathbb{F}_2^{48}$, d.h. die Addition ist ein logisches XOR).

Die kryptographische Funktion dieser Aufblähung ist natürlich wieder eine Diffusion; sie sorgt dafür, daß ein Eingabebit möglichst viele Ausgabebits beeinflußt.

Damit kommen wir zur Verwendung des Schlüssels im Algorithmus. Der Schlüssel hat, wie bereits erwähnt, 56 Bit, wird aber mit 64 Bit gespeichert, wobei jedes achte Bit ein Paritätsbit ist, d.h. die Summe (in $\mathbb{F}_2$) der sieben davorstehenden Bits. Wir numerieren die Schlüsselbits daher von eins bis 64, verwenden aber nur die nicht durch acht teilbaren Indizes.

Aus dem Schlüssel werden zunächst zwei Schlüssel der Länge 28 extrahiert, bestehend aus den folgenden Komponenten:

57	49	41	33	25	17	9	1	58	50	42	34	26	18
10	2	59	51	43	35	27	19	11	3	60	52	44	36

und

63	55	47	39	31	23	15	7	62	54	46	38	30	22
14	6	61	53	45	37	29	21	13	5	28	20	12	4

Die so erhaltenen Teilschlüssel werden vor jeder Runde noch zirkulär nach links verschoben, und zwar vor der i-ten Runde um nochmals a_i Bit gegenüber der vorherigen Runde, wobei a_1 bis a_{16} die Zahlenfolge

1 1 2 2 2 2 2 2 1 2 2 2 2 2 2 1

ist. Man beachte, daß die Summe dieser Zahlen gleich 28 ist, jeder der Halbschlüssel wird also in den sechzehn Runden einmal komplett zyklisch verschoben.

Nachdem die beiden Teilschlüssel so präpariert sind und aneinandergehängt einen 56 Bit-Schlüssel $(s_1, \dots, s_{56})$ bilden, wird daraus ein 48 Bit-Schlüssel für die i-te Runde gewählt, bestehend aus den folgenden Komponenten:

14	17	11	24	1	5	3	28	15	6	21	10
23	19	12	4	26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40	51	45	33	48
44	49	39	56	34	53	46	42	50	36	29	32

Die Rundenschlüssel werden also in einem reinen Diffusionsverfahren aus dem Gesamtschlüssel berechnet.

Die FEISTEL-Funktion f berechnet die Summe aus den 48 aufgeblähten Nachrichtenbits und den 48 Schlüsselbits der i-ten Runde und setzt diesen Vektor aus F_2^{48} in F ein; der Funktionswert ist der Wert der FEISTEL-Funktion.

Dieses Spiel wird insgesamt sechzehnmal gespielt, wobei nach der letzten Runde keine Vertauschung von links und rechts mehr stattfindet.

Danach wird nur noch die Anfangspermutation rückgängig gemacht; die inverse Permutation hat die Wertetabelle

40	8	48	16	56	24	64	32	39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30	37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28	35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26	33	1	41	9	49	17	57	25

Man beachte, daß DES (wie jedes (balancierte) FEISTEL-Netzwerk) in jeder Runde nur einen halben Block verändert; die andere Hälfte bleibt erhalten. Schreiben wir den Nachrichtenblock nach Anwendung der Anfangspermutation also in der Form (m_0, m_1), so wird in der i-ten Runde (m_{i-1}, m_i) zu (m_i, m_{i+1}) mit

$$m_{i+1} = m_{i-1} \oplus f(s_i, m_i)\,,$$

wobei wieder s_i den Schlüssel der i-ten Runde bezeichnet. Das Ergebnis der sechzehn Runden, abgesehen von der Endpermutation, ist dann allerdings nicht (m_{16}, m_{17}), sondern (m_{17}, m_{16}), da nach der letzten Runde die Hälften nicht mehr miteinander vertauscht werden.

Der Grund dafür liegt in der Entschlüsselung: Wegen

$$m_{i+1} = m_{i-1} \oplus f(s_i, m_i) \Leftrightarrow m_{i-1} = m_{i+1} \oplus f(s_i, m_i)$$

läßt sich die Verschlüsselung bei Kenntnis des Schlüssels leicht rückgängig machen, sogar mit derselben Hardware. Da aber vor dem ersten Verschlüsselungsschritt die Hälften nicht vertauscht werden, sollten sie es dann auch nach dem letzten nicht mehr werden, denn die Entschlüsselung läuft ja rückwärts durch die Runden.

§3: Designkriterien und Kryptoanalyse des DES

Die Beschreibung des DES im vorigen Paragraphen läßt den Leser zunächst wohl mit ziemlicher Verwirrung zurück, und es erscheint schwierig, irgendeine Aussage über die kryptographische Sicherheit des Verfahrens zu machen. In der Tat wurde diese von Anfang an sehr kontrovers diskutiert.

a) Geschichtliche Entwicklung

Als das damalige *National Bureau of Standards* der USA (heute National Institute of Standards and Technology, NIST) im Januar 1977 den DES als Standard veröffentlichte (mit einer auf zehn Jahre veranschlagten Laufzeit) enthielt das Dokument im wesentlichen nur die hier reproduzierten Angaben; sowohl IBM als auch die National Security Agency (NSA) lehnten es ab, die Kriterien zu benennen, nach denen die S-Boxen und die Permutationen konzipiert worden waren.

Dies führte schnell auf den Verdacht, daß DES möglicherweise eine nur IBM und NSA bekannte „Falltür“ enthält, mit deren Hilfe eine Entschlüsselung ohne Schlüssel mit vertretbarem Aufwand durchgeführt werden kann. Außerdem gab es bereits damals Kritik an der mit 56 Bit sehr kurzen Schlüssellänge: Das Vorgängersystem LUCIFER hatte eine Schlüssellänge von 128 Bit, im militärischen Bereich waren Systeme mit mehr als zehn mal so langen Schlüsseln nichts Ungewöhnliches.

M.E. HELLMAN: A cryptanalytic time-memory tradeoff, *IEEE Trans. Inf. Theory* **26** (1980), 401–406

schlug ein Verfahren vor, mit dem durch eine Kombination von Vorberechnungen und Probieren die Komplexität der Schlüsselsuche von 2^{56} mit großer Erfolgswahrscheinlichkeit auf ungefähr die Kubikwurzel dieser Zahl reduziert werden konnte; als Baupreis seiner Maschine schätzte er zehn Millionen Dollar, als Zeitrahmen für die Vorberechnungen ungefähr ein Jahr. Da die NSA erheblich größere Geldmittel als nur zehn Millionen Dollar einsetzen kann, bestärkte dies den Verdacht, daß sie DES selbst dann knacken kann, wenn der Algorithmus keine Falltür enthalten sollte.

Im Laufe der Jahre wurden einige der Designkriterien durch *reverse engineering* gefunden; einige dann auch freiwillig veröffentlicht. Erst 1994 veröffentlichte einer der ursprünglichen Entwickler bei IBM die, wie er sagt, vollständige Liste der kryptographisch relevanten Kriterien in

D. COPPERSMITH: The Data Encryption Standard (DES) and its strength against attacks, *IBM J. Res. Develop.* **38** (1994), S. 243–250

– nachdem die kryptoanalytische Technik, gegen die diese Kriterien schützen sollten, auch in der offenen Literatur erschienen war. Dabei zeigte sich, daß DES mit seinen nur 56 Bit zumindest gegen diese Technik eine eher größere Sicherheit bietet als Lucifer mit seinen 128 Bit und daß die Sicherheit von DES nicht unbedingt erhöht würde, indem man für jede der sechzehn Runden einen neuen 48 Bit-Schlüssel verwendet, so daß man insgesamt eine Schlüssellänge von $16 \times 48 = 762$ Bit hätte. NSA hielt diese Technik damals für so wichtig für den Angriff auf gegnerische Systeme, daß die speziell dagegen eingesetzten Designkriterien „aus Gründen der nationalen Sicherheit“ geheimgehalten wurden.

Die Technik, um die es hier geht, war bei IBM um 1974 unter dem Namen *T attack* bekannt; in der offenen Literatur erschienen erste Ansätze dazu ab etwa 1988, vollständige Beschreibungen erschienen ab 1990 unter dem Namen *differentielle Kryptoanalyse.* Bevor wir sie genauer betrachten, wollen wir uns zunächst die inzwischen bekannten Designkriterien des DES ansehen.

b) Designkriterien

D. COPPERSMITH nennt in der oben zitierten Arbeit folgende Designkriterien für die S-Boxen (und sagt, daß dies *alle* kryptographisch relevanten gewesen seien; der Rest habe nur mit Implementierungsfragen zusammengehängt):

(S1) Jede S-Box hat sechs Eingabe- und vier Ausgabebits.

(S2) Kein Ausgabebit einer S-Box sollte zu nahe bei einer linearen Funktion der Eingabebits liegen.

(S3) Bei festgehaltenem linken und rechtem Bit der Eingabe sollte jeder der sechzehn möglichen Ausgabewerte genau einmal vorkommen.

(S4) Wenn sich zwei Eingaben einer S-Box um genau ein Bit unterscheiden, müssen sich die Ausgaben um mindestens zwei Bit unterscheiden.

(S5) Wenn sich zwei Eingaben einer S-Box genau in den beiden mittleren Bits unterscheiden, müssen sich die Ausgaben um mindestens zwei Bit unterscheiden.

(S6) Wenn sich zwei Eingaben einer S-Box in ihren beiden Anfangsbits, nicht aber in ihren beiden Endbits unterscheiden, müssen die Ausgaben verschieden sein.

(S7) Für jede von null verschiedene Differenz Δ zwischen zwei Eingaben dürfen höchstens acht der 32 Paare mit Differenz Δ auf dieselbe Differenz zwischen den Ausgaben führen.

(S8) Ähnlich zu *(S7),* aber mit stärkeren Eigenschaften für den Fall gleicher Ausgaben, wenn in einer Runde drei S-Boxen „aktiv“ sind. *(s.u.)*

Für die Permutation aus S_{32}, die in jeder FEISTEL-Funktion als Abschluß ausgeführt wird, sollten folgende Bedingungen erfüllt sein:

(P1) Die vier Ausgabebits einer S-Box werden so verteilt, daß in der nächsten Runde zwei von ihnen mittlere Bits der Eingabe einer S-Box sind und die beiden anderen nicht (d.h. die kommen an Position 1, 2, 5 oder 6).

(P2) Die vier Ausgabebits einer S-Box sind in der nächsten Runde Eingaben zu sechs verschiedenen S-Boxen; keine zwei von ihnen sind Eingabe derselben S-Box.

(P3) Für zwei (nicht notwendigerweise verschiedene) S-Boxen j, k gilt: Wenn ein Ausgabebit von j als eines der beiden mittleren Bits an k weitergegeben wird, kann kein Ausgabebit von k als mittleres Bit an j weitergegeben werden. Insbesondere darf also kein Ausgabebit von j an j selbst als mittleres Bit weitergegeben werden.

Der Sinn einiger dieser Kriterien ist unmittelbar einsichtig: *(S1)* etwa kommt daher, daß mit der Technologie von 1974 größere S-Boxen dazu geführt hätten, daß man den Algorithmus nicht auf einem Chip untergebracht hätte.

(S2) ist selbstverständlich: Da die S-Boxen der einzige nichtlineare Bestandteil des Algorithmus sind, müssen sie nichtlinear sein; ansonsten hätten wir eine (leicht zu knackende) HILL-Chiffre. Wenn einzelne Bits lineare Funktionen der Eingabebits wären, hätten wir möglicherweise für einzelne Ausgabebits des Algorithmus lineare Zusammenhänge mit den Eingabebits, was dazu führen würde, daß man zumindest einen Teil der Chiffre als HILL-Chiffre betrachten kann und damit die Komplexität des Algorithmus reduziert. Ähnlich verhält es sich, wenn Funktionen nicht exakt, aber doch ungefähr linear sind – mehr dazu gleich bei der *linearen Kryptanalyse.*

Die restlichen Kriterien dienen in erster Linie zur Förderung der Diffusion: Für zwei verschiedene Eingaben W, W' in Runde i sagen wir, eine S-Box sei *aktiv,* wenn sie für W und W' verschiedene Ausgaben liefert. Es muß nicht in jeder Runde aktive S-Boxen geben, aber die obigen Kriterien sollen dafür sorgen, daß im Durchschnitt über alle Runden möglichst viele S-Boxen pro Runde aktiv sind; wie man zeigen kann, sind es im Durchschnitt mindestens 1,6.

(Bei *(S7)* sind die Zahlen, so wie sie genannt wurden, offensichtlich um den Faktor zwei zu klein: Es gibt 64 Paare mit vorgegebener Differenz Δ, und für $\Delta \neq 0$ dürfen dann wohl höchstens 16 davon auf denselben Ausgabewert führen.)

Bevor wir solche Fragen vertiefen können, müssen wir uns zunächst mit der kryptoanalytischen Attacke beschäftigen, vor der dies schützen soll:

c) Differentielle Kryptoanalyse

Ihre Grundidee besteht darin, daß man nicht von einzelnen Klartextblöcken ausgeht, sondern von Paaren (W, W') aus zwei Klartextblöcken. Diese werden aufgefaßt als Elemente von $\mathbb{F}_2^{64}$; da über dem Körper mit zwei Elementen Addition gleich Subtraktion ist, bezeichnen wir die Differenz zwischen den beiden Nachrichten als $W \oplus W'$. Praktisch handelt es sich hier einfach um das bitweise XOR zwischen den beiden Blöcken.

DES unterzieht die beiden Worte zunächst der Anfangspermutation; da XOR eine bitweise Operation ist, wird dabei auch die Differenz $W \oplus W'$ dieser Permutation unterzogen. Danach werden die rechten Hälften der entstandenen Nachrichten betrachtet; ihre Differenz ist natürlich einfach die rechte Hälfte der permutierten Differenz. Die entstandenen 32 Bit-Worte werden durch Bitauswahl auf 48 Bit Worte V, V' aufgebläht; auch diese Aufblähung ist kompatibel mit der Differenzbildung.

Als nächstes kommt der Schlüssel ins Spiel; sowohl V als auch V' werden zum 48-Bit Schlüssel s_1 der ersten Runde addiert; dann gehen die Ergebnisse $V \oplus s_1$ und $V' \oplus s_1$ in acht 6 Bit Stücke aufgespalten in die acht S-Boxen. Die Differenz zwischen den beiden Eingaben ist

$$(V \oplus s_1) \oplus (V' \oplus s_1) = (V \oplus V') \oplus (s_1 \oplus s_1) = V \oplus V' ,$$

d.h. der Schlüssel ist herausgefallen.

Nun kommen die S-Boxen ins Spiel. Falls diese linear wären, wäre die Differenz ihre Ausgabe für zwei gegebene Eingabewerte nur von der Differenz der Eingabewerte abhängig, aber da es gerade der Zweck der S-Boxen ist, die Verschlüsselung nichtlinear zu machen, können wir natürlich nicht erwarten, daß wir hier auch nur bei einer einzigen S-Box eine lineare Funktion finden: Schon die ersten experimentellen Untersuchungen von DES befaßten sich mit etwaigen linearen Zusammenhängen zwischen einzelnen Ausgabebits sowohl einer S-Box wie auch des gesamten DES und der jeweiligen Eingabe, und keine konnte eine lineare Funktion finden.

Nach der Anwendung der S-Boxen können wir also nicht mehr sagen, was die Differenz der Ausgabewerte ist, obwohl wir die Differenz der Eingabewerte auch unabhängig vom Schlüssel kennen.

Trotzdem zeigt sich, daß wir zumindest gewisse Informationen über die Differenz haben: Bei sechs Eingabebits und vier Ausgabebits pro S-Box müssen von den $2^6 = 64$ Eingabepaaren (E, E') mit einer gegebenen Differenz ΔE im Durchschnitt jeweils vier auf jede der sechzehn möglichen Differenzen ΔA der Ausgabewerte A, A' führen. Im Einzelnen gibt es allerdings beträchtliche Schwankungen:

Für $\Delta E = 0$ ist natürlich auch $\Delta A = 0$, denn dasselbe Wort kann nicht auf zwei verschiedene Weisen verschlüsselt werden. Aber auch für andere Werte von ΔE gibt es keine Gleichverteilung der Ausgabewerte:

Für $\Delta E = 100100$ etwa ergibt sich für die (dezimal geschriebenen) Differenzen ΔA folgende Verteilung:

ΔA	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Fälle	12	0	0	2	2	2	2	0	14	14	2	0	2	6	2	4

Wir haben also hier, wie auch bei den anderen Differenzen und anderen S-Boxen eine ziemlich inhomogene Verteilung. (Die Fallanzahlen für jede S-Box und jede Ausgabedifferenz sind aufgelistet im Anhang von

E. BIHAM, A. SHAMIR: Differential Cryptanalysis of the Data Encryption Standard, *Springer,* 1993;

in diesem Buch ist die differentielle Kryptoanalyse des DES einschließlich der Verallgemeinerung auf andere Systeme vollständig beschrieben.)

Auf die Ausgabe der acht S-Boxen wird noch eine Permutation angewandt; die ist wieder mit Differenzbildung kompatibel. Falls wir also die Differenzen der Ausgaben der S-Boxen kennen, bereitet diese Permutation keine Schwierigkeiten und wir kennen die Eingabedifferenzen für die zweite Runde, mit der wir genauso weiter verfahren können.

Tatsächlich kennen wir die Ausgaben der acht S-Boxen der ersten Runde natürlich nicht; wir können nur für jede einzelne S-Box eine Wahrscheinlichkeitsverteilung der Ausgabedifferenzen angeben. Für einzelne Bits oder Bitgruppen kommen wir dabei durchaus auf recht ansehnliche Wahrscheinlichkeiten: Im obigen Beispiel für die Eingabedifferenz 100100 etwa ist mit jeweils einer Wahrscheinlichkeit von 14 : 64, also in mehr als 20% aller Fälle, die Ausgabedifferenz gleich acht oder neun, binär geschrieben also 0100 oder 0101. Die Wahrscheinlichkeit dafür, daß die ersten drei Bits der Ausgabedifferenz gleich 010 sind, ist also 28 : 64 = 7 : 16, und wenn wir uns auf das erste und dritte Bit beschränken, kommen wir auf eine Wahrscheinlichkeit von 40 : 64 = 5 : 8 dafür, daß die Ausgabedifferenz die Form $0x0y$ hat. Das dritte Bit schließlich ist in 52 der 64 möglichen Fällen gleich null, so daß wir zumindest dieses eine Bit mit der recht hohen Wahrscheinlichkeit von 13 : 16 kennen.

Der Sinn eines FEISTEL-Netzwerks ist es, uns solche Informationen, die wir für eine Runde haben, in den weiteren Runden durch Konfusion und Diffusion weitestgehend zu zerstören. Man wird erwarten, daß dies nicht für alle Klartextdifferenzen gleich gut gelingt; die Idee hinter der differentiellen Kryptoanalyse ist, sich auf die zu konzentrieren, bei denen es möglichst schlecht gelingt. Wir müssen uns also genauer anschauen, wie eine Klartextdiffernz durch die Runden geht.

Definition: Eine *r-Rundencharakteristik* ist eine Folge

$$\Delta = (\delta_0, \delta_1, \ldots, \delta_r)$$

von Elementen aus $\mathbb{F}_2^{64}$.
Ein Klartextpaar $(x_0, y_0) \in \mathbb{F}_2^{64} \times \mathbb{F}_2^{64}$ *gehört zur Charakteristik* Δ, wenn für die Paare (x_i, y_i) der Ausgaben der i-ten Runde gilt: $x_i \oplus y_i = \delta_i$ für $i = 0, \ldots, r$.
Die *Wahrscheinlichkeit* einer Charakteristik ist die Wahrscheinlichkeit dafür, daß ein Klartextpaar (x_0, y_0) mit Differenz δ_0 zur Charakteristik Δ gehört.

Natürlich sind die Wahrscheinlichkeiten der meisten Charakteristiken sehr gering: In einem idealen Kryptosystem wären alle Ausgabewerte einer Runde gleich wahrscheinlich, die Wahrscheinlichkeit einer r-Rundencharakteristik sollte also im Mittel bei $1/2^{-64\times r}$ liegen, und schon die Wahrscheinlichkeit dafür, daß überhaupt ein Klartextpaar mit gegebener Differenz nach r-Runden eine vorgegebene andere Differenz hat, sollte im allgeinen bei nur etwa 2^{-64} liegen.

Wenn wir eine gute r-Rundencharakteristik gefunden haben, deren Wahrscheinlichkeit deutlich besser ist als 2^{-64}, können wir daher ziemlich sicher sein, daß ein zufällig gewähltes Klartextpaar (x_0, y_0) mit Anfangsdifferenz δ_0 und Enddifferenz δ_r in den r Runden so verschlüsselt wurde, wie es der Charakteristik entspricht. Ist etwa $p = 2^{-44}$, sollte die Wahrscheinlichkeit für alles andere bei nur etwa 2^{-20} oder etwa eins zu einer Million liegen. Konkret heißt das: Wir kennen die Differenzen $x_i \oplus y_i$ mit hoher Wahrscheinlichkeit und können die Verschlüsselung durch die r Runden verfolgen.

Zunächst brauchen wir aber Charakteristiken. Für nur eine Runde ist das sehr einfach: Für jeden Halbblock $d_0 \in \mathbb{F}_2^{32}$ führt der mit 32 Nullen auf 64 Bit aufgefüllte Block δ_0 auf die Charakteristik (δ_0, δ_0) mit Wahrscheinlichkeit eins: Sind nämlich (m_0, m_1) und (m_0', m_1) zwei Klartexte mit (nach der hier stets ignorierten Anfangspermutation) gleicher rechter Hälfte, so wird die linke Hälfte ersetzt durch

$$m_2 = m_0 \oplus f(s_1, m_1) \quad \textit{bzw.} \quad m_2' = m_0' \oplus f(s_1, m_1)\,,$$

und die Differenz ist

$$m_2 \oplus m_2' = \bigl(m_0 \oplus f(s_1, m_1)\bigr) \oplus \bigl(m_0' \oplus f(s_1, m_1)\bigr) = m_0 \oplus m_0' = d_0\,,$$

wie wir das schon oben gesehen haben. Nach Durchgang durch die erste Runde haben wir also die beiden Paare (m_2, m_1) und (m_2', m_1), deren Differenz wieder δ_0 ist.

Leider können wir diese Charakteristik nicht iterieren, denn nach der ersten Runde werden ja die beiden Hälften vertauscht, so daß wir dann (m_1, m_2') und (m_1, m_2') haben, worüber wir nicht soviel sagen können, da nun die FEISTEL-Funktionen verschiedene Werte liefern.

Um trotzdem zu einer Zweirundencharakteristik zu kommen, benutzen wir die Nichtinjektivität der FEISTEL-Funktion f: Wir suchen zwei Halbblöcke m_0 und m_0' derart, daß $f(s, m_0) = f(s, m_0')$ für möglichst viele Schlüssel s; die Differenz $m_0 \oplus m_0'$ bezeichnen wir mit d_0.

Zwei beliebige Klartexte der Form (m_0, m_1) und (m_0', m_1) mit Differenz $m_0 \oplus m_0' = d_0$ gehen in der ersten Runde nach (m_2, m_1) und (m_2', m_1) mit

$$m_2 = m_0 \oplus f(s_1, m_1) \quad \text{und} \quad m_2' = m_0' \oplus f(s_1, m_1)\,,$$

danach werden die linke und die rechte Hälfte vertauscht, so daß die Eingaben zur zweiten Runde gleich (m_1, m_2) und (m_1, m_2') sind, wobei m_2 und m_2' die Differenz d_0 haben. In der zweiten Runde gehen die beiden Klartexte dann nach

$$\bigl(m_1 \oplus f(s_2, m_2), m_2\bigr) \quad \text{und} \quad \bigl(m_1 \oplus f(s_2, m_2'), m_2'\bigr)\,.$$

Da m_2 und m_2' Differenz d_0 haben, ist dabei mit einer gewissen Wahrscheinlichkeit

$$f(s_2, m_2) = f(s_2, m_2')\,.$$

Falls dem so sein sollte, haben wir $(0, d_0)$ als Differenz zwischen den Ausgabewerten. Es gibt daher eine Zweirundencharakteristik der Form

$$\big((d_0, 0), (d_0, 0), (0, d_0)\big)\ ,$$

die mit einer gewissen Wahrscheinlichkeit p auftritt. Diese Charakteristik kann offensichtlich beliebig oft iteriert werden, denn bevor ihre Ausgabewerte in die nächste Runde gehen, werden die linke und die rechte Hälfte vertauscht, so daß wir wieder die Ausgangsdifferenz haben. Damit haben wir für jedes n eine n-Rundencharakteristik gefunden; ihre Wahrscheinlichkeit ist $p^{[n/2]}$, wobei die GAUSS-Klammer $[x]$ die größte ganze Zahl $\leq x$ bezeichnet.

Diese Charakteristik ist umso nützlicher, je größer p ist. Wie sich zeigt, kann p nur dann größer als null sein, wenn mindestens drei benachbarte S-Boxen aktiv sind; die größten Wahrscheinlichkeiten sind also zu erwarten bei *genau* drei aktiven S-Boxen. Systematisches Probieren zeigt, daß der beste erreichbare Wert dann

$$p = \frac{14}{64} \cdot \frac{8}{64} \cdot \frac{10}{64} = \frac{35}{8192} \approx 0{,}004272461 \approx \frac{1}{234} \approx 2^{-7{,}870716983}$$

ist. Er wird erreicht für

$$d_0 = (19\,60\,00\,00)_{\text{hex}} \quad \text{und} \quad d_0 = (1B\,60\,00\,00)_{\text{hex}}\ .$$

Damit haben wir also für beliebiges n eine n-Rundencharakteristik gefunden; leider ist sie aber nicht für jedes n brauchbar: Für 16 Runden ist ihre Wahrscheinlichkeit nur etwa

$$2^{-7{,}870716983\times 8} \approx 2^{-62{,}96573586}\ ,$$

wir bräuchten also mindestens 2^{63} Klartextpaare bekannter Differenz, um eines zu dieser Charakteristik zu finden, während wir mit nur 2^{56} Versuchen alle Schlüssel durchprobieren könnten. Tatsächlich reichten sogar bereits 2^{55} Versuche, denn nimmt man das Einserkomplement von Schlüssel und Klartext, so entsteht das Einserkomplement des Schlüsseltexts. Deshalb sind auch die 14- und die 15-Rundencharakteristik, deren Wahrscheinlichkeiten bei

$$2^{-7{,}870716983\times 7} \approx 2^{-55{,}09501888}$$

liegen, nicht sonderlich interessant; erst die 13-Rundencharakteristik liefert mit

$$p \approx 2^{-7{,}870716983\times 6} \approx 2^{-47{,}22430190}$$

eine halbwegs interessante Wahrscheinlichkeit.

Wählen wir ein Halbwort m_0 derart, daß $m_0 \oplus f(s, m_1)$ für möglichst viele Schlüssel s gleich ist! Für einen solchen Schlüssel sind dann

$$m_2 = m_0 \oplus f(s, m_1) \quad \text{und} \quad m_2' = m_0' \oplus f(s, m_1)$$

Außerdem interessieren wir uns nicht für die Wahrscheinlichkeitsverteilung der Chiffretexte – Chiffretext ist schließlich das, was wir immer haben – sondern für Klartext oder besser noch den Schlüssel bei gegebenem Chiffretext.

Dazu nutzen wir aus, daß Eingabewert der S-Boxen nicht der ersten Runde nicht der Klartext ist, sondern der mit gewissen Schlüsselbits geXORte Klartext. Wenn wir nun für ein Klartextpaar mit gegebener Differenz die Ausgabedifferenz kennen, haben wir die möglichen Eingabepaare der S-Boxen, deren Differenzen ja genau dieselben sind wie für das Klartextpaar, von 64 auf eine erheblich kleinere Zahl reduziert. Für jedes dieses möglichen Paare können wir die entsprechenden Schlüsselbits durch XOR mit dem tatsächlichen Klartext berechnen und haben somit eine relativ kleine Anzahl potentieller Schlüsselteile. Wenn wir das ganze für hinreichend viele Klartextpaare wiederholen, sollte der für die jeweilige S-Box zuständige Schlüsselanteil relativ bald eindeutig feststehen.

DES mit nur einer Runde ist auf diese Weise also relativ einfach zu entschlüsseln, falls wir genügend viele Paare von Klartext mit fester Differenz haben. Diese können wir uns nur verschaffen durch eine Attacke mit wählbarem Klartext, also der schwierigsten Form der Attacke.

Auch diese Attacke liefert aber nicht die Ausgabedifferenzen der ersten Runde, sondern nur die der *letzten.* Der Ansatz der differentiellen Kryptoanalyse des DES ist daher folgender:

1. Man wähle eine geeignete Differenz zwischen Klartexten.

2. Dazu erzeuge man hinreichend viele Paare von Klartextblöcken mit dieser Differnz, verschlüssele sie und behalte nur die so berechneten Chiffretextpaare.
3. Durch Analyse der Klartextdifferenzen und des Verhaltens der S-Boxen in den verschiedenen Runden bestimme man die zu erwartende Wahrscheinlichkeitsverteilung der Eingabedifferenzen der letzten Runde.

Differentielle Kryptanalyse war der erste Ansatz, DES mit geringerem Aufwand als der vollständigen Durchsuchung des Schlüsselraums zu zu brechen. Da aber, wie bereits erwähnt, die Designer des DES die differentielle Kryptanalyse schon kannten lange bevor sie in der offenen Literatur auftauchte und den Algorithmus so gut wie möglich dagegen immun machten, ist diese Attacke nicht sehr praktikabel: Selbst bei Angriffen mit frei wählbarem Klartext braucht man über 2^{40} Paare aus Klartext und Chiffretext, um den Schlüssel zu finden.

d) Lineare Kryptanalyse

Eine leichte Verbesserung bietet die kurz später entdeckte *lineare Kryptanalyse:* Zwar sind die Ausgabebits der S-Boxen nach Design-Kriterium ($S2$) auch nicht näherungsweise lineare Funktionen der Eingabebits, aber es kann dennoch vorkommen, daß man eine Linearkombination von Ausgabebits mit einer Wahrscheinlichkeit, die deutlich über 50% liegt durch eine Linearkombination der Eingabebits vorhersagen kann. Mit hinreichend vielen Paaren aus Klartext und Chiffretext Läßt sich dadurch ein niedrigdimensionaler affiner Unterraum des Schlüsselraums $\mathbb{F}_2^{56}$ finden, in dem der Schlüssel mit hoher Wahrscheinlichkeit liegen muß. Die vollständige Durchsuchung dieses Unterraums ist unproblematisch, so daß der Schlüssel mit hoher Wahrscheinlichkeit gefunden werden kann.

Auch für diese Attacke sind allerdings unrealistisch viele Paare aus Klartext und Chiffretext erforderlich (in einer Variante genügen noch mehr reine Chiffretexte), so daß der Gesamtaufwand nicht wirklich geringer sein dürfte als die vollständige Durchsuchung des Schlüsselraums.

Nach allem, was in der offenen Literatur bekannt ist, gibt es also zum Knacken des DES keine wesentlich bessere Alternative zur vollständigen Durchsuchung des Schlüsselraums.

e) DES-Cracker

Der erste in der offenen Literatur dokumentierte realistische Angriff auf DES war denn auch die vollständige Durchsuchung des Schlüsselraums. Eine amerikanische Bürgerrechtsorganisation, die *Electronic Frontier Foundation (EFF),* konstruierte eine Maschine mit Spezialhardware zum Knacken von DES mit Chiffretext allein.

Die *Electronic Frontier Foundation* wurde 1990 nach dem großen *Hacker Crackdown* in den USA gegründet; Initiatoren waren unter anderem JOHN PERRY BARLOW, bekannt vor allem durch die Lieder, die er für *The Grateful Dead* schrieb, JOHN GILMORE, einer der Pioniere sowohl von *Sun Microsystems* als auch der *Free Software Foundation,* MITCHELL KAPOR, der Gründer von *Lotus,* sowie STEVE WOZNIAK, einer der beiden Gründer von *Apple.*

Ihr Ansatz ist im wesentlichen der unseres guten alten Feinds, des BAYESschen Gegners: Kodiert man einen englisch- oder deutschsprachigen Klartext im ASCII-Code sollten dem BAYESschen Gegner ein bis zwei Blöcke Chiffretext ausreichen, um den Schlüssel zu finden.

Natürlich verfügen selbst die vier obengenannten Gründer der *Electronic Frontier Foundation* nicht über die unbegrenzten Mittel, die der BAYESsche Gegner einsetzen kann; verglichen mit vielen anderen Gegnern verfügt aber doch jeder von ihnen über beträchtliche Mittel. Trotzdem war das 1997 begonnene und 1998 beendete DES-Cracker-Projekt kein Angriff ohne Rücksicht auf die Kosten: Die *Electronic Frontier Foundation* wollte gerade zeigen, daß DES auch mit begrenzten Mitteln geknackt werden kann. Aus diesem Grund wurde der Ansatz des BAYESschen Gegners an mehreren Stellen optimiert:

Zunächst ist es nicht notwendig, wirklich für *jeden* Schlüssel die bedingte Wahrscheinlichkeit auf Grund des Chiffretexts zu bestimmen: In vielen Fällen werden bei der Entschlüsselung nicht druckbare Zeichen entstehen, so daß schon nach wenigen Byte klar ist, daß die Wahrscheinlichkeit des Schlüssels null ist.

DES Cracker beginnt daher mit der Aussonderung unmöglicher Schlüssel durch massiv parallele Hardware: Die Maschine arbeitet mit zwei Blöcken Chiffretext; sie hat als Kern von EFF entwickelte ASICs *application specific integrated circuits),* die einen 64-Bit-Block mit einem vorgegebenen Schlüssel dechiffrieren können und die Bytes des Ergebnisses auf vom Benutzer einstellbare Bitmuster überprüfen – beispielsweise darauf, ob es sich um ASCII-Codes druckbarer Zeichen handelt. Nur wenn alle Bytes den gewählten Kriterien genügen, wird auch der zweite Block entsprechend untersucht, und wenn auch hier kein im Klartext unmögliches Byte auftaucht, wird der Schlüssel zur weiteren Untersuchung an einen die Maschine steuerenden PC weitergegeben, der eine genauere Untersuchung gemäß dem Ansatz des BAYESschen Gegners durchführt.

Von den 256 möglichen ASCI-Werten sind etwa ein Viertel druckbare Zeichen; da DES eine Ausgabe liefert, die sich nur wenig von einer Zufallsfolge unterscheidet, wird ein Block nur mit einer Wahrscheinlichkeit von etwa $1 : 4^8 = 1 : 65536$ den Test bestehen; für zwei Blöcke liegt die Wahrscheinlichkeit entsprechend bei $1 : 4^{16} = 1 : 2^{32}$. Von den 2^{56} zu untersuchenden Blöcken werden also nur etwa

$$2^{56-32} = 2^{24} = 16\,777\,216$$

an den PC weitergegeben, und die Untersuchung von etwa 17 Millionen Klartextkandidaten ist kein Problem für einen Standard-PC.

Der hauptsächliche Rechenaufwand liegt in der Voruntersuchung der Blöcke durch die ASICs; je nachdem, wie viele von diesen parallel arbeiten, kann dies mehr oder weniger schnell gehen.

Die tatsächlich gebaute Maschine enthält $1536 = 3 \times 2^9$ ASICs, von denen jedes aus 24 parallel arbeitenden Sucheinheiten besteht; insgesamt können also jeweils 36 864 Schlüssel parallel untersucht werden. Jede Sucheinheit kann zweieinhalb Millionen Schlüssel pro Sekunde untersuchen, die gesamte Maschine also etwas über 92 Milliarden.

Insgesamt müssen 2^{56} Schlüssel untersucht werden; im Mittel wird man nach 2^{55} Versuchen den richtigen gefunden haben. Dafür braucht man

$$\frac{2^{55}}{92\,160\,000\,000} \approx 390\,937 \text{ Sekunden} \approx 108{,}5 \text{ Stunden} \approx 4{,}5 \text{ Tage}\,.$$

Die Maschine ist skalierbar: Jeweils 64 Chips sitzen auf einem Board und 12 Boards in einer Chassis (einer ehemaligen SUN); die gebaute Maschine besteht aus dem steuernden PC zusammen mit zwei Chassis; der PC könnte aber auch mit deutlich mehr als zwei Chassis arbeiten und auch mehrere PCs wären denkbar.

In der gebauten Version kostete DES Cracker 210 000 \$, wovon 80 000 \$ Entwicklungskosten waren; der Bau eines zweiten Exemplars wäre also für 130 000 \$ möglich, wobei der Preis bei Serienproduktion wohl noch deutlich gesenkt werden könnte. Mit einer Investition in Größenordnung einer Million Dollar käme man also wohl in den Bereich weniger Stunden für das Auffinden eines Schlüssels.

Da eine Million Dollar auch für Geheimdienste kleiner Länder und (etwas kreative Buchhaltung vorausgesetzt) Großunternehmen kein Problem sind, sollte damit klar sein, daß DES auf keinen Fall mehr heutigen Sicherheitsanforderungen genügt.

Die EFF veröffentlichte sowohl die komplette Hardware-Spezifikation als auch die Software von DES-Cracker; in gedruckter Form findet man sie im Buch

ELECTRONIC FRONTIER FOUNDATION: *Cracking DES. Secrects of Encryption Research, Wiretap Politics & Chip Design,* O'Reilly, 1998

Online ist das Buch unter anderem verfügbar unter

`http://cryptome.org/cracking-des.htm` ;

die DES Cracker Seite der EFF ist

`www.eff.org/Privacy/Crypto/Crypto_misc/DESCracker/` .

§4: Modifikationen

a) Mehrfacher DES

Viele Unternehmen, insbesondere im Bankenbereich, haben viel Geld in DES-Hardware investiert und haben daher wenig Interesse, auf ein neues Verfahren umzusteigen – ganz abgesehen davon, daß 1998 noch kein allgemein anerkannter Nachfolgealgorithmus zur Verfügung stand. (Mit der heutigen Situation werden wir uns im nächsten Kapitel beschäftigen.)

Deshalb bot sich ein Verfahren an, das bereits kurz nach Normierung des DES vorgeschlagen wurde, um den kleinen Schlüsselraum zu vergrößern: Man verschlüssele mehrfach mit verschiedenen Schlüsseln.

Eine zweifache Verschlüsselung hängt von 2^{112} statt von 2^{56} Schlüsselbit ab; der Aufwand für eine Durchsuchung des gesamten Schlüsselraums steigt also um den Faktor 2^{56} gegenüber dem gewöhnlichen DES, was eine Maschine nach Art des DES Crackers nach heutigen Stand der Technik nicht mehr in akzeptabler Zeit durchführen kann.

Eine mehrfache Verschlüsselung bietet allerdings nur dann Vorteile, wenn die Hintereinanderausführung zweier DES-Verschlüsselungen nicht äquivalent zur einfachen DES-Verschlüsselung mit einem anderen Schlüssel ist.

Um dies zu untersuchen, müssen wir noch einmal zurück zur grundsätzlichen Struktur von Blockchiffren: Blockchiffren sind Permutationen auf der Menge aller Blöcke; im Falle von DES also auf der Menge aller bijektiver Abbildungen von der Menge aller 64-Bit-Blöcke auf sich selbst.

Die 2^{56} durch DES definierten Permutationen bilden natürlich nur eine winzige Teilmenge der Gruppe aller $2^{64}!$ solcher Permutationen; falls diese Teilmenge eine Gruppe sein sollte (oder in einer relativ kleinen Untergruppe liegt), kann die mehrfache Anwendung von DES keine (oder nur wenig) zusätzliche Sicherheit bieten.

Wir brauchen daher Informationen darüber, wie groß die kleinste Gruppe ist, die alle 2^{56} DES-Permutationen enthält. Über deren genaue Struktur und Elementanzahl ist leider nichts bekannt, man kann aber immerhin untere Schranken angeben: Für jeden einzelnen DES-Schlüssel kann man die zugehörige Substitution so lange wiederholen, bis die Identität entsteht; da $2^{64}!$ eine zwar große, aber endliche Zahl ist, muß dies nach endlich vielen Schritten der Fall sein.

Angenommen, bei solchen Berechnungen mit verschiedenen Schlüsseln ergaben sich die Ordnungen $n_1, n_2, \ldots, n_r$. Dann enthält die kleinste Gruppe, in der alle DES-Substitutionen liegen, zyklische Untergruppen der Ordnungen $n_1, \ldots, n_r$. Nach einem einfachen Satz der Gruppentheorie, dem Satz von LAGRANGE, müssen die Zahlen $n_1, \ldots, n_r$ dann die Ordnung der gesamten Gruppe teilen; diese ist also mindestens gleich dem kleinsten gemeinsamen Vielfachen der n_i.

Experimentelle Untersuchungen zweier Wissenschaftler von Bell Northern Research in Ottawa zeigten, daß die Ordnung der erzeugten Untergruppe größer als ,9 × 10^{2499} sein muß, was sehr deutlich über 2^{56} liegt; siehe ihre Arbeit

KEITH W. CAMPBELL, MICHAEL J. WIENER: DES is not a Group, *Crypto '92, Springer Lecture Notes in Computer Science* **740** (1993), 512–520

Durch mehrfache Anwendung von DES läßt sich also die Sicherheit erhöhen.

b) Doppelter DES

Beim doppelten DES hat man dann, wie bereits erwähnt, einen Schlüsselraum mit 2^{112} Elementen. Leider muß man diesen aber, falls man genügend viel Speicherplatz hat, nicht vollständig durchsuchen, um die Schlüssel zu finden: Falls man ein Paar aus einander entsprechendem Klartext und Chiffretext hat, reicht es, in einer sogenannten *meet in the middle attack* den Klartext mit allen 2^{56} möglichen Schlüsseln zu *ver*schlüsseln und den Chiffretext mit allen 2^{56} Schlüsseln zu *ent*schlüsseln; sobald man einen Block gefunden hat, der bei beiden Operationen auftritt, hat man mit an Sicherheit grenzender Wahrscheinlichkeit die beiden Schlüssel gefunden.

In der Praxis würde man wohl den beträchtlichen Speicheraufwand für 2^{56} Blöcke scheuen und versuchen, ihn in einem *time-memory-tradeoff* auf Kosten einer höheren Rechenzeit zu kompensieren; trotzdem zeigt die grundsätzliche Möglichkeit einer solchen Attacke, daß der Sicherheitsvorsprung, den ein doppelter DES gegenüber dem einfachen bietet, deutlich geringer ist, als man auf den ersten Blick meint. In der Praxis spielt der doppelte DES daher keine Rolle.

c) Dreifach DES

Der nächste Schritt zu Erhöhung der Komplexität besteht in einer dreifachen Anwendung von DES. Auch hier kann man wieder eine *meet in the middle attack* anwenden, aber auf dem Weg zur Mitte muß von mindestens einer der beiden Seiten aus DES zweimal mit verschiedenen Schlüsseln anwenden, hat dann also einen Aufwand der Größenordnung 2^{112}. Dreifach DES oder, wie man meist sagt, Triple DES, gilt daher heute noch als sicher: Nachdem die DES-Cracker-Attacke publik geworden war, zog die amerikanische Regierung die Zulassung von DES für weniger geheime Nachrichten im Regierungsbereich zurück und verlangte stattdessen Triple DES, genauso wurden in Deutschland die Euroscheckkarten ersetzt durch neue Karten, die auf Triple DES anstelle des einfachen DES beruhen.

Soweit bekannt, bietet ein dreifacher DES mit drei verschiedenen Schlüsseln keine zusätzliche Sicherheit gegenüber einem mit nur zwei Schlüsseln, wobei die erste und die dritte Verschlüsselung denselben Schlüssel benutzen. Meist wird Triple DES in folgender Weise angewandt: Ein Block wird mit dem ersten Schlüssel *ver*schlüsselt, mit dem zweiten *ent*schlüsselt und dann noch einmal mit dem ersten *ver*schlüsselt.

d) Operationsmodi

Bislang haben wir Blockchiffren stets als Permutationen auf der Menge aller Blöcke betrachtet. Diese einfachste Anwendung einer Chiffre bezeichnet man als den *electronic code book* Modus, kurz ECB, da es zumindest im Prinzip möglich wäre, alle Blöcke und ihre Chiffrierung für einen gegebenen Schlüssel in einem Buch festzuhalten.

Für manche Anwendungen hat dieser ECB-Modus allerdings Schwächen und Sicherheitslücken: Falls beispielsweise nur wenige kurze Nachrichten (etwa „Ja“ und „Nein“) häufig übertragen werden, weiß ein Gegner, welche der gesendeten Nachrichten miteinander übereinstimmen und kann daraus vielleicht auch auf den Inhalt schließen.

Manchmal kann ein Gegner auch einfach dadurch Schaden anrichten, daß er unbemerkt die Reihenfolge von Nachrichtenblöcken vertauscht oder aber einen Nachrichtenblock *mehrfach* übermittelt. Auch gegen solche Attacken bietet ECB keinen Schutz.

Wir wollen daher im folgenden die drei weiteren für DES normierten Operationsmodi kennenlernen, die zumindest in diesen Fällen größere Sicherheit bieten.

Grundsätzlich lassen sich alle diese Modi auf *jede* Blockchiffre anwenden; sie werden hier im Zusammenhang mit DES behandelt, da für die Nachfolgechiffre AES bislang noch keine Operationsmodi festgelegt sind.

Alle drei Operationsmodi hängen außer vom DES-Schlüssel noch von einem Anfangsblock ab; dieser muß nicht geheim sein, kann aber als zusätzlicher Sicherheitsfaktor geheimgehalten werden.

Beim *cipher block chaining* (CBC) Modus wird zunächst der Anfangsblock mittels XOR zum ersten Block der Nachricht addiert; das Ergebnis wird verschlüsselt und übertragen. In jedem folgenden Schritt wird das letzte Verschlüsselungsergebnis zum zu verschlüsselnden Block addiert und diese Summe verschlüsselt und übertragen.

Da somit jeder übertragene Block von *allen* Vorgängerblocks abhängt, kann kein Gegner unbemerkt einzelne Blöcke unterdrücken, duplizieren oder aber in ihrer Reihenfolge verändern.

Diesem Vorteil steht der Nachteil gegenüber, daß ein etwaiger Übertragungsfehler nicht nur *einen* Block betrifft (und damit unentschlüsselbar macht), sondern auch noch den Folgeblock. Man überlegt sich aber leicht, daß danach wieder alles korrekt entschlüsselbar ist. Entsprechend führt ein verlorener Block nur zur Nichtentzifferbarkeit *eines* Folgeblocks, danach ist die Chiffre wieder synchronisiert.

Die beiden noch folgenden Modi sind nützlich, wenn Daten in Echtzeit übertragen werden sollen, die kürzer sind als die Blocklänge; die Länge der zu übertragenen Daten sei k Bit. In Anwendungen des DES ist meist $k = 8$, gelegentlich auch $k = 1$.

Bei *cipher feedback* (CFB) Modus wird zunächst ein Anfangsblock gewählt, bei dem rechts k Bit stehen; der Rest dieses Blocks besteht aus

Nullen. Dieser Block kommt in ein Register, das mit DES verschlüsselt wird. Sodann werden die *ersten* k Bit des Ergebnisses zu den ersten k Bit der Nachricht addiert und dies wird übertragen.

In jedem folgenden Schritt wird der Inhalt des Registers um k Bit (nicht-zyklisch) nach links verschoben, und die k zuletzt übertragenen Bits werden am rechten Ende eingesetzt. Das Register wird dann wieder mit DES verschlüsselt, und die ersten k Bit werden zum nächsten Nachrichtenblock addiert und übertragen.

Auch hier hängt wieder jedes übertragene Wort aus k Bit von allen Vorgängern ab; Übertragungsfehler beeinflussen die Folgewörter so lange, wie noch ein fehlerhaft übertragenes Chiffrewort im Register steht.

Bei Satellitenübertragungen, bei denen Bitfehler relativ häufig sind, ist dies nicht tolerierbar; hier verwendet man stattdessen den *output feedback* (OFB) Modus, der im wesentlichen genauso funktioniert wie CFB, nur daß jetzt anstelle des Chiffretexts die linken k Bit des Registers rechts eingefügt werden, d.h. das Register wird zyklisch um k Bit verschoben.

In diesem Modus hängt der Inhalt des Registers weder vom Klartext noch vom Chiffretext ab; DES wird also nur dazu benutzt, einen Schlüsselsgtrom zu erzeugen, der zur Nachricht addiert wird. Im OFB Modus haben wir somit eine Stromchiffre, die zwischen dem *one time pad* und VIGENÈRE liegt: Der Schlüsselstrom hängt zwar nur von wenigen Bit ab, entsteht daraus aber nicht durch bloße Wiederholung, sondern via DES (oder einer anderen Blockchiffre) mit einem geheimen Schlüssel.

Bei dieser Übertragungsart erzeugen Sender und Empfänger die Schlüsselströme unabhängig von der Nachricht, ein Bitfehler bei der Übertragung betrifft also nur ein einziges Bit.

Diesem Vorteil steht der Nachteil entgegen, daß man hier für jede Übertragung einen anderen Anfangsblock und/oder Schlüssel wählen muß, denn ansonsten wird mehrfach derselbe Schlüsselstrom verwendet, ein Gegner weiß also, welche k-Bit-Blöcke jeweils mit demselben Schlüssel chiffriert wurden und kann Häufigkeitsanalysen anwenden.

§5: Literatur

Da DES rund 25 Jahre lang *das* Standardverfahren für ernsthafte symmetrische Verschlüsselung im zivilen Bereich war, ist er natürlich in praktisch jedem Lehrbuch der Kryptologie ausführlich beschrieben, z.B. in

JAN C.A. VAN DER LUBBE: *Basic Methods of cryptography,* Cambridge University Press, 1998

oder

JOHANNES BUCHMANN: *Einführung in die Kryptograohie,* Springer, 1999

Speziellere Fragen sind außer in den bereits im Text zitierten Arbeiten und Büchern auch in fast jeder Konferenz über Kryptologie behandelt; insbesondere gilt dies für Tagungen wie *Crypto, Eurocrypt* und *Asiacrypt,* deren Proceeding jeweils in den Springer Lecture Notes in Computer Science erscheinen.

Kapitel 4
Der Advanced Encryption Standard Rijndael

§ 1: Geschichte und Auswahlkriterien

DES wurde in Zusammenarbeit mit der National Security Agency der Vereinigten Staaten von IBM entwickelt und dann als amerikanischer Standard verkündet. Diese Vorgehensweise weckte von Anfang an den Verdacht, daß möglicherweise eine „Falltür“ eingebaut sei, insbesondere da zumindest ursprünglich nicht alle Design-Kriterien publiziert wurden.

Nachdem dreißig Jahre intensiver Kryptanalyse keine Falltür gefunden haben, müßte diese – so vorhanden – zumindest sehr gut versteckt sein; aber egal ob mit oder ohne Falltür: Wie wir im letzten Kapitel gesehen haben, erfüllt DES mit nur 56 Bit Schlüssellänge nicht mehr die heutigen Sicherheitsanforderungen. Ursprünglich war er ohnehin nur für eine Laufzeit von zehn Jahren vorgesehen und wurde nur immer wieder verlängert, weil die meisten Anwender von Kryptographie äußerst träge sind und sich nicht um Fortschritte der Kryptanalyse kümmern.

Obwohl es die Internationale Standardisierungsorganisation ISO ablehnt, ein Kryptoverfahren zu standardisieren (Ein Grund dafür ist die dann befürchtete Bündelung krimineller Energie auf dieses Verfahren), hat das das amerikanische Handelsministerium die Suche nach einem Nachfolgealgorithmus für DES am 2. Januar 1997 international ausgeschrieben; der vorläufige Name des Algorithmus war AES *(Advanced Encryption Standard).* Federführend für die Auswahl war das *National Institute of Standards and Technology* (NIST) in Gaithersburg, Maryland.

Dieses ging von Anfang an davon aus, daß der zu wählende Algorithmus *stärker* sein müsse als Triple DES; er sollte zwanzig bis dreißig Jahre lang anwendbar sein und dementsprechende Sicherheit bieten. Nach einer internationalen Konferenz über die Auswahlkriterien am 15. April 1997 veröffentlichte es am 12. September 1997 die endgültige Ausschreibung.

Minimalanforderung an die einzureichenden Algorithmen waren danach, daß es sich um symmetrische Blockchiffren handeln muß, die mindestens eine Blocklänge von 128 Bit bei Schlüssellängen von 128 Bit, 192 Bit und 256 Bit vorsieht.

Als Kriterien für die Wahl zwischen den einzelnen Algorithmen wurden die folgenden Aspekte genannt:

1. **Sicherheit:** Wie sicher ist der Algorithmus im Vergleich zu den anderen Kandidaten? Inwieweit ist seine Ausgabe ununterscheidbar von der einer Zufallspermutation? Wie gut ist die mathematische Basis für die Sicherheit des Algorithmus begründet? (Im Gegensatz zu DES sollten dieses Mal alle Kriterien publiziert werden.)
2. **Kosten:** Welche Lizensgebühren werden fällig? Wie effizient geht der Algorithmus mit Rechenzeit und Speicherplatz um?
3. **Flexibilität:** Ein Algorithmus, der auf einer Vielzahl von Plattformen implementierbar ist (PCs, 8-Bit-Prozessoren, ATM-Netze, HDTV, …) ist vorzuziehen, genauso einer, der auch als Stromchiffre, kryptographisch sichere Hash-Funktion oder ähnliches verwendet werden kann.
4. **Software:** Der Algorithmus *muß* auch softwaremäßig effizient implementierbar sein.
5. **Einfachheit:** Der Aufbau des Algorithmus soll möglichst einfach sein.

Nicht nur die Art der Suche unterschied sich also beträchtlich von der DES-Entwicklung hinter verschlossenen Türen, auch die Auswahlkriterien hatten sich stark geändert: DES war noch in erster Linie für Hardware entworfen worden; manche Aspekte von DES wie etwa die für die kryptographische Sicherheit völlig irrelevante Anfangspermutation hatten wohl keinen anderen Sinn als die Verlangsamung von Software-

Implementierungen.

Inzwischen hat freilich die Erfahrung mit der Kryptographie in der Geschäftswelt gezeigt, daß man dort den Kostenaufwand für Spezialhardware scheut und sich lieber mit den dubiosen Verfahren begnügt, die in der ohnehin vorhandenen Office-Software eingebaut ist (Preis für das Knacken durch spezialisierte Unternehmen je nach Programm zwischen 40 \$ und 250 \$), wohingegen sich die Hacker innerhalb und außerhalb der Geheimdienste durch einen hohen Aufwand nur wenig beeindrucken lassen.

Die Ausschreibung führte bis zum Abgabeschluß am 15. Juni 1998 zu fünfzehn Vorschlägen aus aller Welt. Diese wurden auf einer ersten AES-Konferenz vom 20.–22. August 1998 in Ventura, Kalifornien vorgestellt und der Fachwelt zur Analyse und Kommentierung empfohlen. Tatsächlich wurden vierzehn der fünfzehn Algorithmen schon vor der Konferenz veröffentlich; lediglich der Algorithmus *Magenta* der Deutschen Telekom wurde erst am 20. August auf der Konferenz selbst bekanntgegeben. Er war auch der einzige Algorithmus, der bereits während der Konferenz geknackt wurde in einer auf den 20. August 1998 datierten Arbeit von E. BIHAM, A. BIRYUKOV, N. FERGUSON, L. KNUDSEN, B. SCHNEIER und A. SHAMIR, die noch auf der Konferenz verteilt wurde.

Die zweite AES-Konferenz am 22. und 23. April 1999 in Rom führte zu einer ersten Diskussion der Ergebnisse und Empfehlungen, welche der fünfzehn Algorithmen weiter betrachtet werden sollten. Die endgültige Entscheidung des NIST wurde am 9. August 1999 bekanntgegeben:

Bei fünf der eingereichten Algorithmen hatte sich herausgestellt, daß sie entweder mit einem Aufwand zu knacken sind, der deutlich unter der Durchsuchung des gesamten Schlüsselraums liegt (darunter natürlich auch Magenta) oder aber, daß es zu viele „schwache" Schlüssel gibt. Da diese fünf Algorithmen auch mit die langsamsten Kandidaten waren, sprach nichts dafür, sie noch weiter zu betrachten.

Fünf weitere Kandidaten zeigten ebenfalls Schwächen, die zwar für sich allein betrachtet nicht so schwerwiegend waren, daß man die Verfahren eliminieren müßte; da diese fünf Kandidaten andererseits nirgends entscheidende Vorteile boten, wurden auch sie eliminiert, so daß noch fünf Finalisten übrigblieben: Drei aus USA (MARS von IBM, RC6 von RSA und Twofish von Counterpane), Serpent von drei Kryptologen aus England, Israel und Norwegen, und Rijndael von zwei Kryptologen, die von der Katholischen Universität Leuven in Belgien kommen.

Bei der dritten AES-Konferenz am 13. und 14. April 2000 in New York bekam Rijndael 86 Stimmen, Serpen 59, Twofish 31, RC6 23 und MARS 13, so daß es nicht weiter verwunderte, daß das NIST am 2. Oktober 2000 Rijndael für den vorgeschlagenen Standard nominierte. Als offizieller Federal Information Processing Standard FIPS-197 verkündete ihn das amerikanische Handelsministerium am 6. Dezember 2001.

Rijndael hat seinen Namen von seinen beiden Autoren JOAN DAEMEN und VINCENT RIJMEN, die 1995 beziehungsweise 1997 an der Elektrotechnischen Fakultät der Katholischen Universität Leuven über Themen aus der Kryptographie promoviert hatten; RIJMEN arbeitet immer noch dort als Postdoc. Aufgrund der Wahl von Rijndael zum AES wurden sie als flämische Persönlichkeiten des Jahres 2000 gewählt.

Als Aussprachehilfe für Personen, die kein Niederländisch, Flämisch, Surinamer oder Afrikaans sprechen, geben die Autoren folgende englische Approximationen des Wortes „Rijndael": „Reign Dahl", „Rain Doll" und „Rhine Dahl".

§ 2: Algebraische Vorbereitungen

Alle Operationen von Rijndael sind algebraisch definiert. Es gibt einmal Operationen auf Byte-Ebene, die durch die Grundrechenarten im Körper mit 256 Elementen realisiert sind; dazu kommen Operationen auf 4-Byte-Wörtern, die im Polynomring über diesem Körper definiert sind. Wir wollen uns langsam an die Definition dieses Körpers und dieses Ringes herantasten; im Hinblick auf Algorithmen der *public key*-Kryptographie, die uns später in der Vorlesung interessieren werden, ist die Präsentation geringfügig allgemeiner, als es für Rijndael notwendig wäre.

a) Der Euklidische Algorithmus für natürliche Zahlen

Wir werden den EUKLIDischen Algorithmus im folgenden hauptsächlich für Polynome brauchen; zum besseren Verständnis der Idee (und für spätere Anwendungen in der *public key*-Kryptographie) betrachten wir ihn aber zunächst im wohlbekannten Fall der natürlichen Zahlen.

Ziel ist die Berechnung des größten gemeinsamen Teilers (ggT) zweier natürlicher Zahlen a und b, also der größten natürlichen Zahl d, die sowohl a als auch b teilt; wir schreiben kurz

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

Grundidee des EUKLIDischen Algorithmus ist die Anwendung der Division mit Rest: Für je zwei natürliche Zahlen x und y gibt es nichtnegative ganze Zahlen q und r, so daß

$$x = qy + r \quad \text{und} \quad 0 \le r < y$$

ist. Alsdann ist

$$\mathrm{ggT}(x, y) = \mathrm{ggT}(y, r)\,,$$

denn wegen der beiden Gleichungen

$$x = qy + r \quad \text{und} \quad r = x - qy$$

teilt jeder gemeinsame Teiler von x und y auch r, und jeder gemeinsame Teiler von y und q teilt x.

Der EUKLIDische Algorithmus nutzt dies aus, um die Zahlen, deren ggT bestimmt werden muß, sukzessive zu verkleinern, bis der ggT zweier Zahlen berechnet werden muß, von denen die eine Teiler der anderen ist; in diesem Fall ist natürlich die kleinere der beiden Zahlen gleich dem ggT.

Formal sieht der EUKLIDische Algorithmus zur Berechnung des ggT zweier natürlicher Zahlen a und b folgendermaßen aus:

Schritt 0: Setze $r_0 = a$ und $r_1 = b$

Schritt i, $i \ge 1$**:** Falls $r_i = 0$ ist, endet der Algorithmus mit $\mathrm{ggT}(a, b) = r_{i-1}$; andernfalls dividiere man r_{i-1} mit Rest durch r_i mit dem Ergebnis $r_{i-1} = q_i r_i + r_{i+1}$.

(Bei einer tatsächlichen Implementierung bieten sich hier natürlich einige offensichtliche Optimierungen an.)

Der Algorithmus muß nach endlich vielen Schritten enden, denn bei der Division mit Rest ist stets $0 \le r_{i+1} < r_i$, so daß r_i mit jedem Schritt kleiner wird, was bei natürlichen Zahlen nicht unbegrenzt möglich ist. Da außerdem in jedem Schritt

$$\mathrm{ggT}(r_{i-1}, r_i) = \mathrm{ggT}(r_i, r_{i+1})$$

ist und im letzten Schritt, wenn r_{i-1} den vorigen Wert r_{i-2} teilt,

$$\mathrm{ggT}(r_{i-1}, r_{i-2}) = r_{i-1}$$

ist, folgt induktiv

$$\mathrm{ggT}(a, b) = r_{i-1}\,,$$

so daß der Algorithmus das richtige Ergebnis liefert.

Wir brauchen für das folgende vor allem eine Erweiterung des Grundalgorithmus, den sogenannten erweiterten EUKLIDischen Algorithmus: Die Gleichung

$$r_{i-1} = q_i r_i + r_{i+1}$$

läßt sich umschreiben als

$$r_{i+1} = q_i r_i - r_{i+1}\,,$$

so daß r_{i+1} eine ganzzahlige Linearkombination von r_i und r_{i-1} ist. Da entsprechend auch r_i Linearkombination von r_{i-1} und r_{i-2} ist, folgt induktiv, daß der ggT von a und b als ganzzahlige Linearkombination von a und b dargestellt werden kann.

Algorithmisch sieht diese Erweiterung folgendermaßen aus:

Schritt 0: Setze $r_0 = a$, $r_1 = b$, $\alpha_0 = \beta_1 = 1$ und $\alpha_1 = \beta_0 = 0$. Mit $i = 1$ ist dann

$$r_{i-1} = \alpha_{i-1} a + \beta_{i-1} b \quad \text{und} \quad r_i = \alpha_i a + \beta_i b\,.$$

Diese Relationen werden in jedem der folgenden Schritte erhalten:

Schritt i, $i \ge 1$**:** Falls $r_i = 0$ ist, endet der Algorithmus mit

$$\mathrm{ggT}(a, b) = r_{i-1} = \alpha_{i-1} a + \beta_{i-1} b\,.$$

Andernfalls dividiere man r_{i-1} mit Rest durch r_i mit dem Ergebnis

$$r_{i-1} = q_i r_i + r_{i+1}\,.$$

Dann ist

$$\begin{aligned} r_{i+1} = q_i r_i - r_{i-1} &= q_i(\alpha_i a + \beta_i b) - (\alpha_{i-1} a + \beta_{i-1} b) \\ &= (q_i\alpha_i - \alpha_{i-1})a + (q_i\beta_i - \beta_{i-1})b\,; \end{aligned}$$

man setze also

$$\alpha_{i+1} = q_i\alpha_i - \alpha_{i-1} \quad \text{und} \quad \beta_{i+1} = q_i\beta_i - \beta_{i-1}\,.$$

Genau wie oben folgt, daß der Algorithmus für alle natürlichen Zahlen a und b endet und daß am Ende der richtige ggT berechnet wird; außerdem sind die α_i und β_i so definiert, daß in jedem Schritt $r_i = \alpha_i a + \beta_i b$ ist, insbesondere ist also im letzten Schritt der ggT als Linearkombination der Ausgangszahlen dargestellt.

Als Beispiel wollen wir den ggT von 200 und 148 als Linearkombination darstellen. Im nullten Schritt haben wir 200 und 148 als die trivialen Linearkombinationen

$$200 = 1\cdot 200 + 0\cdot 148 \quad \text{und} \quad 148 = 0\cdot 200 + 1\cdot 148\,.$$

Im ersten Schritt dividieren wir, da 148 nicht verschwindet, 200 mit Rest durch 148:

$$200 = 1\cdot 148 + 52 \quad \text{und} \quad 52 = 1\cdot 200 - 1\cdot 148\,.$$

Da auch $52 \neq 0$, dividieren wir im zweiten Schritt 148 durch 52:

$148 = 2\cdot 52 + 44$ und $44 = 148 - 2\cdot(1\cdot 200 - 1\cdot 148) = 3\cdot 148 - 2\cdot 200\,.$

Auch $44 \neq 0$; wir machen also weiter: $52 = 1\cdot 44 + 8$ und

$8 = 52 - 44 = (1\cdot 200 - 1\cdot 148) - (3\cdot 148 - 2\cdot 200) = 3\cdot 200 - 4\cdot 148\,.$

Im nächsten Schritt erhalten wir $44 = 5\cdot 8 + 4$ und

$4 = 44 - 5\cdot 8 = (3\cdot 148 - 2\cdot 200) - 5\cdot(3\cdot 200 - 4\cdot 148) = 23\cdot 148 - 17\cdot 200\,.$

Bei der Division von acht durch vier schließlich ist der Divisionsrest null; damit ist vier der ggT von 148 und 200 und kann in der angegebenen Weise linear kombiniert werden.

Auch wenn wir es im Augenblick nicht ganz so allgemein brauchen, sei zumindest kurz erläutert, wie der erweiterte EUKLIDische Algorithmus zum Lösen diophantischer Gleichungen verwendet werden kann, von Gleichungen also, bei denen nur ganzzahlige Lösungen interessieren. Wir betrachten nur die einfache lineare Gleichung

$$ax + by = c \quad \text{mit} \quad a, b, c \in \mathbb{Z}$$

für zwei Unbekannte $x, y \in \mathbb{Z}$.

Der größte gemeinsame Teiler $d = \text{ggT}(a, b)$ von a und b teilt offensichtlich jeden Ausdruck der Form $ax + by$ mit $x, y \in \mathbb{Z}$; falls d kein Teiler von c ist, kann es also keine ganzzahlige Lösung geben.

Ist aber $c = rd$ ein Vielfaches von d und ist

$$d = \alpha a + \beta b$$

die lineare Darstellung des ggT nach dem erweiterten EUKLIDischen Algorithmus, so ist offensichtlich

$$x = r\alpha \quad \text{und} \quad y = r\beta$$

eine Lösung.

Ist (x', y') eine weitere Lösung, so ist

$$a(x - x') + b(y - y') = c - c = 0 \qquad \text{oder} \qquad a(x - x') = b(y' - y)\,.$$

$v = a(x - x') = b(y' - y)$ ist also ein gemeinsames Vielfaches von a und b und damit auch ein Vielfaches des kleinsten gemeinsamen Vielfachen von a und b. Dieses kleinste gemeinsame Vielfache ist ab/d, es muß also eine ganze Zahl m geben mit

$$x - x' = m\cdot\frac{b}{d} \quad \text{und} \quad y' - y = m\cdot\frac{a}{d}\,.$$

Die allgemeine Lösung der obigen Gleichung ist somit

$$x = r\alpha - m\cdot\frac{b}{d} \quad \text{und} \quad y = r\beta + m\cdot\frac{a}{d} \quad \text{mit} \quad m \in \mathbb{Z}\,.$$

b) Körper von Primzahlordnung

Die Addition und Multiplikation im Ring $\mathbb{Z}$ der ganzen Zahlen induziert für jede natürliche Zahl n eine Addition und Multiplikation in

$$\mathbb{Z}/n \underset{\text{def}}{=} \{0, 1, \ldots, n-1\}$$

durch

$$a \oplus b \underset{\text{def}}{=} (a+b) \bmod n \quad \text{und} \quad a \otimes b = (a \cdot b) \bmod n\,.$$

Im allgemeinen macht dies jedoch $\mathbb{Z}/n$ nicht zum Körper, denn ist $n = p \cdot q$ Produkt zweier Faktoren, die beide kleiner als n und damit größer als eins sind, so ist $p \odot q = 0$, obwohl weder p noch q gleich null sind. Solche sogenannte *Nullteiler* können in einem Körper nicht vorkommen, denn ist in einem Körper $x \cdot y = 0$ und $x \neq 0$, so folgt

$$y = x^{-1} \cdot x \cdot y = y^{-1} \cdot 0 = 0\,.$$

Für zusammengesetzte Zahlen n ist $\mathbb{Z}/n$ daher kein Körper.

Ist allerdings $n = p$ eine Primzahl, so läßt sich jedes von null verschiedene Element aus $\mathbb{Z}/p$ invertieren: Für jede natürliche Zahl a mit $0 < a < p$ ist der ggT von a und p gleich eins, so daß es nach dem erweiterten EUKLIDischen Algorithmus ganze Zahlen α und β gibt mit

$$\alpha a + \beta p = 1 \qquad \text{oder} \qquad \alpha a - 1 = \beta p\,.$$

Damit ist $\alpha a - 1 = \beta p$ durch p teilbar oder, anders ausgedrückt

$$\alpha a \equiv 1 \bmod p\,.$$

Somit ist $\alpha \bmod p$ ein multiplikatives Inverse zu a.

Wenn wir auf der Menge $\mathbb{F}_p \underset{\text{def}}{=} \{0, 1, \ldots, p-1\}$ eine Addition und Multiplikation einführen durch die Vorschriften

$$a \oplus b \underset{\text{def}}{=} (a+b) \bmod p \quad \text{und} \quad a \odot b \underset{\text{def}}{=} ab \bmod p$$

definieren, ist klar, daß – genau wie in den ganzen Zahlen – das Kommutativ- und das Assoziativgesetz sowohl für die $\oplus$ als auch für $\odot$ gilt, und auch das Distributivgesetz folgt sofort aus dem für $\mathbb{Z}$. Bezüglich $\oplus$ ist die Null neutrales Element und $p - a$ invers zu a; bezüglich $\odot$ ist die Eins neutrales Element, und wie wir gerade gesehen haben, gibt es zu jedem $a \neq 0$ ein Inverses. Somit ist $\mathbb{F}_p$ ein Körper.

Das Rechnen in diesem Körper ist größtenteils einfach: Die Addition kann auf die gewöhnliche Addition in $\mathbb{Z}$ zurückgeführt werden; da $a + b$ für $a, b \in \mathbb{F}_p$ zwischen Null und $2p - 2$ liegt, ist

$$a \oplus b = \begin{cases} a+b & \text{falls } a+b < p \\ a+b-p & \text{sonst} \end{cases}\,.$$

Bei der Multiplikation ist die Situation nicht ganz so einfach; hier braucht man eine Division mit Rest, um $a \oplus b$ zu berechnen.

Das additive Inverse ist, wie bereits erwähnt, einfach $p - a$; die Berechnung des multiplikativen Inversen erfordert einen erweiterten EUKLIDischen Algorithmus und ist damit die rechenaufwendigste Operation in $\mathbb{F}_p$.

Bei der Kryptographie mit öffentlichen Schlüssel werden uns Körper $\mathbb{F}_p$ zu großen Primzahlen p interessieren; hier, für AES, brauchen wir nur den Fall $p = 2$, in dem natürlich alles trivial ist:

$\oplus$	0	1
0	0	1
1	1	0

$\odot$	0	1
0	0	0
1	0	1

c) Euklidische Ringe

Für den EUKLIDischen Algorithmus im ersten Abschnitt waren im wesentlichen zwei Dinge wesentlich: Ersten mußten wir wissen, wann eine Zahl Teiler einer anderen ist, und zweitens brauchten wir eine Divison mit Rest, für die der Rest in irgendweiner Weise kleiner als der Divisor ist, wobei jede Folge immer kleiner werdender Reste nach endlich vielen Schritten abbrechen muß. Diese beiden Eigenschaften werden im Begriff des EUKLIDischen Rings formalisiert:

Definition: *a)* Ein Ring ist eine Menge R zusammen mit zwei Verknüpfungen

$$+, \cdot : R \times R \to R\,,$$

für die gilt:
1.) $(R,+)$ ist eine abelsche Gruppe
2.) $a(bc) = (ab)c$ für alle $A, b, c \in R$
3.) $a(b+c) = ab + ac$ für alle $a, b, c \in R$.
b) R heißt *komutativer Ring mit Eins,* falls zusätzlich gilt
4.) Es gibt ein Element $1 \in R$, so daß $1 \cdot a = a \cdot 1 = a$ für alle $a \in R$
5.) $a \cdot b = b \cdot a$ für alle $a, b \in R$.
c) Ein kommutativer Ring mit Eins heißt *Integritätsbereich,* wenn gilt:
6.) Für $a, b \in R \setminus \{0\}$ ist auch $ab \neq 0$.
d) Ein EUKLIDischer Ring ist ein Integritätsbereich R zusammen mit einer Abbildung $\nu\colon R \setminus \{0\} \to \mathbb{N}_0$, für die gilt:
7.) Zu je zwei Elementen $a, b \in R \setminus \{0\}$ gibt es $q, r \in R$ mit

$$a = bq + r \quad \text{und} \quad \nu(r) < \nu(b) \ \text{ falls } r \neq 0\,.$$

Offensichtlich ist der Ring $\mathbb{Z}$ der ganzen Zahlen ein EUKLIDischer Ring; hier können wir einfach $\nu(a) = |a|$ setzen. Als Übung kann man auch leicht zeigen, daß der Ring

$$\Gamma = \mathbb{Z} \oplus \mathbb{Z}i = \bigl\{ a + bi \in \mathbb{C} \bigm| a, b \in \mathbb{Z} \bigr\}$$

der GAUSSschen Zahlen EUKLIDisch ist mit

$$\nu(a + ib) = a^2 + b^2\,.$$

Interessanter für uns ist, daß für jeden Körper k der Polynomring

$$k[X] = \Bigl\{ \sum_{\ell=0}^{n} a_\ell X^\ell \Bigm| n \in \mathbb{N}_0 \ \text{ und } \ a_i \in k \Bigr\}$$

ein EUKLIDischer Ring ist, wobei die Funktion ν hier einfach jedem Polynom seinen Grad zuordnet, also den Exponenten der höchsten vorkommenden X-Potenz. Hier zeigt die übliche Polynomdivison mit Rest, daß es in der Tat zu je zwei Polynomen f, g mit Koeffizienten aus k Polynome q, r gibt, so daß

$$f = gq + r \quad \text{und} \quad r = 0 \ \text{ oder } \ \deg r < \deg g\,.$$

Da EUKLIDische Ringe insbesondere Integritätsbereiche sind, können wir auch dort größte gemeinsame Teiler definieren:

Definition: *a)* Ein Element t eines Integritätsbereichs R heißt Teiler von $r \in R$, in Zeichen $t|r$, wenn es ein $q \in R$ gibt mit $r = qt$.
b) $d \in R$ heißt größter gemeinsamer Teiler von $r, s \in R$, wenn $d|r, s$ und wenn für jeden weiteren gemeinsamen Teiler t von r und s gilt: $t|d$.

Eine besondere Rolle spielen die Teiler der Eins: Für einen solchen Teiler t gibt es ein Element q mit $qt = 1$, d.h. t hat ein Inverses bezüglich der Multiplikation. Solche Elemente bezeichnen wir als *Einheiten;* offensichtlich bilden die Einheiten aus R eine multiplikative Gruppe, die sogenannte *Einheitengruppe* $R^\times$.

Man beachte, daß die so definierten größten gemeinsamen Teiler nicht eindeutig bestimmt sein müssen: In $\mathbb{Z}$ beispielsweise ist sowohl drei als auch minus drei ein größter gemeinsamer Teiler von sechs und minus neun. Allgemein gilt:

Lemma: In einem EUKLIDischen Ring R gibt es zu je zwei Elementen r, s, die nicht beide gleich null sind, einen größten gemeinsamen Teiler d. Dieser kann nach dem EUKLIDischen Algorithmus berechnet werden und läßt sich als Linearkombination

$$d = \alpha r + \beta s \quad \text{mit} \quad \alpha, \beta \in R$$

darstellen. Sind d und d' zwei größte gemeinsame Teiler von r und s, so gibt es eine Einheit e mit $d' = ed$.

Beweis: Die Existenz eines größten gemeinsamen Teilers folgt genau wie im Fall der natürlichen Zahlen: Ist $a = bq + r$, so ist ein Element $t \in R$ genau dann gemeinsamer Teiler von a und b, wenn es gemeinsamer Teiler von b und r ist. Daher gibt es genau dann einen größten gemeinsamen Teiler von a und b, wenn es einen größten gemeinsamen Teiler von b und r gibt, und dieser ist dann gleichzeitig größter gemeinsamer Teiler von a und b.

Der EUKLIDische Algorithmus kann genauso definiert werden wir im letzten Abschnitt, und genau wie dort muß er auch enden mit einem Divisionsrest null, denn die Folge der Zahlen $\nu(r_i) \in \mathbb{N}_0$ ist strikt fallend. Im Falle $r_n = 0$ ist $\mathrm{ggT}(r_{n-2}, r_{n-1}) = r_{n-1}$, da r_{n-1} dann r_{n-2} ohne Rest teilt; induktiv folgt, daß das auch ein ggT von a und b ist.

Sind d und d' zwei größte gemeinsame Teiler von a und b, so sind beide insbesondere gemeinsame Teiler von a und b; nach Definition eines größten gemeinsamen Teilers muß daher d Teiler von d' sein und umgekehrt. Es gibt somit ein $e \in R$ mit $d' = ed$ und ein $e' \in R$ mit $d = e'd'$. Also ist

$$d = e'd' = e'ed \Longrightarrow (1 - e'e)d = 0 \Longrightarrow 1 - e'e = 0 \Longrightarrow e'e = 1\,,$$

da R nach Voraussetzung ein Integritätsbereich ist. Die letzte Gleichung rechts zeigt, daß e eine Einheit ist.

Schließlich müssen wir noch zeigen, daß sich *jeder* größte gemeinsame Teiler d von a und b als Linearkombination von a und b schreiben läßt. Der erweiterte EUKLIDische Algorithmus liefert eine solche Darstellung für *einen* größten gemeinsamen Teiler

$$d' = \alpha a + \beta b\,;$$

sind e, e' die oben betrachtete Einheit zu d und d', so folgt

$$d = e'd' = (\alpha e')a + (\beta e')b\,,$$

wie behauptet.

■

Als Beispiel wollen wir für $k = \mathbb{Q}$ den größten gemeinsamen Teiler der beiden Polynome

$$P = X^8 + X^6 - 3X^4 - 3X^3 + 8X^2 + 2X - 5$$

und

$$Q = 3X^6 + 5X^4 - 4X^2 - 9X + 21$$

berechnen: Division von P durch Q führt auf den Quotienten $X^2/3 - 2/9$ und Divisionsrest

$$R_2 = -\frac{5}{9}X^4 + \frac{1}{9}X^2 - \frac{1}{3}\,.$$

Division von Q durch R_2 ergibt

$$R_3 = -\frac{117}{25}X^2 - 9X + \frac{441}{25}\,,$$

bei der Division von R_2 durch R_3 bleibt Rest

$$R_4 = \frac{233150}{6591}X - \frac{102500}{2197}\,,$$

und bei der letzten Divison verbleibt als Rest der ggT

$$R_5 = \frac{1288744821}{543589225}\,.$$

Da beide Ausgangspolynome ganzzahlige Koeffizienten haben, erscheint ein ggT mit einem so großen Nenner seltsam. Wir wissen aber, daß „der“ größte gemeinsame Teiler nur bis auf Einheiten bestimmt ist, und im Polynomring über einem Körper sind alle von null verschiedenen Konstanten Einheiten. Der größte gemeinsame Teiler ist daher nur indeutig bis auf Multiplikation mit einer nichtverschwindenen Konstanten; diese Konstante kann nach Belieben gewählt werden und wird meist so gewählt, daß das Ergebnis in irgendeinem Sinne einfach wird.

Auf das obige Beispiel angewendet heißt das, daß mit

$$R_5 = \frac{1288744821}{543589225}$$

auch eins ein ggT von A und B ist und man daher im allgemeinen sagen würde, „der“ ggT von A und B sei eins. Es ist ein wohlbekanntes (und umgehbares) Problem der Computeralgebra, daß der EUKLIDische Algorithmus diese einfache Lösung in einer so komplizierten Form liefert; da wir aber vor allem Polynome über endlichen Körpern benötigen, braucht uns das nicht weiter zu kümmern.

d) Endliche Körper von Primzahlpotenzordnung

In Abschnitt *b)* hatten wir die ganzen Zahlen modulo einer Primzahl p zu einem Körper gemacht; der einzige nichttriviale Schritt dabei war die Existenz des multiplikativen Inversen, die wir aus der linearen Kombinierbarkeit des ggT folgerten und daraus, daß der ggT einer Zahl mit einer Primzahl gleich eins ist, falls die Zahl kein Vielfaches der Primzahl ist.

Genauso wollen wir jetzt Körper definieren, indem wir Polynome über einem festen Körper k modulo einem vorgegebenen Polynom P betrachten: Für ein beliebiges Polynom A über k ist $A \bmod P$ gleich dem Rest bei der Division von A durch P.

Falls A kleineren Grad als P hat, ist natürlich einfach $A \bmod P = A$; zum konkreten Rechnen können wir daher ausgehen vom Vektorraum V

aller Polynome vom Grad höchstens d, wobei $d+1$ der Grad von P ist. Die Addition ist die gewöhnliche Addition von Polynomen, das Nullpolynom ist Neutralelement, und $-A$ ist invers zu A.

Das Produkt AB zweier Polynome $A, B \in V$ kann größeren Grad als d haben; wir setzen daher

$$A \odot B = AB \bmod P\,;$$

dies ist ein Polynom vom Grad höchstens d, und es ist klar, daß die so definierte Multiplikation kommutativ und assoziativ ist und das Distributivgesetz erfüllt. Das konstante Polynom 1 ist Neutralelement auch bezüglich dieser Multiplikation. Algebraisch gesehen identifizieren wir V also mit dem Faktorring $k[X]/(P)$.

Ein inverses Polynom zu A ist ein Polynom B, für das $A \odot B = 1$ ist, d.h.

$$AB = 1 + CP \qquad \text{oder} \qquad AB + CP = 1$$

für ein geeignetes Polynom C. Zu vorgegebenen Polynomen A und P gibt es solche Polynome B und C genau dann, wenn der ggT von A und P gleich eins ist; alsdann lassen sich B und C nach dem erweiterten EUKLIDischen Algorithmus berechnen.

Wenn wir möchten, daß jedes Polynom A, dessen Grad kleiner als $\deg P$ ist, ein Inverses hat, müssen wir sicherstellen, daß A und P immer teilerfremd sind; dies ist offensichtlich genau dann der Fall, wenn P keinen nichttrivialen Teiler hat, keinen Teiler also, dessen Grad größer als null und kleiner als $\deg P$ ist. Ein solches Polynom heißt *irreduzibel.*

Falls es ein irreduzibles Polynom P vom Grad n mit Koeffizienten aus k gibt, läßt sich der Vektorraum k^n also zu einem Körper machen, indem wir ein n-tupel $(a_0, \ldots, a_{n-1})$ mit dem Polynom

$$a_{n-1}X^{n-1} + a_{n-2}X^{n-2} + \cdots + a_1X + a_0$$

identifizieren und die Multiplikation als Multiplikation von Polynomen modulo P erklären.

Bekanntestes Beispiel ist $k = \mathbb{R}$: Für $n = 2$ gibt es irreduzible Polynome vom Grad n, beispielsweise das Polynom $P = X^2 + 1$. Da

$$\begin{aligned}(a_1X + a_0)(b_1X + b_0) &= a_1b_1X^2 + (a_0b_1 + a_1b_0)X + a_0b_0 \\ &\equiv (a_0b_1 + a_1b_0)X + (a_0b_0 - a_1b_1) \bmod X^2 + 1\end{aligned}$$

ist, folgt

$$(a_0, a_1) \odot (b_0, b_1) = (a_0b_0 - a_1b_1, a_0b_1 + a_1b_0)\,,$$

wir erhalten also den Körper der komplexen Zahlen. Weitere Beispiele über $\mathbb{R}$ gibt es nicht, denn ein irreduzibles reelles Polynom muß entweder Grad eins oder Grad zwei haben, und da jedes irreduzible quadratische Polynom zwei konjugiert komplexe Nullstellen hat, entstehen dabei immer die komplexen Zahlen – lediglich die Basis über $\mathbb{R}$ ändert sich.

Über endlichen Körpern ist die Situation etwas komplizierter: Hier gibt es für *jedes* n mindestens ein irreduzibles Polynom vom Grad n, allerdings gilt auch hier, daß zwei irreduzible Polynome desselben Grads auf den gleichen Körper führen. Eine kurze Beweisskizze ist unten angedeutet; einen vollständigen Beweis findet man in jedem Lehrbuch der Algebra.

Es gibt keinen einfachen Ausdruck für ein irreduzibles Polynom vom Grad n über einem vorgegebenen endlichen Körper k; in der Computeralgebra behilft man sich meist damit, daß man so lange zufällige Polynome vom Grad n erzeugt, bis man ein irreduzibles gefunden hat – ein Algorithmus, der sich in der Praxis als deutlich effizienter erweist als manch ein deterministischer Algorithmus zur Lösung von Standardproblemen.

Es gibt allerdings auch eine Alternative, die zumindest für kleine Körper durchaus anwendbar ist: Ist $k = \mathbb{F}_q$ ein endlicher Körper mit q Elementen, so ist $k \setminus \{0\}$ eine multiplikative Gruppe der Ordnung $q-1$. Da die Ordnung eines jeden Elements einer Gruppe Teiler der Gruppenordnung ist, folgt

$$x^{q-1} = 1 \quad \text{für alle } x \in \mathbb{F}_q \setminus \{0\}\,.$$

Dies gilt insbesondere für das Element $X \bmod P$, das $\mathbb{F}_q$ über $\mathbb{F}_p$ erzeugt, also ist P ein Teiler von $X^{q-1} - 1$. Die möglichen Polynome P

zur Erzeugung von $\mathbb{F}_{p^n}$ über $\mathbb{F}_p$ sind also genau die irreduziblen Teiler vom Grad n des Polynoms $X^{p^n} - 1$. Die Computeralgebra stellt gerade für Polynome über endlichen Körpern effiziente Faktorisierungsalgorithmen zur Verfügung, so daß man diese Teiler noch für recht große Werte von p^n relativ schnell berechnen kann.

Als weitere Konsequenz aus obiger Formel kommt man auch zu einer neuen Interpretation des Körpers mit $q = p^n$ Elementen: Da alle q-1 Elemente von $\mathbb{F}_q \setminus \{0\}$ Nullstellen des Polynoms $x^{q-1} - 1$ sind und dieses Polynom den Grad $q-1$ hat, handelt es sich hier um *alle* Nullstellen von $x^{q-1} - 1$; in der Sprache der Algebra ist $\mathbb{F}_q$ daher der Zerfällungskörper des Polynoms $x^{q-1} - 1$ über $\mathbb{F}_p$. Daraus folgt wegen der Eindeutigkeit des Zerfällungskörpers, daß alle Körper mit q Elementen isomorph sind.

e) Der Körper mit 256 Elementen

Für AES ist der Körper $\mathbb{F}_{256}$ von zentraler Bedeutung; da $256 = 2^8$ ist, handelt es sich hier um den Vektorraum $\mathbb{F}_2^8$. Die Addition ist sehr einfach: Da die Addition in $\mathbb{F}_2$ mit dem exklusiven Oder übereinstimmt, ist die Addition in $\mathbb{F}_{256}$ das bitweise exklusive Oder, eine Operation, die nicht nur in den gängingen CPUs, sondern auch in vielen Prozessoren für spezielle Anwendungen in der Signalverarbeitung als Grundoperation implementiert und somit sehr schnell ist.

Um eine Multiplikation zu definieren, brauchen wir ein irreduzibles Polynom vom Grad acht über $\mathbb{F}_2$. Da $\mathbb{F}_2$ ein sehr kleiner Körper und acht eine ziemlich kleine Zahl ist, hat zumindest ein Computer keinerlei Schwierigkeiten, alle diese Polynome zu bestimmen: Wie wir im vorigen Abschnitt gesehen haben, sind das genau die irreduziblen Faktoren vom Grad acht in der Faktorisierung des Polynoms $X^{255} - 1$ über $\mathbb{F}_2$. Faktorisierung von Polynomen über Körpern von Primzahlordung ist einer der Grundalgorithmen der Computeralgebra und geht auch noch bei sehr viel komplizierteren Polynomen sehr schnell; hier zeigt das Ergebnis, daß es dreißig Faktoren vom Grad acht gibt. Diese führen zwar alle auf denselben Körper, aber das praktische Rechnen in diesem Körper hängt natürlich stark von der Wahl des Polynoms ab. Insbesondere wird die Geschwindigkeit umso höher, je weniger Terme das Polynom hat.

Dreizehn der dreißig Polynome bestehen aus sieben nichtverschwindenden Termen, die restlichen siebzehn aus fünf; Wir wählen natürlich eines der letzteren. Alle diese Polynome haben, wie jedes Polynom vom Grad acht über $\mathbb{F}_2$, den führenden Term X^8; danach folgen vier weitere Terme. Bei der Reduktion modulo einem solchen Polynom $P = X^8 + \mathit{Rest}$ benutzt man, daß dann

$$X^8 \equiv \mathit{Rest}, \qquad X^9 \equiv X \cdot \mathit{Rest}, \quad \ldots$$

ist; dies wird umso häufiger mehrfach angewandt werden müssen, je höheren Grad die Terme in *Rest* haben. Am effizientesten kann man also rechnen, wenn das Polynom *Rest* den kleinstmöglichen Grad hat, und wenn zudem auch noch die hinteren Terme von *Rest* möglichst kleinen Grad haben. Inspektion der siebzehn Polynome mit fünf Termen zeigt, daß das Polynom

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

in dieser Hinsicht optimal ist.

Die genaue Festlegung für das Rechnen in $\mathbb{F}_{256} = \mathbb{F}_2^8$ für die Zwecke von AES ist folgende: Wir schreiben ein Byte als $(a_7, a_6, \ldots, a_0)$ und identifizieren es mit dem Polynom

$$a_7X^7 + a_6X^6 + \cdots + a_1X + a_0 ;$$

das Byte 0000 0010 entspricht also X.

Der Einfachheit halber schreiben wir Bytes meist als zweiziffrige Hexadezimalzahlen: Im betrachteten Beispiel wäre das 02_{hex}, und das Byte $\text{A5}_{\text{hex}} = 1010\,0101$ entspricht dem Polynom $X^7 + X^5 + X^2 + 1$.

Man beachte, daß trotz dieser Schreibweise die Addition und Multiplikation in $\mathbb{F}_{256}$ natürlich nichts mit der Addition und Multiplikation von Hexadezimalzahlen zu tun haben. Zwar ist $01_{\text{hex}} + 02_{\text{hex}} = 03_{\text{hex}}$, aber $01_{\text{hex}} + 01_{\text{hex}} = 00_{\text{hex}}$ und $05_{\text{hex}} + 04_{\text{hex}} = 01_{\text{hex}}$.

Zur Berechnung von $\text{A5}_{\text{hex}} \odot 01_{\text{hex}}$ müssen wir das Polynom

$$(X^7 + X^5 + X^2 + 1) \cdot X = X^8 + X^6 + X^3 + X$$

berechnen und modulo $m(X)$ reduzieren. Da

$$X^8 \bmod m(X) = X^4 + X^3 + X^2 + 1$$

ist (in $\mathbb{F}_2$ ist $-1 = 1$), ist dies

$$(X^4 + X^3 + X^2 + 1) + (X^6 + X^3 + X) = X^6 + X^4 + X^2 + X + 1\,.$$

Somit ist $\mathrm{A5}_{\mathrm{hex}} \odot 01_{\mathrm{hex}} = 56_{\mathrm{hex}}$.

Trotz der Wahl des optimalen Polynoms ist die Multiplikation als immer noch erheblich aufwendiger als die Addition.

§3: Spezifikation von Rijndael

a) Terminologie und Bezeichnungen

Rijndael arbeitet mit verschiedenen Blocklängen sowie auch verschiedenen Schlüssellängen: Beide können (unabhänging voneinander) alle durch 32 teilbaren Werte zwischen 128 und 256 annehmen. Wir schreiben die Blocklänge als $32N_b$ und die Schlüssellänge als $32N_k$; die Zahlen N_b und N_k liegen also jeweils zwischen vier und acht.

Der offizielle FIPS-Standard AES ist nicht wirklich *gleich* Rijndael; für AES ist nur die eine Blocklänge 128 normiert und nur die drei Schlüssellängen 128, 196 und 256; hier ist also stets $N_b = 4$ und N_k kann die drei Werte 4, 6, 8 annehmen.

Rijndael verschlüsselt somit einen Block von $32N_b$ Bit oder $4N_b$ Bytes, und genau wie DES geschieht dies sukzessive in mehreren Runden. In der Terminologie von Rijndael wird der Block in seinem jeweiligen Bearbeitungsstand als „Zustand“ bezeichnet; die einzelnen Runden finden also jeweils einen bestimmten Zustand vor und verändern diesen.

Die Anzahl der Runden wird als N_r bezeichnet; sie hängt von N_b und N_k ab gemäß der Gleichung

$$N_r = \max(N_b, N_k) + 6\,.$$

b) Die Grundoperationen

Auch wenn Rijndael nicht mehr nach dem Prinzip eines FEISTEL-Netzwerks arbeitet, sind in den einzelnen Runden immer noch die klassischen SHANNONschen Prinzipien von Konfusion und Diffusion realisiert.

Für die Konfusion sind bei DES die verschiedenen S-Boxen zuständig; bei Rijndael gibt es nur eine einzige Funktion zu diesem Zweck; diese ist definiert als Hintereinanderausführung der Bildung des (multiplikativen) Inversen in $\mathbb{F}_{256}$ mit einer über $\mathbb{F}_2$ affinen Abbildung von $\mathbb{F}_{256}$ nach $\mathbb{F}_{256}$.

Die Inversenbildung ist natürlich nur für $\mathbb{F}_{256} \setminus \{0\}$ erklärt; um trotzdem eine bijektive Abbildung von $\mathbb{F}_{256}$ nach $\mathbb{F}_{256}$ zu bekommen, nehmen wir die einzig mögliche Fortsetzung, bilden die Null also auf sich selbst ab. Auch wenn es algebraisch kompletter Unsinn ist, wollen wir zur Vermeidung von Fallunterscheidungen für die Definition von Rijndael die Kurzschreibweise $0^{-1} = 0$ verwenden mit der Interpretation, daß $x \mapsto x^{-1}$ die Fortsetzung der Inversenbildung zu einer bijektiven Abbildung von $\mathbb{F}_{256}$ nach $\mathbb{F}_{256}$ sein soll.

Diese Abbildung ist weder über $\mathbb{F}_{256}$ noch über $\mathbb{F}_2$ linear, und sie ist das einzige nichtlineare Element von Rijndael. Rechnerisch ist die Bestimmung von x^{-1} aufwendig, allerdings gibt es nur 256 mögliche Werte für x, die man vorausberechnen und in 256 Byte abspeichern kann; dieser Speicherbedarf dürfte selbst für Smartcards problemlos sein.

Diffusion wird durch eine Reihe von $\mathbb{F}_{256}$-linearen Abbildungen erzielt, die Operationen sind also im Gegensatz zu den Bit-Permutationen von DES allesamt auf Byte-Niveau definiert. Die Multiplikation von $\mathbb{F}_{256}$ sorgt dafür, daß trotzdem auch eine beträchtliche Diffusion innerhalb der Bytes stattfindet. Durch Tabellen gegebene Permutationen gibt es in AES überhaupt nicht; alle Diffusion wird durch Shift-Operationen sowie durch eine algebraische Operation realisiert.

Letztere arbeitet mit Wörtern von vier Byte, die wiederum mit Polynomen identifiziert werden, jetzt aber nicht, wie bei der Definition der Multiplikation in $\mathbb{F}_{256}$, mit Polynomen über $\mathbb{F}_2$, sondern solchen über $\mathbb{F}_{256}$. Wir schreiben die neuen Polynome daher zur besseren Unterscheidung in einer neuen Variablen Y und identifizieren $\mathbb{F}_{256}^4$ somit mit dem Vektorraum aller Polynome vom Grad höchstens drei in Y mit Koeffizienten aus $\mathbb{F}_{256}$, indem wir das Quadrupel (b_0, b_1, b_2, b_3) auffassen als Polynom

$$b_3Y^3 + b_2Y^2 + b_1Y + b_0 \quad \text{mit} \quad b_i \in \mathbb{F}_{256}\,.$$

Diese Polynome multiplizieren wir modulo dem Polynom $M = Y^4 + 1$.

Man beachte, daß dieses Polynom über $\mathbb{F}_2$ (und erst recht über $\mathbb{F}_{256}$) nicht irreduzibel ist: Da alle Binomialkoeffizienten außer dem ersten und dem letzten gerade sind, ist

$$(Y + 1)^4 = Y^4 + 1 \bmod 2 .$$

Mit der hier definierten Multiplikation wird $\mathbb{F}_{256}^4$ also nicht zu einem Körper. Rijndael verwendendet diese Multiplikation allerdings nur für eine einzige lineare Abbildung von $\mathbb{F}_{256}^4$ nach $\mathbb{F}_{256}^4$, und diese ist gegeben durch Multiplikation mit dem Polynom

$$C = 03_{\text{hex}} Y^3 + Y^2 + Y + 02_{\text{hex}} .$$

An der Stelle $Y = 1$ ist

$$C = 03_{\text{hex}} + 1 + 1 + 02_{\text{hex}} = 03_{\text{hex}} + 02_{\text{hex}} = 1 ,$$

das Polynom ist also nicht durch $Y + 1$ teilbar (zur Erinnerung: über $\mathbb{F}_2$ wie auch $\mathbb{F}_{256}$ ist $Y + 1 = Y - 1$), und damit auch teilerfremd zu $Y^4 + 1 = (Y + 1)^4$. Damit hat zumindest dieses Polynom ein multiplikatives Inverses, das man mit dem erweiterten EUKLIDischen Algorithmus über $\mathbb{F}_{256}$ berechnen kann – auch wenn das von Hand ziemlich aufwendig ist. Im Maple-*worksheet* zu diesem Paragraphen findet man die detailierte Rechnung mit Unterprogrammen für die Rechenoperationen von $\mathbb{F}_{256}$; als Ergebnis findet man dort das inverse Polynom

$$C' = 0\text{B}_{\text{hex}} Y^3 + 0\text{D}_{\text{hex}} Y^2 + 09_{\text{hex}} Y + 0\text{E}_{\text{hex}} .$$

Wer will, kann nachrechnen, daß $C \cdot C'$ modulo M gleich eins ist, allerdings erfordert bereits das recht viele Multiplikationen in $\mathbb{F}_{256}$.

c) Der Aufbau der Runden

Nach diesen Vorbereitungen können wir uns mit dem Aufbau der einzelnen Runden beschäftigen. Abgesehen von einer leichten Modifikation bei der letzten Runde besteht jede Runde aus denselben vier Schritten

1. Bytesubstitution
2. Zeilenshift
3. Spaltenmix
4. Addition des Rundenschlüssels

1.) Die Bytesubstitution: Dies ist der Konfusionsschritt; er operiert auf Bytes und wird auf jedes einzelne Byte des Zustands in derselben Weise angewendet; mit geeigneter Hardware kann dies also auch parallel erfolgen. Da es nur 256 mögliche Byte gibt, wird man diese Operation im allgemeinen über eine Tabelle implementieren; der Speicherbedarf von 256 Byte sollte auch auf einer Smartcard im allgemeinen problemlos sein,

Der erste Schritt ist, wie bereits erwähnt, die Inversenbildung im Körper $\mathbb{F}_{256}$; danach folgt eine Diffusion auf Bitniveau, indem $\mathbb{F}_{256}$ als affiner Raum $\mathbb{F}_2^8$ interpretiert wird und die affine Abbildung

$$\vec{x} \mapsto M\vec{x} + \vec{b}$$

mit

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \quad \text{und} \quad \vec{b} = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

angewandt wird. Insgesamt führt die Bytesubstitution ein Byte x also über in $Mx^{-1} + \vec{b}$.

Betrachtet man $\mathbb{F}_{256}$ nicht als affinen Raum, sondern als Raum der Polynome vom Grad höchstens sieben, kann man die affine Abbildung auch interpretieren als

$$P \mapsto (X^7 + X^6 + X^5 + X^4 + 1)P + (X^7 + X^6 + X^2 + X) \bmod X^8 + 1 ;$$

da $X^7 + X^6 + X^5 + X^4 + 1$ an der Stelle eins den Wert eins annimmt, ist dieses Polynom teilerfremd zu $X^8 + 1 = (X + 1)^8$, die Abbildung und damit die Matrix M sind also invertierbar – was sie natürlich auch sein müssen, damit die Chiffre entschlüsselbar ist.

Als dritte Alternative kann man die Abbildung auch durch ein Polynom über $\mathbb{F}_{256}$ beschreiben, denn da der Körper endlich ist findet sich zu *jeder*

Abbildung $\mathbb{F}_{256} \to \mathbb{F}_{256}$ ein Interpolationspolynom, das sie beschreibt. Im vorliegenden Fall ist dies das Polynom

$$\begin{aligned}63_{\text{hex}} + 05_{\text{hex}} Z \quad + 09_{\text{hex}} Z^2 \ + \text{F9}_{\text{hex}} Z^4 \ + 25_{\text{hex}} Z^8 \\ + \text{F4}_{\text{hex}} Z^{16} + 01_{\text{hex}} Z^{32} + \text{B5}_{\text{hex}} Z^{64} + \text{8F}_{\text{hex}} Z^{128} \,.\end{aligned}$$

Diese Abbildung wird angewandt auf das multiplikative Inverse eines Elements von $\mathbb{F}_{256}$. Im Körper $\mathbb{F}_{256}$ ist für jedes Element $z \neq 0$

$$z^{255} = 1 \quad \text{und} \quad (z^{-1})^n = z^{-n} = z^{255-n} \,,$$

wobei letztere Gleichung wegen unserer Konvention $0^{-1} = 0$ für *alle* z gilt. Damit können wir die Bytesubstitution insgesamt auch beschreiben durch das Polynom

$$\begin{aligned}63 + 05_{\text{hex}} Z^{254} + 09_{\text{hex}} Z^{253} + \text{F9}_{\text{hex}} Z^{251} + 25_{\text{hex}} Z^{247} \\ + \text{F4}_{\text{hex}} Z^{239} + 01_{\text{hex}} Z^{223} + \text{B5}_{\text{hex}} Z^{191} + \text{8F}_{\text{hex}} Z^{127} \,.\end{aligned}$$

Die Umkehrabbildung der Bytesubstitution hat als affine Abbildung die Form

$$\vec{y} \mapsto M^{-1}\vec{y} + M^{-1}\vec{b}\,;$$

dabei ist

$$M^{-1} = \begin{pmatrix} 0&0&1&0&0&1&0&1\\ 1&0&0&1&0&0&1&0\\ 0&1&0&0&1&0&0&1\\ 1&0&1&0&0&1&0&0\\ 0&1&0&1&0&0&1&0\\ 0&0&1&0&1&0&0&1\\ 1&0&0&1&0&1&0&0\\ 0&1&0&0&1&0&1&0 \end{pmatrix} \quad \text{und} \quad M^{-1}\vec{b} = \begin{pmatrix} 1\\0\\1\\0\\0\\0\\0\\0 \end{pmatrix} .$$

Die Inversenabbildung ist natürlich (auch mit der Konvention, daß die Null auf sich selbst abgebildet wird) zu sich selbst invers; die Bytesubstitution wird also rückgängig gemacht, indem man ein Byte $\vec{y}$ zunächst auf $A^{-1}\vec{y} + A^{-1}\vec{b}$ abbildet und dann die erweiterte Inversenabbildung von $\mathbb{F}_{256}$ anwendet.

2.) Die Zeilenshifts: Der Diffusionsschritt ist zweigeteilt: Der zu verschlüsselnde Block ist ein Vektor aus $4N_b$ Bytes; er wird umgeschrieben in eine Byte-Matrix mit vier Zeilen und N_b-Spalten, die spaltenweise aufgefüllt wird. Wird die Matrix als $A = (a_{ij})$ bezeichnet, wobei der Zeilenindex i von 0 bis 3 und der Spaltenindex j von 0 bis $N_b - 1$ geht, ist der ursprüngliche Vektor also (in Zeilenschreibweise)

$$(a_{00}, a_{10}, a_{20}, a_{30}, a_{01}, a_{11}, \ldots, a_{N_b-2,3}, a_{N_b-1,0}, \ldots, a_{N_b-1,3})\,.$$

Indiziert man die Komponenten des Vektors durch einen Index n linear von 0 bis $4N_b - 1$, ist also

$$i = n \bmod 4, \quad j = \left\lfloor \frac{n}{4} \right\rfloor \quad \text{und} \quad n = i + 4j\,.$$

Der erste Diffusionsschritt ist eine zyklische Verschiebung der Zeilen von A: Die 0-te Zeile wird überhaupt nicht verschoben und die erste um eine Stelle; die Richtung der Verschiebung ist, wie auch stets im folgenden, nach links. Bei den beiden unteren Reihen hängt die Weite der Verschiebung von N_b ab: Für $N_b \neq 8$ wird die zweite Zeile um zwei Stellen verschoben, für $N_b = 8$ und drei. Die dritte Zeile wird für $N_b \leq 6$ um drei Stellen, für $N_b \leq 7$ um drei Stellen verschoben. Mit entsprechender Hardware können die Verschiebungen der einzelnen Zeilen natürlich parallel durchgeführt werden.

3.) Der Spaltenmix: Auf Spaltenebene ist die Diffusion aufwendiger: Ein Spaltenvektor ist ein Element von $\mathbb{F}_{256}^4$. Diesen Vektorraum hatten wir oben mit den kubischen Polynomen über $\mathbb{F}_{256}$ identifiziert und dort eine Multiplikation eingeführt über die Polynommultiplikation modulo Y^4+1. Genau dies verwendet der Spaltenmix, indem jeder Spaltenvektor mit dem Polynom

$$C = 03_{\text{hex}} Y^3 + Y^2 + Y + 02_{\text{hex}}$$

multipliziert wird. Wegen der einfachen Struktur des Polynoms $Y^4 + 1$ ist dies eine sehr einfache lineare Abbildung mit Matrix

$$\begin{pmatrix} 02_{\text{hex}} & 03_{\text{hex}} & 01_{\text{hex}} & 01_{\text{hex}} \\ 01_{\text{hex}} & 02_{\text{hex}} & 03_{\text{hex}} & 01_{\text{hex}} \\ 01_{\text{hex}} & 01_{\text{hex}} & 02_{\text{hex}} & 03_{\text{hex}} \\ 03_{\text{hex}} & 01_{\text{hex}} & 01_{\text{hex}} & 02_{\text{hex}} \end{pmatrix}$$

bezüglich der Basis $\{1, Y, Y^2, Y^3\}$. Diese kann für die N_b Spalten parallel durchgeführt werden.

Im Vergleich zum Zeilenshift ist diese Operation relativ aufwendig, da mehrere Multiplikationen in $\mathbb{F}_{256}$ durchgeführt werden müssen. In der letzten Runde, in der dieser Diffusionschritt keiner anschließenden Konfusion mehr unterworfen wird und somit keinen großen Sicherheitsgewinn mehr bietet, wird daher auf diesen Schritt verzichtet.

Wie wir bereits gesehen haben, ist C modulo $Y^4 + 1$ invertierbar mit inversem Polynom

$$C' = 0\mathrm{B}_{\mathrm{hex}}Y^3 + 0\mathrm{D}_{\mathrm{hex}}Y^2 + 09_{\mathrm{hex}}Y + 0\mathrm{E}_{\mathrm{hex}} ;$$

Multiplikation mit C' ist als lineare Abbildung gegeben durch die Matrix

$$\begin{pmatrix} 0\mathrm{E}_{\mathrm{hex}} & 0\mathrm{B}_{\mathrm{hex}} & 0\mathrm{D}_{\mathrm{hex}} & 09_{\mathrm{hex}} \\ 09_{\mathrm{hex}} & 0\mathrm{E}_{\mathrm{hex}} & 0\mathrm{B}_{\mathrm{hex}} & 0\mathrm{D}_{\mathrm{hex}} \\ 0\mathrm{D}_{\mathrm{hex}} & 09_{\mathrm{hex}} & 0\mathrm{E}_{\mathrm{hex}} & 0\mathrm{B}_{\mathrm{hex}} \\ 0\mathrm{B}_{\mathrm{hex}} & 0\mathrm{D}_{\mathrm{hex}} & 09_{\mathrm{hex}} & 0\mathrm{E}_{\mathrm{hex}} \end{pmatrix} ;$$

als Übung für das Rechnen in $\mathbb{F}_{256}$ sollte man zumindest für einige Einträge nachrechnen, daß das Produkt dieser beiden Matrizen über $\mathbb{F}_{256}$ gleich der Einheitsmatrix ist.

4.) Schlüsselexpansion und Rundenschlüssel: Die bisherigen drei Schritte sorgten für Konfusion und Diffusion; sie sind aber für jeden, der das grundsätzliche Verfahren kennt, leicht rückgängig zu machen. Das ist nicht weiter verwunderlich, denn bislang haben wir ja den *Schlüssel* noch nicht ins Spiel gebracht, an dem bei einem normierten Standard wie AES die ganze Sicherheit des Verfahrens hängt.

Der vierte Schritt jeder Runde ist genau wie bei DES eine einfache Addition des Rundenschlüssels; die Addition ist dabei die gewöhnliche Vektoraddition in $\mathbb{F}_2^{32N_n}$, also das logische XOR. Sicherheitsrelevant ist, genau wie bei DES, die Berechnung der Rundenschlüssel aus dem vorgegebenen Schlüssel; der Schlüssel mit seinen $4N_k$ Bytes muß so aufgebläht werden auf $N_r \times 4N_b$ Schlüsselbytes, daß ein Kryptanalytiker aus einem oder auch einigen irgendwie ermittelten Rundenschlüsseln nicht auf den Gesamtschlüssel schließen kann.

Rijndael verwendet dazu im wesentlichen dieselben Techniken wie bei den Verschlüsselungsschritten: Das erweiterte Schlüsselfeld wird aufgefaßt als eine Folge von Wörtern W_i mit einer Länge von vier Byte oder 32 Bit. Die ersten Wörter W_0 bis W_{N_k-1} sind der eigentliche Schlüssel des Verfahrens, der Rest wird rekursiv daraus berechnet. Da die ersten N_b Worte nicht als Rundenschlüssel verwendet werden, sondern vor der ersten Runde zum Klartext addiert werden, hat das gesamte Schlüsselfeld eine Länge von $N_b(N_r + 1)$ Worten.

Für $i \geq N_k$ wird W_i sukkzessive aus seinem unmittelbaren Vorgänger W_{i-1} und dem N_k Wörter zurückliegenden Vorgänger W_{i-N_k} berechnet:

- Falls i nicht durch N_k teilbar ist und im Falle $N_k > 6$ auch nicht durch vier, ist
  $$W_i = W_{i-N_k} \oplus W_{i-1} ,$$
  wobei $\oplus$ die Vektorraumaddition in $\mathbb{F}_2^{32} = \mathbb{F}_{256}^8$ bezeichnet, also das logische XOR für 32-Bit-Wörter.
- Falls i durch N_k teilbar ist, wird W_{i-1} zunächst zyklisch um eins nach links verschoben. Dann wird auf jedes Byte des so entstandenen Worts die Bytesubstitution aus Absatz *1)* angewendet, und das Ergebnis wird mit $\oplus$ zu W_{i-N_k} addiert. Dazu wird noch eine Rundenkonstante addiert, deren erstes Byte dasjenige Element von $\mathbb{F}_{256}$ ist, das der Potenz X^{i/N_k} modulo dem die Multiplikation definierenden Polynom entspricht und dessen weitere drei Bytes alle Null sind.
- Falls $N_k > 6$ und $i \equiv 4 \bmod N_k$, wird die Bytesubstitution auf W_{i-1} angewendet und das Ergebnis zu W_{i-N_k} addiert.

d) Gesamtablauf von Rijndael

Nachdem nun alle Einzelheiten spezifiziert sind, kann der Gesamtablauf von Rijndael leicht angegeben werden:

1. Die ersten N_b Worte des Schlüsselfelds werden zum Klartext addiert.
2. $N_r - 1$ Runden werden gemäß obiger Beschreibung durchgeführt.
3. Die N_r-te Runde wird entsprechend ausgeführt, aber ohne den Spaltenmix.

e) Geschwindigkeitsoptimierung

So wie Rijndael spezifiziert ist, ist vor allem die Bytesubstitution sehr langsam; aber wie bereits erwähnt, läßt sie sich leicht über eine Tabelle implementieren.

Ein weiterer rechnerisch aufwendiger Bestandteil von Rijndael sind die Multiplikationen im Körper $\mathbb{F}_{256}$. Eine vorausberechnete Multiplikationstabelle würde $256 \times 256 = 64 \times 1024$ Byte oder 64 Kilobyte in Anspruch nehmen, was erstens etwas viel und zweitens auch noch optimal ist: Die Multiplikation von $\mathbb{F}_{256}$ wird nur beim Spaltenmix benötigt, und da das hierzu verwendete Polynom C nur $01_{\text{hex}}, 02_{\text{hex}}$ und 03_{hex} als Koeffizienten hat, reicht es für die Verschlüsselung, wenn man die Produkte mit 02_{hex} und 03_{hex} bilden kann. Der jeweils andere Faktor des Produkts ist stets das Ergebnis einer Bytesubstitution, man spart also Rechenzeit, wenn man für jedes Byte das Ergebnis der Bytesubstitution sowie dessen Produkt mit 02_{hex} und mit 03_{hex} abspeichert; mit 768 Byte für Tabellen kann man also zur Laufzeit auf alle Multiplikationen und Divisionen in $\mathbb{F}_{256}$ verzichten.

Mit vier Kilobyte Tabellenplatz läßt sich sogar die gesamte Rundentransformation auf $4N_b$ XOR-Operationen auf 32-Bit-Wörtern zurückführen: Man speichere für jedes Byte $a \in \mathbb{F}_{256}$ die vier 32-Bit-Wörter

$$T_0(a) = \begin{pmatrix} 02_{\text{hex}} \odot S(a) \\ S(a) \\ S(a) \\ 03_{\text{hex}} \odot S(a) \end{pmatrix}, \quad T_1(a) = \begin{pmatrix} 02_{\text{hex}} \odot S(a) \\ 03_{\text{hex}} \odot S(a) \\ S(a) \\ S(a) \end{pmatrix},$$

$$T_2(a) = \begin{pmatrix} S(a) \\ 03_{\text{hex}} \odot S(a) \\ 02_{\text{hex}} \odot S(a) \\ S(a) \end{pmatrix} \quad \text{und} \quad T_3(a) = \begin{pmatrix} S(a) \\ S(a) \\ 03_{\text{hex}} \odot S(a) \\ 02_{\text{hex}} \odot S(a) \end{pmatrix},$$

wobei $S\colon \mathbb{F}_{256} \to F_{256}$ die Bytesubstitution ist.

Ist dann (bei Matrizenschreibweise) $\vec{b}_j$ der j-te Spaltenvektor des Zustands zu Beginn einer Runde, $\vec{e}_j$ der entsprechende Vektor am Ende der Runde und $\vec{k}_j$ der j-te Spaltenvektor des Rundenschlüssels, so ist

$$\vec{e}_j = T_0(b_{0j}) \oplus T_1(b_{1,j\ominus 1}) \oplus T_2(b_{2,j\ominus c_2}) \oplus T_3(b_{3,j\ominus c_3}) \oplus \vec{k}_j\ ;$$

dabei stehen c_2 und c_3 die Beträge, um die die zweite und dritte Zeile verschoben werden, d.h.

$$c_2 = \begin{cases} 2 & \text{für } N_b < 8 \\ 3 & \text{für } N_b = 8 \end{cases} \quad \text{und} \quad c_3 = \begin{cases} 2 & \text{für } N_b < 7 \\ 3 & \text{für } N_b \geq 7 \end{cases},$$

und $\ominus$ steht für die Subtraktion modulo N_b, wie man sie für die Zeilenshifts braucht. Man beachte, daß auch diese Rechnung bei entsprechender Hardware für alle N_b Spalten parallel ausgeführt werden kann.

Da sich auch die sämtlichen Rundenschlüssel mit einem Speicheraufwand von weniger als einem halben Kilobyte vorausberechnen lassen, läßt sich die Verschlüsselung mittels Rijndael also auch auf einem Standard-PC sehr schnell durchführen.

Bei anderen Implementierungen, bei denen es Probleme mit dem Speicherplatz gibt, kann man auf Kosten einer höheren Rechenzeit mit sehr viel weniger Speicher auskommen. Eine relativ billige Art und Weise, wie man bei 32-Bit-Prozessoren drei Kilobyte einsparen kann, folgt beispielsweise aus der Beobachtung, daß die verschiedenen $T_i(a)$ durch zyklische Verschiebung auseinander entstehen. Somit reicht es, die $T_0(a)$ abzuspeichern, und indem man analog zum HORNER-Schema rechnet, liegt der zusätzliche Rechenaufwand nur bei drei zyklischen Vertauschungen pro Spalte und pro Runde: Ist $\mathcal{Z}$ die zyklische Linksverschiebung um ein Byte, so ist

$$\vec{e}_j = \vec{k}_j \oplus T_0(b_{0j}) \oplus \mathcal{Z}\Big(T_0(b_{1,j-1}) \oplus \mathcal{Z}\big(T_0(b_{2,j-c_2}) \oplus \mathcal{Z}(T_0(b_{3,j-c_3}))\big)\Big)\ .$$

Für die Entschlüsselung braucht man natürlich entsprechende Tabellen; hier werden die Produkte mit $0\mathrm{B}_{\text{hex}}, 0\mathrm{D}_{\text{hex}}, 09_{\text{hex}}$ und $0\mathrm{E}_{\text{hex}}$ benötigt.

Das Arbeiten mit Tabellen kann übrigens unter Sicherheitsaspekten vorteilhaft sein gegenüber direktem Rechnen: Ein Gegner, der den Stromverbrauch der Rechnung kontinuierlich messen kann (z.B. weil eine Smartcard ihren Strom aus seinem Lesegerät bezieht), kann daraus eventuell Rückschlüsse auf Klartext und/oder Schlüssel ziehen, da etwa eine Ziffer eins in einer Multiplikation mehr Aufwand erfordert als eine Ziffer null. Bei Multiplikation via Tabellen ist der Stromverbrauch jedoch unabhängig von den Faktoren, da stets nur ein Tabellenwert gelesen und weiterverwendet wird.

§4: Angriffe auf Rijndael

Wie jede andere Blockchiffre bietet auch Rijndael keinerlei Sicherheit gegen einen BAYESschen Gegner – zumindest dann nicht, wenn man (wie praktisch immer) redundante Information verschlüsselt. Da allerdings bereits die einfachste Variante von Rijndael mit einer Schlüssellänge von 128 arbeitet, ist das Durchprobieren aller Schlüssel für real existierende Gegner mit heutiger Technologie unrealistisch: Gegenüber DES mit seiner Schlüssellänge 56 steigt der Aufwand immerhin um den Faktor

$$2^{128-56} = 2^{72} = 4\,722\,366\,482\,869\,645\,213\,696\,,$$

also um mehr als zwanzig Größenordnungen. Ein Gegner muß daher, um erfolgreich zu sein, Methoden finden, die mit deutlich geringerem Aufwand auskommen. Da er in der Wahl seiner Methoden frei ist, können wir nie wirklich wissen, wie er arbeitet; alle Sicherheitsaussagen beruhen nur darauf, daß keine realistische Attacke *bekannt* ist, obwohl im Verlauf des Begutachtungsprozesses international führende Experten mehrere Jahre lang danach gesucht haben. Natürlich geht die Suche auch jetzt noch weiter, und in der Tat sind inzwischen auch neue Ansätze aufgetaucht, die bei der Wahl von Rijndael noch nicht bekannt waren.

Natürlich ist das Design von Rijndael so gewählt, daß der Algorithmus resistent ist gegen differentielle und lineare Kryptanalyse; beides war schließlich zum Zeitpunkt seines Entwurfs wohlbekannt und gut verstanden. Auch ist die grundsätzliche Struktur von Rijndael nicht neu: Es ist zwar kein FEISTEL-Netzwerk mehr, aber er kommt aus einer Familie von Kryptoverfahren um den Algorithmus Square, der bereits seit einiger Zeit kryptanalysiert worden war. Insbesondere hatten DAEMEN und RIJMEN die sogenannte Square attack entwickelt, die mit speziell gewählten Klartexten den letzten Rundenschlüssel angreift: Verschlüsselt werden 256 Blöcke, die in allen Bytes mit einer Ausnahme übereinstimmen; das variable Byte nimmt alle 256 möglichen Werte an.

Verfolgt man diese Blöcke geschickt durch den Algorithmus, läßt sich mit hinreichend vielen Gruppen solcher Blöcke auch ein auf sechs Runden reduzierter Rijndael kryptanalysieren; da dies den beiden Designern bewußt war, ist die Rundenzahl entsprechend größer gewählt.

Ein Kritikpunkt an Rijndael war seine relativ einfache algebraische Struktur, die zwar die Implementierung erleichtert und beschleunigt, aber eventuell auch Sicherheitsprobleme aufwerfen könnte.

Im Mai 2001 gelang es NIELS FERGUSON, RICHARD SCHROEPPEL und DOUG WHITING, die allesamt in der Wirtschaft (bei verschiedenen Unternehmen) über Sicherheitsfragen arbeiten, Rijndael in einer geschlossenen Formel darzustellen; für die 128 Bit Version hat diese Formel $2^{50} \approx 10^{15}$ Terme, für 256 Bit sind es $2^{70} \approx 10^{21}$. Obwohl die Formel natürlich hoch strukturiert ist, ist allerdings völlig unklar, ob dieser Ansatz je zu einer Bedrohung für Rijndael werden kann; schließlich ist eine Formel nur selten die beste Möglichkeit für den Umgang mit einer Funktion. Die Originalarbeit ist zu finden unter `www.xs4all.nl/ vorpal/pubs/rdalgeq.html` .

Potentiell gefährlicher ist ein Ansatz von NICOLAS COURTOIS und JOSEF PIEPRZYK, deren XSL-Attacke (`eprint.iacr.org/2002/044/`) das Knacken von Rijndael übersetzt in die Lösung eines überbestimmten nichtlinearen Gleichungssystems. Ob dieser Ansatz langfristig zu einem Angriff führt, der schneller ist als das Durchprobieren aller Schlüssel, ist im Augenblick noch nicht abzuschätzen.

§5: Literatur

Die beste und ausführlichste Quelle ist das Buch der beiden Autoren von Rijndael:

JOAN DAEMEN, VINCENT RIJMEN: *The Design of Rijndael: AES – the Advanced Encryption Standard,* Springer 2002

Kürzere Darstellungen findet man in neueren Lehrbüchern der Kryptologie, z.B.

DOUGLAS R. STINSON: CRYPTOGRAPHY – THEORY AND PRACTICE, Chapman & Hall/CRC, ^{2}2002

oder

RICHARD A. MOLLIN: AN INTRODUCTION TO CRYPTOGRAPHY, Chapman & Hall/CRC, 2001

Kapitel 5
Das RSA-Verfahren

§ 1: New directions in cryptography

Bei allen bisher betrachteten Codes verläuft die Entschlüsselung entweder genauso oder zumindest sehr ähnlich wie die Verschlüsselung; insbesondere kann jeder, der eine Nachricht verschlüsseln kann, jede andere entsprechend verschlüsselte Nachricht auch entschlüsseln. Man bezeichnet diese Verfahren daher als *symmetrisch.*

Der Nachteil eines symmetrischen Verfahrens besteht darin, daß in einem Netzwerk jeder Teilnehmer mit jedem anderen einen Schlüssel vereinbaren muß. In militärischen Netzen war dies traditionellerweise so geregelt, daß das gesamte Netz denselben Schlüssel benutzte, der in einem Codebuch für jeden Tag im voraus festgelegt war; in kommerziellen Netzen wie beispielsweise einem Mobilfunknetz ist dies natürlich unmöglich.

1976 publizierten MARTIN HELLMAN, damals Assistenzprofessor in Stanford, und sein Forschungsassistent WHITFIELD DIFFIE eine Arbeit mit dem Titel *New directions in cryptography* (IEEE Trans. Inform. Theory **22**, 644–654), in der sie vorschlugen, den Vorgang der *Ver*schlüsselung und den der *Ent*schlüsselung völlig voneinander zu trennen: Es sei schließlich nicht notwendig, daß der Sender einer verschlüsselten Nachricht auch in der Lage sei, diese zu *ent*schlüsseln.

Der Vorteil eines solchen Verfahrens wäre, daß jeder potentielle Empfänger nur einen einzigen Schlüssel bräuchte und dennoch sicher sein könnte, daß nur er selbst seine Post entschlüsseln kann. Der Schlüssel

müßte nicht einmal geheimgehalten werden, da es ja nicht schadet, wenn jedermann Nachrichten *ver*schlüsseln kann. In einem Netzwerk mit n Teilnehmern bräuchte man also nur n Schlüssel, um es jedem Teilnehmer zu erlauben, mit jeden anderen so zu kommunizieren, und diese Schlüssel könnten sogar in einem öffentlichen Verzeichnis stehen. Bei einem symmetrischen Kryptosystem wäre der gleiche Zweck nur erreichbar mit $\frac{1}{2}n(n-1)$ Schlüsseln, die zudem noch durch ein sicheres Verfahren wie etwa ein persönliches Treffen oder durch vertrauenswürdige Boten ausgetauscht werden müßten.

DIFFIE und HELLMAN machten nur sehr vage Andeutungen, wie so ein System mit öffentlichen Schritten aussehen könnte. Es ist zunächst einmal klar, daß ein solches System keinerlei Sicherheit gegen einen BAYESschen Gegner bieten kann, denn die Verschlüsselungsfunktion ist eine bijektive Abbildung zwischen endlichen Mengen, und jeder, der die Funktion kennt, kann zumindest im Prinzip auch ihre Umkehrfunktion berechnen.

Wer im Gegensatz zum BAYESschen Gegner nur über begrenzte Ressourcen verfügt, kann diese Berechnung allerdings möglicherweise nicht mit realistischem Aufwand durchführen, und nur darauf beruht die Sicherheit eines Kryptosystems mit öffentlichen Schlüsseln. DIFFIE und HELLMAN bezeichnen eine Funktion, deren Umkehrfunktion nicht mit vertretbarem Aufwand berechnet werden kann, als *Einwegfunktion* und schlagen als Verschlüsselungsfunktion eine solche Einwegfunktion vor.

Damit hat man aber noch kein praktikables Kryptosystem, denn bei einer echten Einwegfunktion ist es auch für den legitimen Enpfänger nicht möglich, seinen Posteingang zu entschlüsseln. DIFFIE und HELLMAN schlagen deshalb eine Einwegfunktion mit *Falltür* vor, wobei der legitime Empfänger zusätzlich zu seinem öffentlichen Schlüssel noch über einen geheimen Schlüssel verfügt, mit dem er (und nur er) diese Falltür öffnen kann.

Natürlich hängt alles davon ab, ob es solche Einwegfunktionen mit Falltür wirklich gibt. DIFFIE und HELLMAN gaben keine an, und es gab unter den Experten einige Skepsis bezüglich der Möglichkeit, solche Funktionen zu finden.

Tatsächlich aber gab es damals bereits Systeme, die auf solchen Funktionen beruhten, auch wenn sie nicht in der offenen Literatur dokumentiert waren: Die britische *Communications-Electronics Security Group* (CESG) hatte bereits Ende der sechziger Jahre damit begonnen, nach entsprechenden Verfahren zu suchen, um die Probleme des Militärs mit dem Schlüsselmanagement zu lösen, aufbauend auf (impraktikablen) Ansätzen von AT&T zur Sprachverschlüsselung während des zweiten Weltkriegs. Die Briten sprachen nicht von Kryptographie mit öffentlichen Schlüsseln, sondern von *nichtgeheimer Verschlüsselung,* aber das Prinzip war das gleiche.

Erste Ideen dazu sind in einer auf Januar 1970 datierten Arbeit von JAMES H. ELLIS zu finden, ein praktikables System in einer auf den 20. November 1973 datierten Arbeit von CLIFF C. COCKS. Wie im Milieu üblich, gelangte nichts über diese Arbeiten an die Öffentlichkeit; erst 1997 veröffentlichten die *Goverment Communications Headquarters* (GCHQ), zu denen CESG gehört, einige Arbeiten aus der damaligen Zeit unter `http://www.cesg.gov.uk/about/nsecret/`.

Im akademischen Bereich gab es ein Jahr nach Erscheinen der Arbeit von DIFFIE und HELLMAN das erste Kryptosystem mit öffentlichen Schlüsseln: Drei Professoren am Massachussetts Institute of Technology fanden nach rund vierzig erfolglosen Ansätzen 1977 schließlich jenes System, das heute nach ihren Anfangsbuchstaben mit RSA bezeichnet wird: RON RIVEST, ADI SHAMIR und LEN ADLEMAN.

Das System wurde 1983 von der eigens dafür gegründeten Firma RSA Computer Security Inc. patentiert und mit großem kommerziellem Erfolg vermarktet. Das Patent lief zwar im September 2000 aus, die Firma ist aber weiterhin erfolgreich im Kryptobereich tätig; sie hatte beispielsweise auch einen Kandidaten für AES entwickelt, der es immerhin bis in die Endrunde schaffte.

RSA ist übrigens identisch mit dem von COCKS vorgeschlagenen System, so daß einige Historiker auch Zweifel an den Behauptungen der GCHQ haben. Die Beschreibung durch RIVEST, SHAMIR und ADLEMAN erschien 1978 unter dem Titel *A method for obtaining digital signatures and public-key cryptosystems* in Comm. ACM **21**, 120–126.

§2: Algebraische Vorbereitungen

Das RSA-Verfahren hängt ab von Multiplikationen modulo einer großen Zahl N, die Produkt zweier Primzahlen p und q ist; wir wollen uns daher zunächst etwas mit dem Rechnen modulo einer solchen Zahl N beschäftigen.

Wir wissen bereits, daß die ganzen Zahlen modulo einer Primzahl p einen Körper bilden, den Körper $\mathbb{F}_p$ mit p Elementen. Die für das RSA-Verfahren wichtigste Eigenschaft des Rechnens modulo p ist der folgende Satz:

Kleiner Satz von Fermat: Für jedes $a \in \mathbb{Z}$ ist

$$a^p \equiv a \bmod p\,;$$

ist a nicht durch p teilbar, ist auch

$$a^{p-1} \equiv 1 \bmod p\,.$$

Beweis: Für zwei Zahlen $x, y \in \mathbb{Z}$ ist

$$(x+y)^p = \binom{p}{0}x^p + \binom{p}{1}x^{p-1}y + \cdots + \binom{p}{p-1}xy^{p-1} + \binom{p}{p}y^p$$

mit

$$\binom{p}{j} = \frac{p!}{j!(p-j)!}\,.$$

Für $j = 0$ und $j = p$ ist $\binom{p}{0} = \binom{p}{p} = 1$, ansonsten sind j und $p-j$ beide kleiner als p. Im Nenner steht daher dann kein Faktor p, der das p aus dem Zähler wegkürzen könnte, so daß alle $\binom{p}{j}$ mit $1 \le j \le p-1$ durch p teilbar sind, und

$$(x+y)^p \equiv x^p + y^p \bmod p\,.$$

Induktiv folgt, daß eine entsprechende Gleichung auch für Summen mit mehr als zwei Summanden gilt, insbesondere gilt also für beliebig viele Summanden eins, daß

$$(1+\cdots+1)^p \equiv 1^p + \cdots + 1^p = 1 + \cdots + 1 \bmod p\,.$$

Damit ist $a^p \equiv a \bmod p$ für jede natürliches Zahl a. Für $a = 0$ gilt die Beziehung auch, und da für jedes positive a

$$0 = \big(a + (-a)\big)^p = a^p + (-a)^p = a + (-a)^p \bmod p$$

ist, gilt sie auch für negative a.

Falls a nicht durch p teilbar ist, gibt es, da $\mathbb{F}_p$ ein Körper ist, ein b mit $ba \equiv 1 \bmod p$; mit diesem b ist

$$a^{p-1} \equiv (ba)a^{p-1} = ba^p \equiv ba \equiv 1 \bmod p\,.$$

■

Korollar: Falls die natürliche Zahl e keinen Teiler mit $p-1$ gemeinsam hat, gibt es eine natürliche Zahl d, so daß

$$\big(a^e\big)^d \equiv a \bmod p$$

für alle $a \in \mathbb{Z}$.

Beweis: Nach dem erweiterten EUKLIDischen Algorithmus läßt sich der größte gemeinsame Teiler eins von e und $p-1$ als ganzzahlige Linearkombination dieser beiden Zahlen schreiben, es gibt also ganze Zahlen d und r, so daß

$$d \cdot e + r \cdot (p-1) = 1$$

ist. Hier könnte d eventuell noch negativ sein, aber indem wir nötigenfalls noch ein Vielfaches der Gleichung

$$(p-1) \cdot e - e \cdot (p-1) = 0$$

dazuaddieren, können wir annehmen, daß d positiv ist (und r dann natürlich negativ). Für nicht durch p teilbares a ist dann

$$\big(a^e\big)^d = a^{de} = a^{1-r(p-1)} = a \cdot \big(a^{p-1}\big)^{-r} \equiv a \cdot 1^{-r} = a \bmod p\,,$$

wie behauptet.

Für durch p teilbares a sind beide Seiten der zu beweisenden Kongruenz durch p teilbar, so daß sie auch in diesem Fall richtig ist.

■

Für RSA interessiert, wie bereits erwähnt, vor allem das Rechnen modulo des Produkts zweier Primzahlen; hier gilt entsprechend

Satz: Ist $N = pq$ Produkt zweier verschiedener Primzahlen p und q, so gilt
a) Für jede ganze Zahl a ist $a^{1+(p-1)(q-1)} \equiv a \bmod N$.
b) Falls die natürliche Zahl e keinen Teiler mit $(p-1)(q-1)$ gemeinsam hat, gibt es eine natürliche Zahl d, so daß

$$\big(a^e\big)^d \equiv a \bmod p$$

für alle $a \in \mathbb{Z}$.
c) Die Berechnung von $(p-1)(q-1)$ aus N ist äquivalent zur Faktorisierung von N.

Beweis: a) Nach dem kleinen Satz von FERMAT ist für jedes nicht durch p teilbare a und jedes r

$$a^{1+r(p-1)} \equiv a \bmod p\,.$$

Für Vielfache von p sind beide Seiten durch p teilbar, so daß die Gleichung tatsächlich für alle ganzen Zahlen a gilt.

Entsprechendes gilt natürlich auch für die Primzahl q, und damit ist für jede ganze Zahl s

$$a^{1+s(p-1)(q-1)} \equiv a \bmod p \quad \text{und} \quad a^{1+(p-1)(q-1)} \equiv a \bmod q\,.$$

Diese beiden Kongruenzen besagen, daß die Differenz zwischen linker und rechter Seite sowohl durch p als auch durch q teilbar ist, also auch durch deren Produkt N, d.h.

$$a^{1+s(p-1)(q-1)} \equiv a \bmod N\,.$$

b) Geht wegen *a)* genau wie beim Primzahlfall im obigen Korollar.

c) Falls die Faktorisierung $N = pq$ bekannt ist, läßt sich natürlich leicht $(p-1)(q-1)$ berechnen. Ist umgekehrt

$$(p-1)(q-1) = pq - p - q + 1 = N + 1 - (p+q)$$

bekannt, so kennt man außer dem Produkt N auch die Summe M der beiden Primfaktoren, und diese sind die leicht berechenbaren Lösungen der quadratischen Gleichung $x^2 - Mx + N = 0$.

■

§3: Das RSA-Verfahren zur Verschlüsselung und für elektronische Unterschriften

Zur praktischen Durchführung des RSA-Verfahrens wählt man sich zwei verschiedene Primzahlen p, q, die unbedingt geheim gehalten werden müssen, und eine natürliche Zahl e, die keinen gemeinsamen Teiler mit $(p-1)(q-1)$ hat. Dann veröffentlicht man die beiden Zahlen $N = pq$ und e als öffentlichen Schlüssel.

Sodann berechnet man zu $(p-1)(q-1)$ gemäß obigem Satz nach dem EUKLIDischen Algorithmus eine Zahl d, so daß

$$\left(a^e\right)^d \equiv a \bmod p$$

für alle a; diese Zahl ist der geheime Schlüssel.

a) Verschlüsselung

Jeder, der den öffentlichen Schlüssel (N, e) kennt, kann Nachrichten verschlüsseln: Er bricht die Nachricht auf in Blöcke, die durch ganze Zahlen zwischen null und $N-1$ dargestellt werden können, berechnet für jeden so dargestellten Block den Chiffretext $b \equiv a^e \bmod N$, der als Zahl zwischen null und $N-1$ interpretiert und an den Inhaber des geheimen Schlüssels geschickt wird. (Wir schreiben in solchen Fällen in Zukunft kurz $b = a^e \bmod N$, wobei „mod“ in diesem Zusammenhang als die Berechnung des Divisionsrests bei Division durch N zu interpretieren ist.)

Der Empfänger berechnet $b^d \bmod N$; da

$$b^d \equiv a^{ed} \equiv a \bmod N$$

ist, entschlüsselt dies die Nachricht.

b) Identitätsnachweis

Im Gegensatz zu symmetrischen Kryptoverfahren endet die Nützlichkeit des RSA-Verfahrens nicht mit der bloßen Möglichkeit einer Verschlüsselung; das Verfahren kann beispielsweise auch benutzt werden, um in Zugangskontrollsystemen, vor Geldautomaten oder bei einer Bestellung im Internet die Identität einer Person Zu beweisen: Nur der Inhaber des geheimen Schlüssels d kann zu einem gegebenen a eine Zahl b berechnen, für die $b^e \equiv a \bmod N$ ist.

Falls also der jeweilige Gegenüber eine Zufallszahl a erzeugt und als Antwort das zugehörige b verlangt, kann er anhand eines öffentlichen Schlüsselverzeichnisses die Richtigkeit von b überprüfen und sich so von der Identität seines Partners überzeugen. Im Gegensatz zu Kreditkarteninformation oder Paßwörtern ist dieses Verfahren auch immun gegen Abhören: Falls jedesmal ein neues zufälliges a erzeugt wird, nützt ein einmal abgehörtes b nichts.

Trotzdem ist das Verfahren in dieser Form nicht als Ersatz zur Übertragung von Kreditkarteninformation oder ähnlichem geeignet, da der Gegenüber anhand des öffentlichen Schlüssels jederzeit zu einer willkürlich gewählten Zahl b die Zahl $a = b^e \bmod N$ erzeugen kann um dann zu behaupten, er habe b als Antwort darauf empfangen. Man müßte also beispielsweise noch zusätzlich verlangen, daß die Zahl a eine spezielle Form hat, etwa daß die vordere Hälte der Ziffernfolge identisch mit der hinteren Hälfte ist.

c) Elektronische Unterschriften

Praktische Bedeutung hat vor allem eine weitere Variante: die elektronische Unterschrift. Her geht es darum, daß der Empfänger erstens davon überzeugt wird, daß eine Nachricht tatsächlich vom behaupteten Absender stammt, und daß er dies zweitens auch einem Dritten gegenüber beweisen kann. (In Deutschland sind solche elektronischen Unterschriften, sofern gewisse formale Voraussetzungen erfüllt sind, rechtsverbindlich.)

Um einen Nachrichtenblock a mit $0 \leq a < N$ zu unterschreiben, berechnet der Inhaber des öffentlichen Schlüsseln (N, e) mit seinem geheimen Schlüssel d die Zahl

$$b = a^d \bmod N$$

und sendet das Paar (a, b) an den Empfänger. Dieser überprüft, ob

$$b^e \equiv a \bmod N\,;$$

falls ja, akzeptiert er dies als unterschriebene Nachricht a. Da er ohne Kenntnis des geheimen Schlüssels d nicht in der Lage ist, den Block (a, b) zu erzeugen, kann er auch gegenüber einem Dritten beweisen, daß der Absender die Nachricht a unterschrieben hat.

Für kurze Nachrichten ist dieses Verfahren in der vorgestellten Form praktikabel; in vielen Fällen kann man sogar auf die Übermittlung von a verzichten, da $b^e \bmod N$ für ein falsch berechnetes b mit an Sicherheit grenzender Wahrscheinlichkeit keine sinnvolle Nachricht ergibt.

Falls die übermittelte Nachricht geheimgehalten werden soll, müssen a und b natürlich noch vor der Übertragung mit dem öffentlichen Schlüssel des Empfängers oder nach irgendeinem anderen Kryptoverfahren verschlüsselt werden.

Bei langen Nachrichten ist die Verdoppelung der Nachrichtenlänge nicht mehr akzeptabel, und selbst, wenn man auf die Übertragung von a verzichten kann, ist das Unterschreiben jedes einzelnen Blocks sehr aufwendig. Deshalb unterschreibt man meist nicht die Nachricht selbst, sondern einen daraus extrahierten Hashwert. Dieser Wert muß natürlich erstens von der gesamten Nachricht abhängen, und zweitens muß es für den Empfänger (praktisch) unmöglich sein, zwei Nachrichten zu erzeugen, die zum gleichen Hashwert führen. Mit der Theorie (und Praxis) dieser sogenannten kollisionsfreien Hashfunktionen werden wir uns später beschäftigen.

d) Blinde Unterschriften und elektronisches Bargeld

Einer der erfolgversprechendsten Ansätze zum Aushebeln eines Kryptosystems besteht darin, sich auf die Dummheit seiner Mitmenschen zu verlassen.

So sollte es durch gutes Zureden nicht schwer sein, jemanden zu Demonstrationszwecken zum Unterschreiben einer sinnlosen Nachricht zu bewegen: Eine Folge von Nullen und Einsen ohne sinnvolle Interpretation hat schließlich keine rechtliche Wirkung.

Nun muß eine sinnlose Nachricht aber nicht unbedingt eine Zufallszahl sein: Sie kann sorgfältig präpariert sein. Sei dazu etwa m eine Nachricht,

die ein Zahlungsversprechen enthält, (N, e) der öffentliche Schlüssel des Opfers und r eine Zufallszahl zwischen 2 und $N - 2$. Dann wird

$$x = m \cdot r^e \bmod N$$

wie eine Zufallsfolge aussehen, für die man eine Unterschrift

$$u = x^d \bmod N = (mr^e)^d \bmod N = m^d r \bmod N$$

bekommt. Multiplikation mit r^{-1} macht daraus eine Unterschrift unter die Zahlungsverpflichtung m.

Das angegebene Verfahren kann nicht nur von Trickbetrügern benutzt werden; blinde Unterschriften sind auch die Grundlage von *digitalem Bargeld.*

Zahlungen im Internet erfolgen meist über Kreditkarten; die Kreditkartengesellschaften haben also einen recht guten Überblick über die Ausgaben ihrer Kunden und machen teilweise auch recht gute Geschäfte mit Kundenprofilen.

Digitales Bargeld will die Anonymität von Geldscheinen mit elektronischer Übertragbarkeit kombinieren und so ein anonymes Zahlungssystem z.B. für das Internet bieten.

Es wir ausgegeben von einer Bank, die für jede angebotene Stückelung einen öffentlichen Schlüssel (N, e) bekanntgibt. Eine Banknote ist eine mit dem zugehörigen geheimen Schlüssel unterschriebene Seriennummer.

Die Seriennummer kann natürlich nicht einfach *jede* Zahl sein; sonst wäre jede Zahl kleiner N eine Banknote. Andererseit dürfen die Seriennummern aber auch nicht von der Bank vergeben werden, denn sonst wüßte diese, welcher Kunde Scheine mit welchem Seriennummern hat. Als Ausweg wählt man Seriennummern einer sehr speziellen Form: Ist $N > 10^{150}$, kann man etwa als Seriennummer eine 150-stellige Zahl wählen, deren Ziffern spiegelsymmetrisch zur Mitte sind, d.h. ab der 76. Ziffer werden die vorherigen Ziffern rückwärts wiederholt. Die Wahrscheinlichkeit, daß eine zufällige Zahl x nach Anwendung des öffentlichen Exponenten auf so eine Zahl führt, ist 10^{-75} und damit vernachlässigbar.

Seriennummern werden von den Kunden zufällig erzeugt. Für jede solche Seriennummer m erzeugt der Kunde eine Zufallszahl r, schickt mr^e mod N an die Bank und erhält (nach Belastung seines Kontos) eine Unterschrift u für diese Nachricht zurück. Wie oben berechnet er daraus durch Multiplikation mit r^{-1} die Unterschrift $v = m^d$ mod N für die Seriennummer N, und mit diesem Block kann er bezahlen.

Der Zahlungsempfänger berechnet v^e mod N; falls dies die Form einer gültigen Seriennummer hat, kann er sicher sein, einen von der Bank unterschriebenen Geldschein vor sich zu haben. Er kann allerdings noch nicht sicher sein, daß dieser Geldschein nicht schon einmal ausgegeben wurde.

Deshalb muß er die Seriennummer an die Bank melden, die mit ihrer Datenbank bereits ausbezahlter Seriennummern vergleicht. Falls sie darin noch nicht vorkommt, wird sie eingetragen und der Händler bekommt sein Geld; andernfalls verweigert sie die Zahlung.

Bei 10^{75} möglichen Nummern liegt die Wahrscheinlichkeit dafür, daß zwei Kunden, die eine (wirklich) zufällige Zahl wählen, dieselbe Nummer erzeugen, bei etwa $10^{-37,5}$. Die Wahrscheinlichkeit, mit jeweils einem Spielschein fünf Wochen lang hintereinander sechs Richtige im Lotto zu haben, liegt dagegen bei $\binom{49}{6}^{-5} \approx 5 \cdot 10^{-35}$, also etwa um den Faktor sechzig höher. Zwei gleiche Seriennummern sind also praktisch auszuschließen, wenn auch theoretisch möglich.

Das System kann nur funktionieren, wenn in diesem Fall der zweite Geldschein mit derselben Seriennummer nicht anerkannt wird, so daß der zweite Kunde sein Geld verliert. Dies muß als eine zusätzliche Gebühr gesehen werden, die mit an Sicherheit grenzender Wahrscheinlichkeit nie fällig wird.

Da digitales Bargeld überdies nur in kleinen Stückelungen sinnvoll ist (Geldscheinen im Millionenwert wären auf Grund ihrer Seltenheit nicht wirklich anonym und würden, wegen der damit verbundenen Möglichkeiten zur Geldwäsche auch in keinem seriösen Wirtschaftssystem angeboten), wäre der theoretisch mögliche Verlust auch nicht sehr groß.

§4: Wie findet man große Primzahlen?

a) Wie groß sollten die Primzahlen sein?

Das RSA-Verfahren ist nur sicher, wenn die Primzahlen p und q hinreichend groß gewählt werden. Als erstes müssen wir uns also die Frage stellen, was „hinreichend groß“ bedeuten soll.

Ein treu sorgender Staat bleibt seinen Bürgern auf eine so wichtige Frage natürlich keine Antwort schuldig: Zwar gibt es noch keine oberste Bundesbehörde für Primzahlen, aber das Bundesamt für Sicherheit in der Informationstechnik (BSI) und die Regulierungsbehörde für Telekommunikation und Post erarbeiten jedes Jahr ein gemeinsames Dokument mit dem Titel *Geeignete Kryptoalgorithmen gemäß §17 (2) SigV.*

SigV steht für die aufgrund des Signaturgesetztes SigG erlassene Signaturverordnung; beide gemeinsam legen fest, daß elektronische Unterschriften in Deutschland grundsätzlich zulässig und rechtsgültig sind, sofern gewisse Bedingungen erfüllt sind. Zu diesen Bedingungen gehört unter anderem, daß das Verfahren und die Schlüssellänge gemeinsam einen „geeigneten Kryptoalgorihmus“ im Sinne der jeweils gültigen Veröffentlichung der Regulierungsbehörde ist.

Da Rechner immer schneller und leistungsfähiger werden und auch auf der theoretisch-algorithmischen Seite fast jedes Jahr kleinere oder größere Fortschritte zu verzeichnen sind, gelten die jeweiligen Empfehlungen nur für etwa sechs Jahre. Offiziell geht es dabei jeweils nur um die Empfehlung geeigneter Algorithmen für elektronische Unterschriften und deren Schlüssellängen, aber wie die Entwicklung der letzten Jahre zeigte, drehen sich die Diskussionen, die zu den jeweiligen Empfehlungen führen, tatsächlich fast ausschließlich um die jeweils notwendige Schlüssellänge für RSA.

Natürlich hat in einer Demokratie bei so einer wichtigen Frage auch die Bevölkerung ein Mitspracherecht; deshalb beginnt das BSI jeweils zunächst einen Entwurf, zu dem es um Kommentare bittet; erst einige Monate später wird die endgültige Empfehlung verkündet und im Bundesanzeiger veröffentlicht.

Die interessierte Öffentlichkeit, von der die Kommentare zu den Entwürfen kommen, besteht naturgemäß in erster Linie aus Anbietern von Kryptographie-Software, und als erfahrene Experten für Datensicherheit wissen diese, daß ein Verfahren nur dann wirklich geeignet sein kann, wenn es die eigene Firma im Angebot hat. (Am geeignetsten sind natürlich die Verfahren, die keines der Konkurrenzunternehmen anbietet.)

Derzeit erhältliche Hardware-Implementierungen von RSA unterstützen typischerweise Schlüssellängen von bis zu 1024 Bit; größere Schlüssel sind vor allem in *public domain* Software wie PGP zu finden. Dies erklärt, warum es in den letzten Jahren recht lebhafte Diskussionen gab:

Bis Ende 2000 galten 768 Bit als ausreichende Größe für das Produkt N der beiden Primzahlen, aber die Richtlinien für 2000 legten fest, daß danach bis Mitte 2005 eine Mindestgröße von 1024 Bit erforderlich sei, danach bis Ende 2005 sogar 2048 Bit.

Anbieterproteste führten dazu, daß nach den Richtlinien von 2001 eine Schlüssellänge von 1024 dann doch noch bis Ende 2006 sicher war; die Schlüssellänge 2048 war nur noch „empfohlen“, also nicht mehr verbindlich.

Im April 2002 erschien der erste Entwurf für die 2002-Richtlinien; darin war für 2006 und 2007 nur eine Mindestlänge von 2048 Bit wirklich sicher. Einsprüche führten im September 2002 zu einem revidierten Entwurf, wonach 2006 doch noch 1024 Bit reichen, 2007 aber mindestens 1536 notwendig werden. Die Mindestlänge von 2048 Bit wurde wieder zur „Empfehlung“ zurückgestuft.

Am 2. Januar 2003 erschienen endlich die offiziellen Richtlinien des Jahres 2002; danach reichen 1024 Bit auch noch bis Ende 2007, erst 2008 werden 1280 Bit erforderlich. Die 2048 Bit bleiben dringend empohlen, und in wenigen Monaten wird wohl der erste Entwurf der Richtlinien für 2003 erscheinen …

Die beiden Primfaktoren p, q sollen zufällig und unabhängig voneinander erzeugt werden und aus einem Bereich stammen, in dem

$$\varepsilon_1 < |\log_2 p - \log_2 q| < \varepsilon_2$$

gilt. Als *Anhaltspunkte* werden dabei die Werte

$$\varepsilon_1 = 0{,}5 \quad \text{und} \quad \varepsilon_2 = 30$$

vorgeschlagen; ist p die kleinere der beiden Primzahlen, soll also

$$\sqrt{2}p < q < 2^{30}p \approx 10^9 p$$

sein.

Wir sollten also, wenn wir wirklich sicher sein wollen, in der Lage sein, Primzahlen mit etwa 1024 Bit oder 309 Dezimalstellen zu finden.

Im Land der unbegrenzten Möglichkeiten ist das kein großes Problem: Dort kann man Primzahlen, wie alles andere auch, einfach kaufen. Unter Sicherheitsgesichtspunkten ist das allerdings nicht unbedingt die beste Strategie, denn erstens kann dann der Verkäufer der Primzahlen die gesamte verschlüsselte Korrespondenz des Käufers lesen und auch dessen elektronische Unterschrift nachmachen, und zweitens wird man nie ganz sicher sein können, ob Billiganbieter wie *Thrifty Primes* oder *Primes for a Buck* nicht gelegentlich dieselbe Primzahl an mehrere Kunden verkaufen, so daß ein Kunde die öffentlichen Schlüssel seiner Konkurrenz durch die gerade gekauften Primzahlen teilen kann und damit teilweise Erfolg hat.

Sowohl aus Sicherheitsgründen als auch weil wir Mathematiker sind, sollten wir also unsere Primzahlen selbst erzeugen. Die größte derzeit bekannte Primzahl, gefunden am 14.November 2001, ist $2^{13\,466\,917} - 1$ mit 4 053 946 Dezimalstellen; da etwa 300 Dezimalstellen (entsprechend 1024 Bit) verglichen damit recht wenig ist, sollte die Suche nach solchen Primzahlen eigentlich nicht sonderlich schwierig sein.

Zu bedenken ist allerdings, daß Rekordprimzahlen allesamt eine sehr spezielle Struktur haben; 13 466 917 etwa ist eine Primzahl, so daß $2^{13\,466\,917} - 1$ die Form $2^p - 1$ hat; solche Zahlen bezeichnet man als MERSENNEsche Primzahlen. Bislang sind 39 solcher Zahlen bekannt, und schon daraus folgt, daß diese Zahlen für die Kryptographie völlig nutzlos sind: 39 Möglichkeiten kann schließlich jeder durchprobieren.

Unsere Primzahlen müssen also zufällig und allgemein sein, da sonst ein Gegner einfach alle Primzahlen einer speziellen Form durchprobieren kann. Idealerweise sollte jede Primzahl der richtigen Größenordnung die gleiche Chance haben, daß sie gewählt wird. Das läßt sich am sichersten dadurch erreichen, daß wir Zufallszahlen erzeugen und diese Testen, ob sie Primzahlen sind; wir brauchen daher einen Test, der uns sagt, ob eine gegebene Zahl prim ist.

b) Der Fermat-Test

Der kleine Satz von FERMAT gibt uns sofort eine Aussage darüber, wann eine Zahl p *nicht* prim ist:

Falls für eine natürliche Zahl $1 \leq a \leq p-1$ *gilt* $a^{p-1} \not\equiv 1 \bmod p$, *kann* p *keine Primzahl sein.*

Beispiel: Ist $F_{20} = 2^{2^{20}} + 1$ eine Primzahl? Falls ja, ist nach dem kleinen Satz von FERMAT insbesondere

$$3^{F_{20}-1} = 1 \bmod F_{20} .$$

Nachrechnen zeigt, daß

$$3^{(F_{20}-1)/2} \not\equiv \pm 1 \bmod F_{20} ,$$

die Zahl ist also nicht prim. (Das „Nachrechnen“ ist bei dieser 315653-stelligen Zahl natürlich keine Übungsaufgabe für Taschenrechner: 1988 brauchte eine Cray X-MP dazu 82 Stunden, eine Cray-2 immerhin noch zehn; siehe *Math. Comp.* **50** (1988), 261–263. Die anscheinend etwas weltabgewandt lebenden Autoren meinen, das sei die teuerste bislang produzierte 1-Bit-Information.)

Die Umkehrung der obigen Aussage gilt leider nicht: Ist beispielsweise

$$p = (6t+1)(12t+1)(18t+1)$$

Produkt dreier *Primzahlen* dieser speziellen Form, so ist nach FERMAT

$$a^{6t} = 1 \bmod (6t+1), \qquad a^{12t} = 1 \bmod (12t+1)$$

und

$$a^{18t} = 1 \bmod (18t+1) ,$$

also

$$a^{36t} = 1 \bmod p .$$

Wegen

$$p - 1 = 36t(36t^2 + 11t + 1)$$

ist dann auch

$$a^{p-1} = 1 \bmod p$$

für alle zu p teilerfremden ganzen Zahlen a.

Zahlen dieser Art gibt es tatsächlich, z.B.

$$1729 = 7 \times 13 \times 19, \qquad 294409 = 37 \times 73 \times 109 \quad usw. ;$$

es gibt auch entsprechende Zahlen, die nicht von dieser speziellen Form sind: Wir benötigen offenbar nur, daß für jeden Primteiler q von p auch $q-1$ ein Teiler von $p-1$ ist. Es ist nicht bekannt, ob es unendlich viele Zahlen mit dieser Eigenschaft gibt, es gilt aber als wahrscheinlich.

Trotzdem wird es für große Zahlen zunehmend unwahrscheinlich, daß eine Zahl p für auch nur ein a den obigen Test besteht, ohne Primzahl zu sein. Rechnungen von

SU HEE KIM, CARL POMERANCE: The probability that a Random Probable Prime is Composite, *Math. Comp.* **53** (1989), 721–741

geben folgende obere Schranke für die Fehlerwahrscheinlichkeit ε:

$p \approx 10^{60}$	10^{70}	10^{80}	10^{90}	10^{100}
$\varepsilon \leq 7{,}16 \cdot 10^{-2}$	$2{,}87 \cdot 10^{-3}$	$8{,}46 \cdot 10^{-5}$	$1{,}70 \cdot 10^{-6}$	$2{,}77 \cdot 10^{-8}$
$p \approx 10^{120}$	10^{140}	10^{160}	10^{180}	10^{200}
$\varepsilon \leq 5{,}28 \cdot 10^{-12}$	$1{,}08 \cdot 10^{-15}$	$1{,}81 \cdot 10^{-19}$	$2{,}76 \cdot 10^{-23}$	$3{,}85 \cdot 10^{-27}$

(Sie geben natürlich auch eine allgemeine Formel an, jedoch ist diese zu grausam zum Abtippen.)

Selbst wenn wir noch mit 512-Bit-Moduln arbeiten und somit knapp achtzigstellige Primzahlen brauchen, liegt also die Fehlerwahrscheinlichkeit bei nur etwa 10^{-5}; falls man sie erniedrigen möchte, testet man einfach mit mehreren zufällig gewählten Basen und hat dann etwa bei drei verschiedenen Basen eine Irrtumswahrscheinlichkeit von höchsten etwa 10^{-15}, daß alle drei Tests das falsche Ergebnis liefern.

Bei den etwa 155-stelligen Zahlen, die wir für 1024-Bit-Moduln brauchen, hat schon ein einziger Test eine geringere Fehlerwahrscheinlichkeit als 10^{-15}, so daß es sich nur selten lohnt, einen größeren Aufwand zu treiben. Die Regulierungsbehörde empfiehlt allerdings, bei probabilistischen Primzahltests eine Irrtumswahrscheinlichkeit von höchsten $2^{-80} \approx 8{,}27 \cdot 10^{-25}$ zuzulassen; diese Schranke wird erst bei knapp zweihundertstelligen Primzahlen mit einem einzigen FERMAT-Test erreicht. Immerhin, wenn sich die 1280 Bit Mindestlänge von RSA-Moduln durchsetzen sollte, braucht man etwas mehr als 190-stellige Primzahlen, und damit dürfte man in diesem Bereich sein.

Einige Leute reden bei Zahlen, die einen FERMAT-Test bestanden haben, von „wahrscheinlichen Primzahlen“. Das ist natürlich Unsinn: Eine Zahl ist entweder *sicher* prim oder *sicher* zusammengesetzt; für Wahrscheinlichkeiten gibt es hier keinen Spielraum. Besser ist der Ausdruck „industrial grade primes“, also „Industrieprimzahlen“, der ausdrücken soll, daß wir als Mathematiker zwar nicht entscheiden können, ob die Zahl wirklich prim ist, daß sie aber für industrielle Anwendungen gut genug ist.

Natürlich kennt die Mathematik auch Verfahren, um exakt festzustellen, ob eine Zahl prim ist oder nicht; das einfachste besteht darin, alle potentiellen Primteiler einfach auszuprobieren. Wir wollen uns im folgenden überlegen, wie man den FERMAT-Test zu einem solchen Verfahren ausbauen kann. Dazu brauchen wir allerdings zunächst einige weitere algebraische Vorbereitungen:

c) Die multiplikative Gruppe eines endlichen Körpers

Im Körper mit q Elementen erfüllt jedes Element $x \neq 0$ die Gleichung $x^{q-1} = 1$, ist also eine $(q-1)$-te Einheitswurzel. Wenn wir Einheitswurzeln in den *komplexen Zahlen* statt in einem endlichen Körper betrachten, können wir die Lösungen dieser Gleichung leicht angeben: Es sind genau die Zahlen

$$z_\nu = e^{\frac{2\pi i\nu}{q-1}} \quad \text{mit} \quad k = 1, 2, \ldots, q-1 \, .$$

Jede dieser Zahlen läßt sich beispielsweise als Potenz der ersten schreiben, denn

$$e^{\frac{2\pi i\nu}{q-1}} = \left(e^{\frac{2\pi i}{q-1}}\right)^\nu \, .$$

Gruppen, in denen sich wie hier jedes Element als Potenz eines festen Elements schreiben läßt, bezeichnen wir als *zyklisch:*

Definition: *a)* Die Gruppe G heißt *zyklisch,* wenn es ein Element $g \in \mathbb{Z}$ gibt, so daß sich jedes andere Element als Potenz von g schreiben läßt.
b) Die *Ordnung* eines Elements $g \in G$ einer Gruppe G ist die kleinste natürliche Zahl n, für die g^n gleich dem Neutralelement von G ist. Falls es keine solche Zahl n gibt, sagen wir, g habe unendliche Ordnung.

Offensichtlich ist eine endliche Gruppe der Ordnung n genau dann zyklisch, wenn sie (mindestens) ein Element der Ordnung n enthält.

So ist beispielsweise die additiven Gruppen Z/n zyklisch, da das Element 1 die Ordnung n hat. (Man beachte, daß wir in dieser additiv geschriebenen Gruppe an Stelle von Potenzen Vielfache betrachten müssen.) Es ist allerdings im allgemeinen nicht das einzige: In $\mathbb{Z}/3$ beispielsweise hat auch 2 diese Eigenschaft, den $1 \cdot 2 = 2, 2 \cdot 2 = 1$ und $3 \cdot 2 = 0$ sind die sämtlichen Elemente von $\mathbb{Z}/3$. Dagegen hat das Element 2 von $\mathbb{Z}/6$ nur die Ordnung drei; als Erzeugende können wir hier nur die Elemente 1 und 5 nehmen. Allgemein überlegt man sich leicht, daß das Element $r \in \mathbb{Z}/n$ die Ordnung $n/\operatorname{ggT}(n,r) = \operatorname{kgV}(n,r)$ hat; r ist also genau dann ein erzeugendes Element, wenn r als ganze Zahl betrachtet teilerfremd zu n ist.

Die Ordnung r eines Elements g einer endlichen Gruppe ist stets ein Teiler der Gruppenordnung n, denn wegen $x^n = x^r = 1$ und der linearen ombinierbarkeit des ggT ist auch $x^{\operatorname{ggT}(n,r)} = 1$, und die Ordnung ist der *kleinste* Exponent mit dieser Eigenschaft. Da $ggT(n,r) \leq r$ ist, muß also hier das Gleichheitszeichen gelten, d.h. $r|n$.

Das (bis auf Isomorphie) einzige Beispiel einer unendlichen zyklischen Gruppe ist die Gruppe $\mathbb{Z}$ der ganzen Zahlen; als erzeugendes Element können wir +1 oder -1 wählen. Andere Elemente kommen nicht in Frage, obwohl außer der Null jedes Element unendliche Ordnung hat.

Für gewisse Primzahltests und später auch für Kryptosystemen auf der Basis diskreter Logarithmen wichtig ist der folgende

Satz: Die multiplikative Gruppe eines endlichen Körpers ist zyklisch.

Beweis: Wir wissen bereits, daß jedes Element außer Null eines Körpers mit q Elementen der Gleichung $x^{q-1} = 1$ genügt; die Ordnung eines jeden Elements ist also ein Teiler von $q-1$. Nach obiger Bemerkung folgt die Behauptung des Satzes, wenn es ein Element gibt, dessen Ordnung *genau* $q-1$ ist.

Für jeden Primteiler p_i von $q-1$ hat die Polynomgleichung

$$x^{\frac{q-1}{p_i}} = 1$$

höchstens $(q-1)/p_i$ Lösungen im Körper; es gibt also zu jedem p_i ein Körperelement a_i mit

$$a_i^{\frac{q-1}{p_i}} \neq 1\,.$$

Sei q_i die größte Potenz von p_i, die $q-1$ teilt, und

$$g_i = a_i^{\frac{q}{q_i}}$$

die $(q-1)/q_i$-te Potenz von a_i. Dann ist

$$g_i^{q_i} = a_i^{q-1} = 1 \quad \text{und} \quad g_i^{\frac{q_i}{p_i}} = a_i^{\frac{q-1}{p_i}} \neq 1\,;$$

g_i hat also die Ordnung q_i. Da die verschiedenen q_i Potenzen verschiedener Primzahlen p_i sind, hat daher das Produkt g aller g_i somit das Produkt aller q_i als Ordnung, also $q-1$. Damit ist die multiplikative Gruppe des Körpers zyklisch.

■

Der obige Beweis ist nicht konstruktiv, und in der Tat gibt es keinen guten Algorithmus zur deterministischen Konstruktion eines Elements g der Ordnung $q-1$. Üblicherweise verwendet man daher in der Computeralgebra bei der Suche nach einem solchen sogenannten primitiven Erzeugers probabilistische Algorithmen, die zufällige Elemente g erzeugen und testen.

d) Anwendung auf Primzahltests

Aus obigem Satz läßt sich leicht ein Test machen, der *garantiert,* daß eine vorgegebene Zahl p prim ist:

Satz: Die natürliche Zahl p ist genau dann prim, wenn es eine ganze Zahl a gibt, so daß

$$a^{p-1} \equiv 1 \bmod p \text{ und } a^{\frac{p-1}{q}} \not\equiv 1 \bmod p \text{ für alle Primteiler } q \text{ von } p-1\,.$$

Beweis: Falls p prim ist, ist $\mathbb{Z}/p$ ein Körper, dessen multiplikative Gruppe die Ordnung $p-1$ hat und zyklisch ist. Ein erzeugendes Element a dieser Gruppe hat daher die Ordnung $p-1$ und erfüllt damit die Bedingung des Satzes.

Umgekehrt folgt für eine Zahl p, die obige Bedingung erfüllt, daß die multipliktive Gruppe von $\mathbb{Z}/p$ die Ordnung $p-1$ hat, d.h. jedes Element außer der Null ist invertierbar. Damit ist $\mathbb{Z}/p$ ein Körper, also p eine Primzahl.

■

Die praktische Nützlichkeit dieses Satzes ist leider eher klein, denn er ist nur anwendbar, wenn man alle Primteiler von $p-1$ kennt, d.h. man braucht eine komplette Faktorisierung von $p-1$. Eine Modifikation dieses Satzes führte aber immerhin im August 2002 zum ersten Algorithmus, der beweisbar in polynomialer Zeit feststellt, ob eine gegebene Zahl prim ist oder nicht; er ist unter

`http://www.cse.iitk.ac.in/news/primality.X`

zu finden mit `X` = `ps` oder `pdf` für die Arbeit von Manindra Agrawal, Neeraj Kayal, Nitin Saxena und `X` = `html` für eine Übersicht.

Seine Laufzeit wächst allerdings mit mindestens der sechsten Potenz der Ziffernlänge von p, so daß auch er bei den heute interessierenden Zahlen weder für Kryptologen noch für algorithmische Zahlentheoretiker interessant ist: Diese haben Algorithmen, die zwar asymptotisch schlechter, dafür aber in realistischen Anwendungen deutlich besser sind.

Diese Algorithmen benutzen anspruchsvollere Mathematik als Fermat; die meisten arbeiten mit Charaktersummen und/oder elliptischen Kurven. Da sich Fermat, wie wir gesehen haben, nur selten irrt, sei hier hier auf ihre Behandlung verzichtet.

e) Wie dicht liegen die Primzahlen?

Bevor man daran geht, Zufallszahlen zu testen, sollte man zunächst wissen, wie wahrscheinlich es ist, daß eine zufällig gewählte Zahl prim ist. Dies ist ein sehr altes Problem, das immer noch nicht ganz vollständig gelöst ist; die bekannte Antwort ist aber für praktische Zwecke gut genug. Die Mathematik, die im Beweis der folgenden Aussagen steckt, ist allerdings leider jenseits des zeitlichen Rahmens dieser Vorlesung – obwohl einige der Beweise inzwischen vergleichsweise sehr elementar geworden sind. Wer einigermaßen mit Funktionentheorie vertraut ist, sollte in der Lage sein, die Darstellung in

HELMUT KOCH: Einführung in die klassische Mathematik I, *Akademie-Verlag* und (in Lizenz) *Springer-Verlag,* 1986, §27

zu lesen; alle notwendigen Voraussetzungen aus Funktionentheorie, Zahlentheorie usw. sind im Buch selbst zu finden.

Wir bezeichnen die Anzahl der Primzahlen, die kleiner oder gleich einer Zahl x sind, mit $\pi(x)$. Demnach ist also $\pi(2) = 1$ und $\pi(3) = \pi(4) = 2$ und so weiter. Der englische Mathematiker SYLVESTER bewies 1892 folgende Verschärfung einer etwa vierzig jahre älteren Ungleichung von TSCHEBYSCHEFF:

$$0{,}95695\frac{x}{\ln x} < \pi(x) < 1{,}04423\frac{x}{\ln x}\,,$$

$\pi(x)$ verhält sich also asymptotisch ungefähr wie $x/\ln x$. Nach dem Primzahlsatz, den GAUSS vermutete und den HADAMARD und DE LA VALLÉE POUSSIN 1896 bewiesen, ist die Asymptotik sogar exakt, d.h.

$$\lim_{x\to\infty}\frac{\pi(x)}{x/\ln x} = 1\,.$$

Numerisch besser ist die Approximation durch den Integrallogarithmus:

$$\pi(x) = \int_2^x \frac{d\xi}{\ln\xi} + O(xe^{-c\sqrt{\ln x}})$$

mit einer (im Prinzip berechenbaren) Konstanten $c > 0$.

Für uns wichtig ist die Folgerung: Unter den Zahlen der Größenordnung N haben der Primzahlen die Dichte $1/\ln N$, d.h. der Abstand zwischen zwei Primzahlen liegt im Durchschnitt bei etwa $\ln N$. Für $N = 10^n$ ist dies

$$\ln 10^n = n\ln 10 \approx 2{,}3n\,;$$

bei hundertstelligen Zahlen ist also etwa jede 230ste prim.

f) Das Sieb des Eratosthenes

Da ein FERMAT-Test für hundertfünfzigstellige Zahlen einiges an Zeit kostet, verzichtet man meist auf die Erzeugung wirklich zufälliger Primzahlen sondern nimmt stattdessen zu einer Zufallszahl n die kleinste Primzahl $p \geq n$. Damit sind nicht mehr alle Primzahlen gleich wahrscheinlich, sondern eine Primzahl wird mit umso größerer Wahrscheinlichkeit gewählt, je weiter sie von der nächstkleineren Primzahl entfernt ist. Bislang wurden keine Ansätze bekannt, wie ein Gegner diese Ungleichbehandlung der Primzahlen ausnutzen kann.

Diese Zahl p kann man wie folgt suchen: Man definiert ein Suchintervall $[n, n + a]$, in dem Primzahlen zu erwarten sind; beispielsweise könnte man $a \approx 5\ln n$ oder $a \approx 10\ln n$ setzen, so daß etwa fünf oder zehn Primzahlen zu erwarten sind.

In diesem Intervall sind alle geraden Zahlen uninteressant; diese können also gleich gestrichen werden. Sodann sind auch alle Dreierzahlen uninteressant; wir bestimmen also $n \bmod 3$; sodann kennen wir die erste auf n folgende Dreierzahl und streichen von dort ausgehend jede dritte Zahl. Genauso geht es weiter mit fünft, sieben und so weiter, bis zu einer Primzahl, die etwa die Gröse von a hat. (Bei größeren Primzahlen gibt es nur noch selten etwas zu streichen.) Die übriggebliebenen Zahlen haben dann zumindest keine kleinen Primteiler, und wir beschränken uns darauf, den FERMAT-Test für diese Zahlen durchzuführen.

§5: Faktorisierungsverfahren

Der offensichtliche Angriff auf RSA ist die Faktorisierung der öffentlich bekannten Zahl N; sobald man diese in ihre beiden Primfaktoren p und q zerlegt hat, ist das Verfahren gebrochen. Wir wollen daher in diesem Paragraphen sehen, welche Möglichkeiten es gibt, N in seine Primfaktoren zu zerlegen.

a) Mögliche Ansätze zur Faktorisierung

Grundsätzlich gibt es zwei Klassen von Verfahren, mit denen man einen Teiler einer natürlichen Zahl N finden kann: Einmal Verfahren, deren erwartete Laufzeit von der Länge des Faktors abhängt, zum anderen solche, deren Laufzeit nur von N abhängt.

Bei der Anwendung von RSA wird man, um Verfahren der ersten Kategorie auszubremsen, p und q ungefähr gleich groß wählen, so daß diese Verfahren im schlechtestmöglichen Fall arbeiten müssen.

Die einfachste Art der Faktorisierung ist das Abdividieren von Primzahlen; zumindest für kleine Primfaktoren; mindestens bis etwa 2^{16}) ist dies auch die schnellste und effizienteste Methode, da die anderen Verfahren Schwierigkeiten haben, Produkte kleiner Primzahlen zu trennen.

Für etwas größere Faktoren bis zu etwa acht Dezimalstellen ist die POLLARDsche Monte-Carlo-Methode oder ρ-Methode sehr gut geeignet: Man erzeugt mit einem quadratischen Generator (z.B. $x_{i+1} = x_i^2 + c \bmod N$) Zufallszahlen und berechnet deren ggT mit der zu faktorisierenden Zahl. Da der EUKLIDische Algorithmus im Vergleich zur Erzeugung der Zufallszahlen relativ teuer ist, empfiehlt es sich, die erzeugten Zufallszahlen zunächst modulo N miteinander zu multiplizieren und dann erst in etwa jedem hundertsten Schritt den ggT von N mit diesem Produkt zu berechnen. (Dies setzt voraus, daß alle sehr kleinen Faktoren bereits abdividert sind; sonst ist die Gefahr zu groß, daß im Produkt von hundert Zufallszahlen mehr als ein Primfaktor steckt.) Bei Faktoren mit mehr als acht Dezimalstellen wird die Methode schnell langsamer, so daß man dann zu alternativen Verfahren übergehen sollte.

Die nächste Klasse von Verfahren beruht auf gruppentheoretischen Überlegungen, im wesentlichen dem kleinen Satz von FERMAT im Falle zyklischer Gruppen und Verallgemeinerungen auf weitere Gruppen wie die multiplikative Gruppe eines Körpers $\mathbb{F}_{p^2}$ oder einer elliptischen Kurve. Diese Verfahren sind sehr effizient, wenn die Gruppenordnung nur relativ kleine Primteiler hat. Aus diesem Grund wurde früher häufig empfohlen, daß für die Primteiler p eines RSA-Modulus N sowohl $p-1$ als auch $p+1$ jeweils mindestens einen „großen“ Primfaktor haben sollen. Da diese Verfahren jedoch ihre Stärke bei Faktoren mit einer Länge von bis etwa 30 oder 35 Dezimalstellen haben und ein N mit 70 Dezimalstellen, wie wir bald sehen werden, heute auch aus anderen Gründen völlig unsicher ist, gilt diese Empfehlung heute nicht mehr, und wir müssen uns nicht genauer mit diesen Verfahren befassen.

Umso interessanter ist dagegen ein Verfahren, dessen Stärke bei nahe beieinander liegenden Primfaktoren liegt. Es wurde von PIERRE DE FERMAT vorgeschlagen und beruht auf der Formel

$$x^2 - y^2 = (x+y)(x-y)\,.$$

Ist $N = pq$ Produkt zweier ungerader Primzahlen, so ist

$$N = (x+y)(x-y) \quad \text{mit} \quad x = \frac{p+q}{2} \quad \text{und} \quad y = \frac{p-q}{2}\,;$$

zusammen mit obiger Formel folgt, daß dann

$$N + y^2 = x^2$$

ist.

FERMAT berechnet nun für $y = 0, 1, 2, \ldots$ die Zahlen $N + y^2$; falls er auf ein Quadrat x^2 stößt, berechnet er

$$\mathrm{ggT}(N, x+y) \quad \text{und} \quad \mathrm{ggT}(N, x-y)\,.$$

Wenn er Pech hat, sind dies die beiden Zahlen eins und N, wenn er Glück hat, sind es p und q. Für zufällig gewählte Paare (x, y) kommt beides jeweils mit fünfzigprozentiger Wahrscheinlichkeit vor.

Falls p und q nahe beieinander liegen, führt schon ein kleines y zur korrekten Faktorisierung; bei der Wahl der Primzahlen muß also darauf geachtet werden, daß sie zwar die gleiche *Größenordnung* haben, aber nicht zu weit beieinander liegen. Liegt etwa p in der Größenordnung von $2q$, hat die Differenz die gleiche Größenordnung wie p, und FERMATS Verfahren bräuchte etwa p Rechenschritte, wird also vom Aufwand her vergleichbar mit der Faktoriserung durch Abdividieren. Somit kann man sich auch gegen diese Attacke recht gut schützen.

Wirklich gefährlich sind eine Klasse von Siebverfahren, die auf FERMATS Methode aufbauen. Diese Verfahren sind die schnellsten derzeit bekannten zur Faktorisierung von RSA-Moduln; die Wahl einer sicheren Ziffernlänge hängt also davon ab, welche Zahlen diese Verfahren faktorisieren können.

b) Das quadratische Sieb

Das quadratische Sieb ist der Grundalgorithmus der ganzen Klasse; es ist logisch einfacher, allerdings vor allem für große Zahlen auch deutlich langsamer als die Variationen, mit denen wir uns im nächsten Abschnitt beschäftigen werden.

Bei allen diesen Verfahren geht es darum, Zahlenpaare (x, y) zu finden, für die

$$x^2 \equiv y^2 \bmod N$$

ist. Für diese erwarten wir, daß in etwa der Hälfte aller Fälle

$$\operatorname{ggT}(x+y, N) \quad \text{und} \quad \operatorname{ggT}(x-y, N)$$

nichttriviale Teiler von N sind.

Bei quadratischen Sieb betrachten wir dazu das Polynom

$$f(x) = \left(x + \left[\sqrt{N}\right]\right)^2 - N\,.$$

Offensichtlich ist für jedes x

$$f(x) \equiv \left(x + \left[\sqrt{N}\right]\right)^2 \bmod N\,,$$

wobei links und rechts verschiedene Zahlen stehen. Insbesondere steht links im allgemeinen keine Quadratzahl.

Falls wir allerdings Werte $x_1, x_2, \ldots, x_r$ finden können, für die das Produkt der $f(x_i)$ eine Quadratzahl ist, dann ist

$$\prod_{i=1}^{r} f(x_i) \equiv \prod_{i=1}^{r} \left(x + \left[\sqrt{N}\right]\right)^2 \bmod N$$

eine Relation der gesuchten Art.

Um die x_i zu finden, betrachten wir eine Menge $\mathcal{B}$ von Primzahlen, die sogenannte Faktorbasis. Typischerweise enthält $\mathcal{B}$ für die Faktorisierung einer etwa hundertstelligen Zahl etwa 100–120 Tausend Primzahlen, deren größte somit, wie die folgende Tabelle zeigt, im einstelligen Millionenbereich liegt.

n	n-te Primzahl	n	n-te Primzahl
100 000	1 299 709	600 000	8 960 453
200 000	2 750 159	700 000	10 570 841
300 000	4 256 233	800 000	12 195 257
400 000	5 800 079	900 000	13 834 103
500 000	7 368 787	1 000 000	15 485 863

Beim quadratischen Sieb interessieren nur x-Werte, für die $f(x)$ als Produkt von Primzahlen aus $\mathcal{B}$ (und eventuell auch Potenzen davon) darstellbar ist. Ist

$$f(x_i) = \prod_{p\in\mathcal{B}} p^{e_{ip}}\,,$$

so ist

$$\prod_{i=1}^{r} f(x_i)^{\varepsilon_i} = \prod_{p\in\mathcal{B}} p^{\sum_{i=1}^{r} \varepsilon_i e_{ip}}$$

genau dann ein Quadrat, wenn

$$\sum_{i=1}^{r} \varepsilon_i e_{ip}$$

für alle $p \in \mathcal{B}$ gerade ist. Dies hängt natürlich nur ab von den $\varepsilon_i \bmod 2$ und den $e_{ip} \bmod 2$; wir können ε_i und e_{ip} daher als Elemente des Körpers mit zwei Elementen auffassen und bekommen dann über $\mathbb{F}_2$ das Gleichungssystem

$$\sum_{i=1}^{r} \varepsilon_i e_{ip} = 0 \quad \text{für alle } p \in \mathcal{B}\,.$$

Betrachten wir die ε_i als Variablen, ist dies ein homogenes lineares Gleichungssystem in r Variablen mit soviel Gleichungen, wie es Primzahlen in der Faktorbasis gibt. Dieses Gleichungssystem hat nichttriviale Lösungen, falls die Anzahl der Variablen die der Gleichungen übersteigt, falls es also mehr Zahlen x_i gibt, für die $f(x_i)$ über der Faktorbasis faktorisiert werden kann, als Primzahlen in der Faktorbasis.

Für jede nichttriviale Lösung ist

$$\prod_{i=1}^{r} f(x_i)^{\varepsilon_i} = \prod_{i=1}^{r} \left(x + \left[\sqrt{N}\right]\right)^{2\varepsilon_i} \bmod N$$

eine Relation der Form $x^2 \equiv y^2 \bmod N$, die mit einer Wahrscheinlichkeit von etwa ein halb zu einer Faktorisierung von N führt. Falls wir zehn linear unabhängige Lösungen des Gleichungssystems betrachten, führt also mit einer Wahrscheinlichkeit von etwa 99,9% mindestens eine davon zu einer Faktorisierung.

Da ε_i nur die Werte 0 und 1 annimmt, stehen in obigem Produkt natürlich keine echten Potenzen: Man multipliziert einfach nur die Faktoren miteinander, für die $\varepsilon_i = 1$ ist. Außerdem interessieren nicht die links- und rechtsstehenden Quadrate, sondern deren Quadratwurzeln; tatsächlich also berechnet

$$x = \prod_{p\in\mathcal{B}} p^{\frac{1}{2}\sum_{i=1}^{r} \varepsilon_i e_{ip}} \quad\text{und}\quad y = \prod_{i=1}^{r} \Bigl(x + \bigl[\sqrt{N}\bigr]\Bigr)^{\varepsilon_i} \,.$$

wobei beide Produkte nur modulo N berechnet werden müssen.

Zum besseren Verständnis des Verfahrens wollen wir versuchen, damit die Zahl 15 zu faktorisieren. Dies ist zwar eine sehr untypische Anwendung, da das quadratische Sieb üblicherweise erst für mindestens etwa vierzigstellige Zahlen angewandt wird, aber zumindestens das Prinzip sollte auch damit klarwerden.

Als Faktorbasis verwenden wir die Menge

$$\mathcal{B} = \{2, 3, 7, 11\}\,;$$

die Primzahl fünf fehlt, da $3 \cdot 5 = 15$ ist und daher bei einer Faktorbasis, die sowohl drei als auch fünf enthält, die Gefahr zu groß ist, daß die linke wie auch die rechte Seite der Kongruenz durch fünfzehn teilbar ist. Bei realistischen Anwendungen muß man auf solche Überlegungen keine Rücksicht nehmen, denn dann sind die Elemente der Faktorbasis höchstens siebenstellig und somit erheblich kleiner als die gesuchten Faktoren.

Wir berechnen $f(x)$ für $x = 1, 2, \ldots$, bis wir einige Funktionswerte haben, die über der Faktorbasis faktorisiert werden können. Die faktorisierbaren Werte sind in folgender Tabelle zusammengestellt:

x	$x + \bigl[\sqrt{N}\bigr]$	$f(x)$	*Faktorisierung*
1	4	1	
3	6	21	$3 \cdot 7$
5	8	49	7^2
6	9	66	$2 \cdot 3 \cdot 11$
10	13	154	$2 \cdot 7 \cdot 11$
54	57	3234	$2 \cdot 3 \cdot 7^2 \cdot 11$

Die erste und die dritte Zeile sind selbst schon Relationen der gesuchten Art, nämlich

$$4^2 \equiv 1 \bmod 15 \quad\text{und}\quad 8^2 \equiv 7^2 \bmod 15\,.$$

Die zweite Relation ist nutzlos, denn $8 - 7 = 1$ und $8 + 7 = 15$. Die erste dagegen führt zur Faktorisierung, denn

$$\mathrm{ggT}(4 + 1, 15) = 5 \quad\text{und}\quad \mathrm{ggT}(4 - 1, 15) = 3\,.$$

Da dies aber ein Zufall ist, der bei großen Werten von N so gut wie nie vorkommt, wollen wir das ignorieren und mit den Relationen zu $x = 3, 6, 10$ und 51 arbeiten:

$$\begin{aligned} 6^2 &\equiv 3 \cdot 7 && \bmod 15\\ 9^2 &\equiv 2 \cdot 3 \cdot 11 && \bmod 15\\ 13^2 &\equiv 2 \cdot 7 \cdot 11 && \bmod 15\\ 54^2 &\equiv 2 \cdot 3 \cdot 7^2 \cdot 11 && \bmod 15 \end{aligned}$$

Multipliziert man die ersten drei dieser Relationen miteinander, folgt

$$(6 \cdot 9 \cdot 13)^2 \equiv (2 \cdot 3 \cdot 7 \cdot 11)^2 \bmod 15$$

oder $702^2 \equiv 462^2 \bmod 15$. Da

$$\mathrm{ggT}(702 - 462, 15) = \mathrm{ggT}(240, 15) = 15$$

ist, bringt das leider nichts.

Wir erhalten auch dann rechts ein Quadrat, wenn wir das Produkt der ersten, dritten und vierten Relation bilden; dies führt auf

$$(6 \cdot 13 \cdot 57)^2 \equiv (2 \cdot 3 \cdot 7^2 \cdot 11)^2 \bmod 15$$

oder $4446^2 \equiv 3234^2 \bmod 15$. Hier ist

$$\mathrm{ggT}(4446 - 3234, 15) = \mathrm{ggT}(1212, 15) = 3\,,$$

womit wir die Zahl 15 faktorisiert haben – wenn auch nicht unbedingt auf die einfachstmögliche Weise.

Bei realistischen Beispielen sind die Funktionswerte $f(x)$ deutlich größer als die Primzahlen aus der Faktorbasis; außerdem liegen die vollständig faktorisierbaren Zahlen viel dünner als hier: Bei der Faktorisierung einer hundertstelligen Zahl etwa muß man davon ausgehen, daß nur etwa jeder 10^9-te Funktionswert über der Faktorbasis zerfällt.

Daher ist es wichtig, ein Verfahren zu finden, mit dem diese wenigen Funktionswerte schnell und einfach bestimmt werden können. Das ist zum Glück möglich:

Der Funktionswert $f(x)$ ist genau dann durch p teilbar, wenn

$$f(x) \equiv 0 \bmod p$$

ist. Da für $x, y, a, b \in \mathbb{Z}$ und mit $x \equiv y \bmod p$ und $a \equiv b \bmod p$ gilt

$$a + x \equiv b + y \bmod p \quad \text{und} \quad a \cdot x \equiv b \cdot y \bmod p$$

und für $n \in \mathbb{N}$ auch

$$x^n \equiv y^n \bmod p\,,$$

ist für jedes Polynom f mit ganzzahligen Koeffizienten

$$f(x) \equiv f(y) \bmod p\,.$$

Ist also insbesondere $f(x) \equiv 0 \bmod p$, so ist auch

$$f(x + kp) \equiv 0 \bmod p \quad \text{für alle } k \in \mathbb{Z}\,.$$

Es genügt daher, im Bereich $0 \le x < p - 1$ nach Werten zu suchen, für die $f(x)$ durch p teilbar ist.

Dazu kann man f auch als Polynom über dem Körper mit p Elementen betrachten und nach Nullstellen in diesem Körper suchen. Für Polynome großen Grades und große Werte von p kann dies recht aufwendig sein; hier, bei einem quadratischen Polynom, müssen wir natürlich einfach eine quadratische Gleichung lösen: In $\mathbb{F}_p$ wie in jedem anderen Körper auch gilt

$$f(x) = \left(x - \left[\sqrt{N}\right]\right)^2 - N = 0 \iff \left(x - \left[\sqrt{N}\right]\right)^2 = N\,,$$

und diese Gleichung ist genau dann lösbar, wenn es ein Element $w \in \mathbb{F}_p$ gibt mit Quadrat N, wenn also in $\mathbb{Z}$

$$w^2 \equiv N \bmod p$$

ist. Für $p > 2$ hat $f(x) = 0$ in $\mathbb{F}_p$ dann die beiden Nullstellen

$$x = \left[\sqrt{N}\right] \pm w\,;$$

andernfalls gibt es keine Lösung.

Insbesondere kann also $f(x)$ nur dann durch p teilbar sein, wenn N modulo p ein Quadrat ist; dies ist für etwa die Hälfte aller Primzahlen der Fall. Offensichtlich sind alle anderen Primzahlen nutzlos, die Faktorbasis sollte also nur Primzahlen enthalten, für die N modulo p ein Quadrat ist.

Für solche p kann man dann die beiden Lösungen der Gleichung $f(x) = 0$ in $\mathbb{F}_p$ berechnen: Im Vergleich zum sonstigen Aufwand der Faktorisierung ist die Nullstellensuche durch Probieren durchaus vertretbar, allerdings kann man Quadratwurzeln modulo p mit etwas besseren Zahlentheoriekenntnissen auch sehr viel schneller berechnen.

Das eigentliche Sieben zum Auffinden der komplett über der Faktorbasis zerlegbaren Funktionswerte $f(x)$ geht nun folgendermaßen vor sich: Man legt ein Siebintervall $x = 0, 1, \ldots, M$ fest und speichert in einem Feld der Länge $M + 1$ für jedes x eine ganzzahlige Approximation von $\log_2 f(x)$.

Für jede Primzahl p aus der Faktorbasis berechnet man dann die beiden Nullstellen $x_{1/2}$ von f modulo p im Intervall von 0 bis $p - 1$ und subtrahiert von jedem Feldelement mit Index der Form $x_1 + kp$ oder $x_2 + kp$ eine ganzzahlige Approximation von $\log_2 p$.

Falls $f(x)$ über der Faktorbasis komplett faktorisierbar ist, sollte dann am Ende der entsprechende Feldeintrag bis auf Rundungsfehler gleich null sein; um keine Fehler zu machen, untersucht man daher für alle Feldelemente, die unterhalb einer gewissen Grenze liegen, durch Abdividieren, ob sie wirklich komplett faktorisieren, und man bestimmt auf diese Weise auch *wie* sie faktorisieren. Damit läßt sich dann das oben erwähnte Gleichungssystem über $\mathbb{F}_2$ aufstellen und, falls genügend viele Relationen gefunden sind, nichttrivial so lösen, daß eine der daraus resultierenden Gleichungen $x^2 \equiv y^2 \bmod p$ zu einer nichttrivialen Faktorisierung von N führt.

c) Varianten des quadratischen Siebs

Der Rechenaufwand beim quadratischen Sieb entfällt größtenteils auf das Sieben: Nur ein verschwindend kleiner Teil aller Zahlen zerfällt über der gewählten Faktorbasis, und je größer x wird, desto weniger dicht liegen diese Zahlen. Bei einer Zahl um 10^{100} und einer Faktorbasis aus 10 000 Primzahlen etwa kann man für $x \leq 10^{10}$ etwa fünf vollständig faktorisierbare Werte von $f(x)$ erwarten, im neunmal so großen Intervall $[10^{10}, 10^{11}]$ nur noch etwa 23 und so weiter.

Zumindest qualitativ ist dies klar, denn je größer die Zahlen werden, desto größer wird die Wahrscheinlichkeit großer Primfaktoren. Eine Abschätzung mit (teils nur heuristischen) Formeln über die Verteilung von Primfaktoren zeigt, daß man in einem solchen Fall ein Intervall sieben muß, das bis über 10^{14} hinausreicht. Verbesserungen des quadratischen Siebs konzentrieren sich daher darauf, die Siebphase zu optimieren um so in kürzerer Zeit mehr Relationen zu finden.

1.) Die Multipolynomialversion: Die Multipolynomialversion des quadratischen Siebs optimiert dieses an zwei Stellen: Einmal betrachtet sie außer dem Polynom $\left(x - [\sqrt{N}]\right)^2 - N$ noch weitere Polynome, um Relationen zu bekommen, so daß man jedes dieser Polynome nur über ein kürzeres Intervall sieben muß; zum andern betrachtet sie auch negative Werte von x, so daß – bei geschickt gewählten Polynomen f – der Betrag von $f(x)$ für ein längeres Intervall klein bleibt.

Dabei muß natürlich berücksichtigt werden, daß nun auch negative Werte $f(x)$ auftreten können; dies geschieht einfach dadurch, daß man die Zahl -1 wie eine Primzahl behandelt, die genau dann in der Faktorisierung von $f(x)$ auftritt, wenn $f(x)$ negativ ist.

Als Polynome betrachtet man quadratische Polynome der Form

$$f(x) = ax^2 + 2bx + c \quad \text{mit} \quad 0 \leq b < a \quad \text{und} \quad b^2 - ac = N\,.$$

Für diese ist

$$af(x) = (ax+b)^2 - b^2 + ac = (ax+b)^2 - N\,,$$

so daß $af(x)$ zwar kongruent $(ax+b)^2$ ist, aber nicht gleich. Auch diese Polynome liefern also die Art von Relationen, die wir brauchen, und sie können genauso gesiebt werden wie das spezielle Polynom aus dem letzten Abschnitt.

Ein gewisser Nachteil dabei ist, daß man vor dem Sieben für jedes neue Polynom f und jede Primzahl p neu die Nullstellen von f modulo p ausrechnen muß. Bei geschickter Vorgehensweise fällt diese Zeit jedoch kaum ins Gewicht:

Zur Konstruktion von Polynomen f wählt man zunächst eine Zahl a so, daß das Polynom in einem Intervall $[-M, M]$ möglichst beschränkt bleibt. Falls a, b, c deutlich kleiner sind als M, liegt der Maximalwert von $f(x)$ an den Intervallenden, liegt das Maximum bei

$$f(-M) \approx \frac{1}{a}\left(a^2M^2 - N\right);$$

das Minimum wird bei $x = -b/a$ angenommen und ist

$$f\left(-\frac{b}{a}\right) = \frac{b^2}{a} - \frac{2b^2}{a} + c = \frac{-b^2 + ac}{a} = -\frac{N}{a}\,.$$

Für $a \approx \sqrt{2N}/M$ haben beide Zahlen ungefähr denselben Betrag, aber entgegengesetzte Vorzeichen; also wählen wir a in dieser Größenordnung.

Da $b^2 - 4ac = N$ werden muß, kommen für b nur Werte in Frage, für die $b^2 \equiv N \bmod a$ ist, uns sobald ein solches b gewählt ist, liegt auch c eindeutig fest.

Mit Hilfe der Theorie quadratischer Reste lassen sich die Nullstellen modulo p aller möglicher Polynome zu einem gegebenen Wert von a simultan berechnen, wobei der Aufwand pro Polynom ziemlich gering ist. Man verwendet daher bei der Multipolynomialversion des quadratischen Siebs sehr viele Polynome und relativ kleine Siebintervalle $[-M, M]$.

Die Anzahl der Polynome, die Polynome selbst und die Zahl M sollten idealerweise so gewählt werden, daß die Rechenzeit minimal wird. Deren Abschätzung hängt ab von einer ganzen Reihe von Größen, die teils nur mit großem Aufwand berechnet werden können, teils nur modulo unbewiesener Vermutungen wie etwa der RIEMANN-Vermutung bekannt sind und für die teils sogar nur rein heuristische Formeln existieren. Ich möchte auf die damit verbundenen Probleme nicht eingehen, sondern nur das Ergebnis angeben, wonach bei optimaler Wahl aller Größen der Rechenaufwand zum Faktorisieren einer Zahl N mit der Multipolynomialvariante des quadratischen Siebs proportional ist zu $e^{\sqrt{\ln N \ln\ln N}}$.

2.) Das Zahlkörpersieb: Die derzeit schnellste Verbesserung des quadratischen Siebs ist das *Zahlkörpersieb,* das nicht mehr mit quadratischen Polynomen arbeitet, sondern mit Polynomen beliebigen Grades.

Der Grad d dieser Polynome wird in Abhängigkeit der zu faktorisierenden Zahl N festgelegt, sodann wählt man führende Koeffizienten a_d und eine natürliche Zahl

$$m \approx \sqrt[d]{\frac{N}{a_d}}\,.$$

Alle Polynome sind homogen, haben also die Form

$$F(x,y) = a_d x^d + a_{d-1}x^{d-1}y + \cdots + a_1 xy^{d-1} + a_0 y^d\,,$$

wobei die a_i mit $i < d$ höchstens Betrag $m/2$ haben.

Hinzu kommt das homogene lineare Polynom

$$G(x,y) = x - my\,;$$

das Sieb sucht nach Zahlenpaaren (x,y), für die sowohl $F(x,y)$ als auch $G(x,y)$ über der Faktorbasis zerfallen. Da die Polynome von zwei Variablen abhängen, muß man entweder jeweils eine Variable festhalten und über die andere sieben, oder aber ein zweidimensionales Siebverfahren anwenden; üblich ist eine Kombination beider Methoden.

Gesucht sind Paare (x,y) teilerfremder ganzer Zahlen, für die

$$F(x,y) \quad \text{und} \quad G(x,y)$$

beide über der gewählten Faktorbasis zerfallen, und das Ziel ist, wie beim quadratischen Sieb, eine Relation der Form

$$\prod_{(x,y)\in\mathcal{M}} F(x,y) \equiv \prod_{(x,y)\in\mathcal{M}} G(x,y) \mod N\,,$$

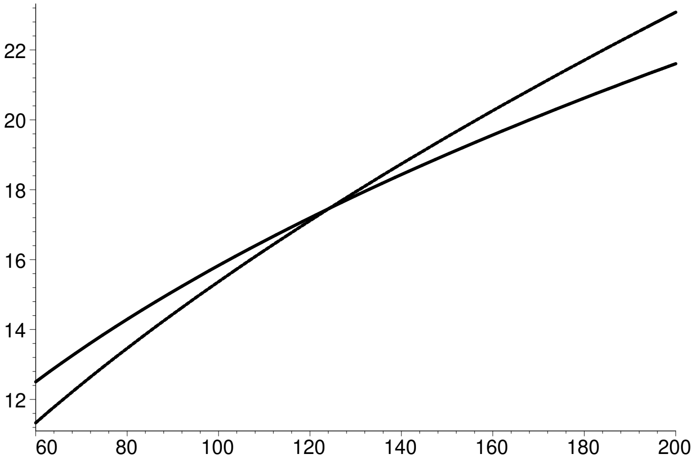
in der rechts und links Quadrate stehen.

Die besten derzeit bekannten Strategien zur Polynomauswahl *usw.* führen auf eine Laufzeitabschätzung proportional

$$e^{c(\ln N)^{\frac{1}{3}}(\ln\ln N)^{\frac{2}{3}}} \quad c = \sqrt[3]{\frac{64}{9}} \approx 1{,}923$$

für die Faktorisierung einer Zahl N nach dieser Methode.

Für hinreichend große N ist diese Methode offensichtlich schneller als das quadratische Sieb, bei dem $\ln N$ als Quadratwurzel im Exponenten steht; in der Abbildung, wo der Aufwand für die Faktorisierung einer Zahl 10^x aufgetragen ist, sieht man, daß das Zahlkörpersieb (durgezogene Linie) ab etwa 125-stelligen Zahlen dem quadratischen Sieb (gestrichelte Linie) überlegen ist.



Zeitaufwand für quadratisches Sieb und Zahlkörpersieb

d) Faktorisierungsrekorde

Neue Faktorisierungsverfahren werden meist vorgestellt mit einem Faktorisierungsbeispiel, das den bisherigen Verfahren getrotzt hat. Berühmt sind dabei die sogenannten *ten most wanted factorisations* des *Cunningham-Projekts,* wo es vor allem um Zahlen der Form $b^n \pm a$ für kleine Werte von a und b. Mit diesen Problemen befaßten sich die algorithmischen Zahlentheoretiker schon lange vor der praktischen Bedeutung von Faktorisierungen im Zusammenhang mit dem RSA-Verfahren.

Im Zusammenhang mit der Kryptographie interessanter ist ein Liste von *challenges,* die *RSA Computer Security Incorperated* regelmäßig zusammenstellt, denn hier geht es um Zahlen, die sorgfältig unter kryptographischen Gesichtspunkten ausgewählt wurden.

Die größte bislang faktorisierte Zahl aus dieser Liste ist RSA-155, eine, wie der Name sagt, 155-stellige natürliche Zahl. Diese Ziffernlänge ist vor allem deshalb interessant, weil sie 512 Binärziffern entspricht, also jener Schlüssellänge, die Web-Browser typischerweise für die sogenannten „sicheren“ Verbindungen wählen und die auch im SSL-Protokoll *(Secure Socket Layer)* verwendet wird.

RSA-155 wurde 1999 mit dem Zahlkörpersieb faktorisiert; beteiligt waren 16 Wissenschaftler von 14 Institutionen aus drei Kontinenten, 160 *workstations,* acht R10000-Prozessoren, 120 Pentium II PCs, vier Digital/Compaq boxes, die in 3,7 Monaten mit einer Rechenleistung von 35,7 CPU-Jahren das Sieben durchführten. Sodann brauchte ein SGI Origin 2000 Computer einen Monat, um eine Matrix mit 6 699 191 Zeilen und 6 711 336 Spalten aufzustellen, aus der schließlich eine CRAY C916 in 214 CPU-Stunden mit zwei Gigabyte RAM 64 Relationen fand. Für vier dieser Relationen berechnete die SGI Origin 2000 die Quadratwurzeln links und rechts mit einem Rechenaufwand von zwischen etwa vierzig und etwa sechzig CPU-Stunden pro Relation. Die anschließenden Faktorisierungsversuch evia EUKLIDischen Algorithmus benötigten natürlich kaum eine Sekunde und lieferten die Zerlegung in zwei 78-stellige Primzahlen.

Wie man sieht, ist diese Faktorisierung noch deutlich jenseits der Möglichkeiten eines Gelegenheitshackers; Zugriff auf ein Netz von etwa dreihundert *workstations* und PCs ist zwar kein großes Problem, aber 214 CPU-Stunden auf einer CRAY sind außer für Regierungsstellen und Forschungszentren höchstens noch für Großunternehmen realistisch, wobei selbst bei denen einiges an kreativer Buchführung notwendig wäre, falls der Faktorisierungsversuch geheim bleiben soll.

Dieser aus Sicht eines Anwenders von 512-Bit RSA-Verschlüsselung einzig positive Aspekt der Faktorisierung von RSA-155 galt aber leider nur wenige Monate; dann wurde eine weitere Faktorisierung einer 512-Bit-Zahl bekannt, die unter völlig anderen Rahmenbedingungen zustandekam.

Der durch sein populärwissenschaftliches Buch zum Beweis der FERMAT-Vermutung bekannte Autor SIMON SINGH veröffentlichte 1999 unter dem Titel *The Code Book* ein Buch über Kryptographie und verband dies mit einem zehnstufigen Preisrätsel, in dem Kryptogramme verschiedener Schwierigkeitsstufe geknackt werden mußten. Die zehnte Stufe enthielt unter anderem eine RSA-Verschlüsselung mit 512 Bit.

Die Gruppe schwedischer Studenten, die das Gesamträtsel im Oktober 2000 als erste löste, bestand nicht, wie die RSA-155-Gruppe, aus führenden Experten für algorithmische Zahlentheorie; sie enthielt kein einziges Mitglied, das die Mathematik des Zahlkörpersiebs verstanden hätte. Da der Programmcode der RSA-155-Faktorisierung allerdings frei erhältlich ist, konnten sie damit arbeiten und ihn an ihre Bedürfnisse anpassen.

Wegen der fehlenden Erfahrung der Anwender war die Polynomauswahl hier deutlich schlechter als bei RSA-155, was zu einer fast doppelt so langen Siebzeit führte.

Das nächste Problem war die Lösung des linearen Gleichungssystems, für die bis dahin nur CRAY-Code zur Verfügung stand. Die Gruppe entwickelte daher ein neues Programm, das auch auf kleineren Rechnern läuft, und löste das Gleichungssystem in dreizehn Tagen auf einem Compaq quad processor ES40 System mit vier Alpha-Prozessoren mit 667 MHz und 8 GB Hauptspeicher, der von der Herstellerfirma zu diesem Zweck zur Verfügung gestellt wurde. Damit war gezeigt, daß auch relative Laien mit relativ beschränkten Mitteln eine 512-Bit-Faktorisierung durchführen können; einschließlich aller theoretischer Vorbereitungen mußten sie dazu ungefähr elf Monate aufwenden.

Es steht zu erwarten, daß entsprechende Faktorisierungen in wenigen Jahren routinemäßig durchgeführt werden können; die Tage der 512-Bit-Moduln sollten also hoffentlich bald gezählt sein.

Die aktuellen *Factoring Challenges* von RSA Security sind unter `www.rsasecurity.com/rsalabs/challenges/factoring/numbers.html` zu finden; sie beginnen mit einer 576-Bit-Zahl (174 Dezimalstellen), die immer noch darauf wartet, daß jemand die 10 000 $ einfordert, die für ihre Faktorisierung ausgesetzt sind. Für die größte Zahl der Liste, RSA-2048 mit 617 Dezimalstellen, gibt es 200 000$.

e) Faktorisierung durch Spezialhardware

Faktorisierungen mit dem quadratischen oder Zahlkörpersieb benötigen zwar zumindest für einige Schritte wie die Lösung des linearen Gleichungssystems leistungsfähige Rechner mit viel Speicher, aber im wesentlichen handelt es sich dabei doch um Standardcomputer. Da die Grundoperation des Siebens so einfach ist, dürfte es sich wohl auch kaum lohnen, dafür spezielle Chips zu entwerfen und einzusetzen.

Es gibt allerdings zumindest einen Vorschlag, wie man das Sieben mit einem optoelektronischen Gerät etwa um den Faktor Tausend beschleünigen kann: ADI SHAMIR, einer der Erfinder des RSA-Verfahrens, entwarf 1999 TWINKLE, *The Weitzman INstitut Key Locating Engine.*

Das Gerät sitzt in einer schwarzen Röhre mit etwa 15 cm Durchmesser und 25 cm Länge, deren wesentlicher Chip im Innern etwa eine Million LEDs enthält. Jede dieser LEDs steht für eine Primzahl p aus der Faktorbasis und hat eine Leuchtkraft proportional $\log_2 p$, was etwa über ihre Größe oder (einfacher) mittels einer Abdeckfolie mit kontinuierlichem Grauschleier realisiert werden kann.

Hierin liegt der wesentliche Unterschied zu Software-Implementierungen der Siebe: Dort werden die Primzahlen nacheinander behandelt, während den x-Werten Speicherzellen entsprechen. Bei TWINKLE werden die x-Werte auf die Zeitachse abgebildet; da die x-Werte, für die $f(x)$ durch p teilbar ist, von der Form $x_{1/2} + kp$ sind, muß also jede LED periodisch aufleuchten, was nicht schwer zu realisieren ist. Die Taktrate, mit der das Gerät arbeitet, soll bei 10 GHz liegen, ein Wert, der in optischen Hochgeschwindigkeitsnetzen heute durchaus normal ist.

In jedem Takt mißt ein den LEDs gegenüberliegender Sensor die Gesamtlichtstärke. Da ein Takt nur eine Länge von 10^{-10} Sekunden hat, läßt sich die Ausbreitungsgeschwindigkeit des Lichts hier nicht vernachlässigen; bei einer Geschwindikeit von etwa 300 000 km/sec legt es pro Takt etwa drei Zentimeter zurück. Die Laufwegunterschiede der Lichtstahlen zwischen den verschiedenen Dioden und der Meßzelle müssen also deutlich kleiner als drei Zentimeter sein. Dies wird dadurch erreicht, daß alle LEDs auf einem einzigen Wafer sitzen.

Die Meßgenauigkeit der Zelle muß nicht übermäßig hoch sein: Beim klassischen Sieb arbeitet man schließlich auch nur mit ganzzahligen Approximationen der $\log_2 p$. Eine Zahl x ist uninteressant, wenn zum entsprechenden Zeitpunkt eine Lichtstärke gemessen wird, die kleiner ist als $\log_2 f(x)$ minus einem Sicherheitsabstand; ansonsten wird sie an konventionelle Elektronik weitergereicht und dort bearbeitet. Wie wir oben gesehen haben, ist dies ein sehr seltenes Ereignis, das im Schnitt höchstens einmal pro einer Milliarde x-Werte eintritt, also etwa zehnmal pro Sekunde. Mit diesen Datenraten können auch einfache Computer leicht fertigwerden.

Das Hauptproblem ist der Bau das Chips mit den Dioden und deren Steuerelektronik; SHAMIR meint, daß dies mit der heute existierende GaAs-Technologie gerade möglich sein sollte, und möglicherweise stehen inzwischen schon solche oder ähnliche Maschinen in Labors von NSA und ähnlichen Organisationen. In der offenen Literatur ist nicht über die Existenz solcher Maschinen bekannt und auch nichts über Pläne welche zu bauen. Der Entwicklungsaufwand dürfte sicherlich in die Hundertausende oder gar Millionen gehen, aber SHAMIR schätzt, daß das Gerät dann mit Stückkosten von etwa 5 000 $ hergestellt werden kann.

2000 stellte SHAMIR zusammen mit A. LENSTRA, einem der Erfinder des Zahlkörpersiebs, eine verbesserte Version vor; die beiden Autoren schätzen, daß diese Version zusammen mit 15 PCs eine 512-Bit-Zahl in einem halben Jahr faktorisieren kann. Wegen der sehr guten Parallelisierbarkeit des Zahlkörpersiebs könnte man mit mehr TWINKLEs und PCs natürlich auf deutlich kürzere Zeiten kommen.

Für 768-Bit-Faktorisierungen schätzen sie den Aufwand auf fünf Tausend TWINKLEs, unterstützt von achtzig Tausend PCs, bei einem Zeitbedarf von insgesamt neun Monaten.

f) Folgerungen für die Wahl der Schlüssellänge

Wie wir bereits gesehen haben, veröffentlichen das Bundesamt für Sicherheit in der Informationstechnik und die Regulierungsbehörde für Telekommunikation und Post jedes Jahr Empfehlungen für die Länge von RSA-Schlüsseln; falls man rechtsgültige elektronische Unterschriften braucht, muß man auf jeden Fall darauf achten, daß während der Vertragslaufzeit diese Mindestanforderungen erfüllt sind.

Für die Übermittlung verschlüsselter privater oder auch geschäftlicher Informationen darf allerdings jeder seine eigenen Standards festlegen; wir wollen sehen, was sich auf Grund der vorherigen Abschnitte für die Sicherheit in Abhängigkeit von der Schlüssellänge folgern läßt.

Die Faktorisierung von RSA-155 zeigt, daß ein öffentlicher Schlüssel N mit 512 Bit faktorisiert werden kann, wenn auch derzeit noch mit beträchtlichem Aufwand sowohl an Rechenzeit als auch an Kalenderzeit. Ein solcher Schlüssel bietet also keine Sicherheit, gegen einen Opponenten, der ein größeres Netzwerk an Rechnern einsetzen kann und für den die Information auch noch in etwa sechs Monaten diesen Einsatz wert ist. (In den nächsten Jahren wird natürlich sowohl die Gesamtrechenzeit als auch die verflossene absolute Zeit für eine solche Faktorisierung recht schnell sinken.) Zur Übertragung einer Kreditkartennummer mit einem Kreditlimit von wenigen Tausend Euro oder eines Paßworts, das man (hoffentlich) ohnehin in wenigen Monaten ändert, sollten 512 Bit allerdings zumindest gegen Gelegenheitshacker im Augenblick noch genügend Schutz bieten.

Jedes weitere Bit Schlüssellänge erhöht die Sicherheit, macht aber wegen der magischen Grenze von 512 Bit bei vieler Software einen Upgrade erforderlich, der in vielen Fällen nicht erhältlich sein dürfte: Für amerikanische Hersteller etwa gelten 512 Bit immer noch als Grenze, jenseits derer für RSA zumindest keine Exportgenehmigungen im vereinfachten Verfahren mehr gegeben werden. Trotzdem gibt es natürlich sowohl Hardware-Unterstützung als auch reine Software-Implementierungen für 2048-Bit-RSA, allerdings kostet Hardware und meist auch Software

Geld, und 2048-Bit-RSA ist um den Faktor $4^3 = 64$ aufwendiger als 512-Bit-RSA, so daß viele Anwender weiter mit nur 512 Bit arbeiten.

Niemand kann vorhersagen, welche neuen Faktorisierungsverfahren in den nächsten Jahren möglicherweise entdeckt werden oder welche völlig neuen Rechnerarchitekturen möglichweise für einen Durchbruch bei der Faktorisierung sorgen werden: Falls beispielsweise die „Quantencomputer“, mit denen wir uns am Ende der Vorlesung beschäftigen werden, je Realität werden mit Quantenregistern, deren Länge in der Größenordnung der Bitzahl der RSA-Moduln liegt, wird die Faktorisierung der Moduln nicht mehr schwieriger sein als die Verschlüsselung eines kurzen Texts.

Bislang ist es allerdings noch ein ungelöstes Problem, die Zahl 15 mit einem Quantencomputer zu faktorisieren; daher wollen wir diese im Augenblick ignorieren und einfach ausrechnen, wie stark der Aufwand für eine Faktorisierung mit dem Zahlkörpersieb als dem schnellsten derzeit bekannten Algorithmus ansteigt, wenn wir die Bitzahl auf mehr als 512 erhöhen.

Die folgende Tabelle zeigt dies anhand der oben angegebenen Formel für den Zeitbedarf des Zahlkörpersiebs:

Dezimalziffern	*Bits*	*Zeitfaktor*
155	512	1,00
160	529	1,89
165	545	3,53
170	562	6,52
175	579	11,93
180	595	21,56
185	612	38,63
190	628	68,51
195	645	120,29
200	662	209,75
210	695	621,91
220	728	1788,09
230	761	4996,63
240	794	13596,95

250	828	36094,23
260	861	93615,09
270	894	237562,59
280	927	590593,30
290	961	1440060,02
300	994	3447569,30
310	1027	8111574,17

Falls man abschätzen möchte, welchen Aufwand ein Gegner *heute* treiben muß, um einen RSA-Schlüssel einer gegebenen Länge zu faktorisieren, gibt diese Tabelle wohl einen recht guten Überblick; falls man allerdings abschätzen möchte, wie lange eine Schlüssellänge noch sicher ist, dürfte sie auf deutlich zu optimistische Schlußfolgerungen führen.

Wie in der Vergangenheit werden wohl auch in der Zukunft immer neue Verfahren gefunden und bei den alten die Konstanten verbessert: So war beispielsweise die Faktorisierung von RSA-155 etwa viermal so schnell, wie anhand der (ebenfalls mit dem Zahlkörpersieb durchgeführten) Faktorisierung von RSA-150 zu erwarten gewesen wäre. Hinzu kommt, daß die Hardware immer schneller wird und vielleicht auch neue Spezialhardware hinzukommt.

Die Autoren der RSA-155-Faktorisierung haben Rekordfaktorisierungen der letzten dreißig Jahre untersucht (erster Rekord war 1970 die 39-stellige FERMAT-Zahl $2^{2^7}+1$, die Maple heute auf einem Pentium II PC mit 266 MHz und 128 kB RAM in gut einer Viertelstunde faktorisiert) und kamen zu der heuristischen Formel, daß eine d-stellige allgemeine Dezimalzahl wohl erstmals im Jahr

$$13{,}24\sqrt[3]{d}+1928{,}6$$

faktorisiert werden dürfte. Für gängige Bitlängen von RSA-Moduln sind also folgende Jahre zu erwarten, in denen entsprechende Moduln erstmalig faktorisiert werden:

Bit:	768	1024	1280	1536	2048	2560	3072	4096
Jahr:	2010	2018	2025	2031	2041	2050	2057	2070

Im Jahr 2100 würden demnach Zahlen mit etwa 7200 Bit faktorisiert; für das Jahr 3000, bis zu dem die Formel ganz sicher nicht gilt, ergäben sich etwa 1,7 Millionen Bit.

Dies bietet zumindest einen gewissen Anhaltspunkt für Verschlüsselungen, die längerfristig sicher sein müssen – auch wenn niemand garantieren kann, daß die nächsten Jahren keinen so dramatischen Durchbruch bei den Faktorisierungsverfahren bringen, daß dadurch all diese Extrapolationen über den Haufen geworfen werden.

§6: Weitere Aspekte zum RSA-Verfahren

Wir haben die Sicherheit des RSA-Verfahrens bislang nur unter dem Aspekt der Faktorisierung des Moduls N betrachtet. Tatsächlich muß man ein Kryptoverfahren aber immer als ganzes betrachten; viele im Prinzip gute Verfahren wurden in der Praxis schon durch unsachgemäße Anwendung angreifbar und konnten geknackt werden. In diesem Paragraphen sollen einige im Zusammenhang mit RSA wesentliche Aspekte diskutiert werden

a) Das Problem der Schlüsselübergabe

Der große Vorteil asymmetrischer Kryptoverfahren besteht darin, daß die beiden Teilnehmer keinen gemeinsamen Schlüssel zu vereinbaren brauchen; jeder verwendet einfach den öffentlichen Schlüssel des Empfängers. Wie bekommt er den aber?

Falls es einem Gegner gelingt, dem Absender einen selbst konstruierten Schlüssel als Schlüssel des Empfängers zu unterschieben, kann nur der *Gegner* die so verschlüsselte Nachricht lesen, nicht aber der intendierte Empfänger.

Für die Sicherheit von RSA ist daher von entscheidender Wichtigkeit, daß die benutzten Schlüssel die korrekten sind. Da E-Mails und Webseiten gefälscht werden können, empfiehlt sich bei unbekannten Empfängern, den Schlüssel von eienr Zentrale zu holen, die Name und Schlüssel des Empfängers digital unterschreibt und dann übermittelt.

Auf diese Weise muß nur der öffentliche Schlüssel der Zentrale allgemein bekannt sein, und der kann leicht über die verschiedensten elektronischen und klassischen Medien häufig publiziert werden. In Deutschland zertifiziert die Regulierungsbehörde für Telekommunikation und Post solche Zentralen, sofern sie die Gewähr für eine sichere und zuverlässige Schlüsselübermittlung bieten.

b) Primzahlen sind Wegwerfartikel

Die Suche nach einer Primzahl mit 1024 Bit kann auf einem nicht sonderlich leistungsfähigen PC rund eine Minute dauern; wenn man viele Schlüssel erzeugen muß, bietet sich also an, mit den teuren Primzahlen sparsam umzugehen.

Genau das darf man aber aus Sicherheitsgründen auf keinen Fall tun: Wenn jemals zwei unterschiedliche Moduln M, N dieselbe Primzahl p enthalten, kann p leicht als ggT von M und N berechnet werden, so daß beide Moduln faktorisiert sind. Der EUKLIDische Algorithmus erfordert auch für 2048-Bit-Zahlen auf einem Standard-PC einen Aufwand, der höchstens im unteren Sekundenbereich liegt; es ist also durchaus möglich, auch bei einer Liste von mehreren Tausend Moduln für jedes Paar den ggT zu berechnen.

Der sichere Umgang mit Primzahlen besteht darin, die Primzahlen sofort nach Berechnung des öffentlichen und des privaten Schlüssels zu vernichten. Die Wahrscheinlichkeit, daß später wieder einmal eine dieser Primzahlen als Zufallsprimzahl auftaucht, liegt bei vernünftiger Implementierung des „Zufalls“ deutlich unter 10^{-100} und kann daher für alle praktischen Zwecke ignoriert werden.

c) Kenntnis des privaten Exponenten führt zur Faktorisierung

Zu zwei RSA-Moduln, die eine Primzahl gemeinsam haben, liefert der EUKLIDische Algorithmus diese Primzahl und damit die Faktorisierung der beiden Moduln. Falls die beiden Moduln allerdings gleich sind, läßt sich mit dem EUKLIDischen Algorithmus natürlich nichts anfangen-.

Innerhalb einer Organisation, in der RSA-Schlüssel zentral vergeben werden, könnte es sich daher anbieten, den Modul N für alle Teilnehmer gleich zu wählen und lediglich für jeden Teilnehmer einen eigenen öffentlichen Exponenten e und zugehörigen geheimen Exponenten d zu erzeugen.

Bei einem solchen System könnte dann natürlich die Zentrale alle Nachrichten aller Teilnehmer entschlüsseln, aber daß kann in gewissen Fällen durchaus erwünscht sein: Falls es sich etwa im Bankenbereich um Nachrichten handelt, die Geld transferieren, muß die Zentrale alles kontrollieren können.

Schlimm wäre allerdings, wenn auch einzelne Teilnehmer alles entschlüsseln könnten, denn dann wäre es möglich, Überweisungsaufträge mit falschem Namen zu unterschreiben, womit jede Art von Betrug möglich würde.

Wir wollen uns überlegen, wie ein Teilnehmer mit seinem öffentlichen und privaten Exponenten die Zahl N faktorisieren kann und damit die privaten Exponenten aller anderer Teilnehmer aus den öffentlichen Exponenten berechnen kann.

Dazu sei $N = pq$ Produkt zweier Primzahlen, e sei der öffentliche Exponent und d der private. Dann ist für alle $a \in \mathbb{Z}$

$$(a^e)^d = a^{ed} \equiv a \bmod N\,;$$

für ein zu N teilerfremdes a ist also

$$a^{de-1} \equiv 1 \bmod N\,.$$

Wir schreiben

$$de - 1 = 2^r \cdot u \quad \text{mit einer ungeraden Zahl } u$$

und betrachten die Folge der Zahlen

$$a^u \bmod N, \quad a^{2u} \bmod N, \quad \ldots, \quad a^{2^r u} \bmod N\,.$$

Die letzte dieser Zahlen ist stets gleich eins; wenn wir Pech haben, ist für das gerade betrachtete a auch schon die erste gleich eins. Wenn nicht, gibt es einen kleinsten Exponenten s, so daß

$$a^{2^s u} \not\equiv 1 \bmod N \quad \text{und} \quad a^{2^{s+1} u} \equiv 1 \bmod N\,.$$

Es könnte sein, daß dann $a^{2^s u} \equiv -1 \bmod N$ ist, aber da N eine zusammengesetzte Zahl ist, muß das nicht der Fall sein: Modulo 15 ist beispielsweise auch vier eine Zahl mit Quadrat eins.

Modulo einer Primzahl hat eins natürlich nur die beiden Quadratwurzeln ± 1, denn die natürlichen Zahlen modulo einer Primzahl bilden einen Körper, und in einem Körper kann das quadratische Polynom $x^2 - 1$ höchstens zwei Nullstellen haben.

Ist aber $x^2 \equiv 1 \bmod N = pq$, so ist auch $x^2 \equiv 1 \bmod p$ und $\bmod q$, also $x \equiv \pm 1 \bmod p$ und $x \equiv \pm 1 \bmod q$, wobei nicht in beiden Fällen das gleiche Vorzeichen stehen muß. In der Hälfte aller Fälle werden beide Vorzeichen gleich sein; dann ist $x \equiv \pm 1 \bmod N$ mit demselben Vorzeichen.

Ist aber etwa $x \equiv 1 \bmod p$ und $x \equiv -1 \bmod q$, so ist

$$p = \mathrm{ggT}(x-1, N) \quad \text{und} \quad q = \mathrm{ggT}(x+1, N)\,,$$

sobald wir eine solche Zahl kennen, haben wir N also faktorisiert.

Wenn wir mit einem zufällig gewählten a so wie oben verfahren, werden wir in etwa der Hälfte aller Fälle eine von ± 1 verschiedene Quadratwurzel der Eins erhalten. Mit nur wenigen Werten für a bekommen wir fast sicher eine Faktorisierung von N.

d) Der chinesische Restesatz

Das Argument im vorigen Abschnitt kann auch zu einer schnelleren Durchführung der RSA-Entschlüsselung und damit zu schnelleren elektronischen Unterschriften führen: Die Berechnung von $m^d \bmod N$ besteht aus Quadrierungen und Multiplikationen modulo N, und deren Aufwand steigt quadratisch mit der Ziffernlänge von N. Kennt man also die Faktorisierung $N = pq$, kann man die beiden Werte $a^d \bmod p$ und $m^d \bmod q$ in jeweils einem Viertel der Zeit für $m^d \bmod N$ berechnen, beide zusammen also in der halben Zeit. Das Gesamtergebnis ist dadurch eindeutig bestimmt, denn da p und q teilerfremd sind, ist jede Zahl die modulo p und modulo q bekannt ist, auch modulo N eindeutig bestimmt.

Dies läßt sich leicht konstruktiv durchführen: Mit dem erweiterten Euklidischen Algorithmus findet man Zahlen α, β, so daß $\alpha p + \beta q = 1$ ist; dann ist

$$\beta q \equiv 1 \bmod p \quad \text{und} \quad \alpha p \equiv 1 \bmod q\,.$$

Für zwei beliebig vorgegebene Zahlen a, b ist entsprechend

$$a\beta q \equiv a \bmod p \quad \text{und} \quad b\alpha p \equiv b \bmod p$$

eine Lösung der Kongruenz

$$x \equiv a \bmod p \quad \text{und} \quad x \equiv b \pmod{q}\,.$$

Da α und β nur einmal für p und q berechnet werden müssen, ist der Aufwand für das Zusammensetzen der Restklassen modulo p und modulo q zu einer Restklasse modulo N sehr gering.

Mit diesem Verfahren läßt sich der Aufwand für eine elektronische Unterschrift also praktisch halbieren, was vor allem dann von Bedeutung ist, wenn die Unterschrift mit einer Smartcard geleistet wird.

Leider ist das Verfahren aber mit einem potentiell katastrophalen Risiko verbunden: Fall bei einer der beiden Rechnungen

$$a \leftarrow m^d \bmod p \quad \text{und} \quad b \leftarrow m^d \bmod q$$

ein Fehler auftritt, ist das Ergebnis für $m^d \bmod N$ korrekt modulo der einen, nicht aber modulo der anderen Primzahl. Nehmen wir etwa an, das Ergebnis modulo q sei falsch; dann wird also eine Unterschrift u berechnet, die modulo p kongruent ist zu m^d, nicht aber modulo q.

Zum Überprüfen der Unterschrift berechnet der Empfänger $u^e \bmod N$ und vergleicht dies mit m; im vorliegenden Beispiel stimmen die beiden Zahlen nur modulo p überein, nicht aber modulo q, sie sind also insbesondere verschieden, so daß die Unterschrift nicht akzeptiert wird.

Da $u^e - m$ aber durch p teilbar ist und nicht durch q, kann der Prüfer nun

$$p = \mathrm{ggT}(u^e - m, N)$$

berechnen und dadurch N faktorisieren; er kann also künftig die Unterschrift des anderen nachmachen.

Hardwarefehler, die bei der Berechnung von einem der beiden Teilergebnisse zu einem falschen Ergebnis führen sind sicherlich sehr seltene Ereignisse, allerdings kann man dem nachhelfen: Durch Magnetfelder, Mikrowelle, Hitze, mechanische Belastung und ähnliches kann man die Karte durchaus so beeinflussen, daß Fehler wahrscheinlich, aber nicht zu wahrscheinlich werden, so daß man eine gute Chance hat, daß genau eines der beiden Zwischenergebnisse falsch berechnet wird.

Sicherer ist es also auf jeden Fall, wenn man trotz des rund doppelt so hohen Aufwands direkt modulo N rechnet; die oben aufgestellte Regel, wonach man die Primzahlen so schnell wie möglich vergessen sollte, hat durchaus ihren Sinn.

Die Methode, eine Zahl modulo N aus ihren Restklassen modulo von Teilern von N zu bestimmen, hat aber noch viele andere Anwendungen, so daß wir den Satz hier allgemein formulieren wollen. Er wurde angeblich früher von chinesischen Generälen benutzt, um Truppenstärken zu berechnen, indem sie die Soldaten in Reihen verschiedener Breite antreten ließen und dann die Anzahl der Soldaten in der letzten Reihe zählten.

Chinesischer Restesatz: Die natürlichen Zahlen $d_1, \ldots, d_r$ seien paarweise teilerfremd und $N = d_1 \cdots d_r$ sei ihr Produkt. Dann hat das Gleichungssystem

$$x \equiv a_1 \bmod d_1, \quad \ldots, \quad x \equiv a_r \bmod d_r$$

für jede Wahl der a_i eine modulo N eindeutig bestimmte Lösung; diese kann mit Hilfe des erweiterten EUKLIDischen Algorithmus berechnet werden.

Beweis: Für nur zwei Zahlen $d_1 = p$ und $d_2 = q$ haben wir das oben nachgerechnet, wobei offensichtlich nur eine Rolle spielte, daß p und q teilerfremd sind, nicht aber, daß sie Primzahlen sind. Der allgemeine Fall folgt durch vollständige Induktion, denn auch $d_1 \cdots d_s$ und $d_1 \cdots d_s d_{s+1}$ sind teilerfremd. ■

e) Kleine öffentliche Exponenten

Da der Verschlüsselungsaufwand bei RSA linear in der Ziffernzahl des Exponenten steigt, wird gelegentlich vorgeschlagen, einen kleinen Verschlüsselungsexponenten e zu verwenden; populär sind etwa $e = 3$ und $e = 2^{16} + 1$.

Im allgemeinen ist das unproblematisch, insbesondere wenn es um den öffentlichen Exponenten zur Überprüfung einer digitalen Unterschrift geht, in einem Fall kann es allerdings Probleme geben: Wenn nämlich dieselbe Nachricht (oder derselbe Nachrichtenteil, wie z.B. eine Anlage oder ein Block ASCII-Kunst am Ende) an mehrere Empfänger geht.

Nehmen wir etwa an, die Nachricht m solle an drei Empfänger geschickt werden, deren öffentliche Schlüssel $(N_1, 3)$, $(N_2, 3)$ und $(N_3, 3)$ seien. Verschickt werden also die drei Blöcke

$$m^3 \bmod N_1, \quad m^3 \bmod N_2 \quad \text{und} \quad m^3 \bmod N_3 .$$

Ein Gegner, der alle drei abfängt, kann dann nach dem chinesischen Restesatz

$$m^3 \bmod N_1 N_2 N_3$$

berechnen, und da m kleiner als jedes N_i sein muß, kennt er somit m^3. Die Berechnung der Kubikwurzel auch einer sehr großen Zahl ist vom Aufwand her mit einer Division vergleichbar, liegt also höchstens im unteren Sekundenbereich.

(Falls N_1, N_2, N_3 nicht paarweise teilerfremd sein sollten, merkt man das bei der Anwendung des erweiterten EUKLIDischen Algorithmus und hat dann sogar eine Faktorisierung von mindestens zwei Moduln, was dann über die privaten Exponenten insbesondere auf die Nachricht führt.)

f) Kleine private Exponenten

Da elektronische Unterschriften häufig mit Smartcards oder in Zukunft vielleicht auch Mobiltelephonen und ähnlichen Geräte mit vergleichsweise schwacher Rechenleistung erzeugt werden, bietet sich an, nicht den öffentlichen, sondern den privaten Exponenten möglichst klein zu wählen. Ein privater Exponent drei wäre natürlich unmöglich, denn der private Exponent ist schließlich geheim und darf nicht durch systematisches Durchprobieren kleiner Zahlen gefunden werden.

Systematisches Durchprobieren ist aber, wie wir bei der Diskussion symmetrischer Kryptoverfahren gesehen haben, mit heutiger Technologie nur bis zu etwa 2^{64} Fällen möglich; 2^{128} Möglichkeiten, wie sie etwa der AES bietet, gelten heute als sicher. Ein privater Exponent mit 128 statt 1024 Bit führt zu einer Reduktion des Rechenaufwands um den Faktor acht, was gerade bei Smartcards spürbar sein sollte.

Leider gilt aber auch hier, wie in so vielen Fällen bei RSA, das Rechenerleichterungen zu Sicherheitsmängeln führen; ein privater Exponent mit 128 Bit würde bei einem Modul von 1024 oder 2048 Bit innerhalb von Sekunden zu dessen Primfaktorzerlegung führen.

Der Grund ist folgender: Der öffentliche Exponent e und der private Exponent d erfüllen die Gleichung

$$de - k(p-1)(q-1) = 1$$

mit einer natürlichen Zahl k. Division durch $d(p-1)(q-1)$ macht daraus

$$\frac{e}{(p-1)(q-1)} - \frac{k}{d} = \frac{1}{d(p-1)(q-1)} .$$

Der linke Bruch hat einen Nenner, der ungefähr die gleiche Größenprdnung hat wie der Moduk $N = pq$; davon subtrahiert wird ein Bruch mit Nenner d, und die rechte Seite der Gleichung sagt uns, daß die Differenz sehr klein ist.

Ist also der private Exponent d klein, so kann der linksstehende Bruch durch einen Bruch mit sehr viel kleinerem Nenner sehr gut approximiert werden. Damit kann ein Gegner noch nichts anfangen, denn er kennt den Nenner $(p-1)(q-1)$ nicht; andererseits kennt er $N = pq$, und die

Differenz ändert sich nicht sehr, falls man den Nenner durch N ersetzt:

$$\begin{aligned}
\left|\frac{e}{N} - \frac{k}{d}\right| &= \left|\frac{e}{N} - \frac{e}{(p-1)(q-1)} + \frac{e}{(p-1)(q-1)} - \frac{k}{d}\right| \\
&\le \left|\frac{e(p-1)(q-1) - epq}{N(p-1)(q-1)}\right| + \frac{1}{d(p-1)(q-1)} \\
&= \frac{e(p+q-1)}{N(p-1)(q-1)} + \frac{1}{d(p-1)(q-1)} .
\end{aligned}$$

Da p und q in der Größenordnung von $\sqrt{N}$ liegen, ist auch das noch eine recht kleine Zahl.

Bei kleinem d kann sich das ein Gegner mittels des folgenden Satzes zunutze machen:

Satz: Für die reelle Zahl $x > 0$ gebe es teilerfremde natürliche Zahlen a, b derart, daß

$$\left|x - \frac{a}{b}\right| < \frac{1}{2b^2} .$$

Dann ist a/b eine Konvergente der Kettenbruchentwicklung von x.

Um damit etwas anfangen zu können, müssen wir zunächst wissen, was die Kettenbruchentwicklung einer reellen Zahl ist. Diese berechnet sich nach folgendem Algorithmus:

1. Schritt: Setze $x_0 = x$ und $a_0 = [x]$.

n-ter Schritt, $n \ge 1$: Falls $x_{n-1} = a_{n-1}$ bricht der Algorithmus an dieser Stelle ab; andernfalls setze

$$x_n = \frac{1}{x_{n-1} - a_{n-1}} \quad \text{und} \quad a_n = [x_n] .$$

Die n-te Konvergente dieser Kettenbruchentwicklung ist die rationale Zahl

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cfrac{\cdots}{a_{n-1} + \cfrac{1}{a_n}}}}} .$$

Der Beweis des obigen Satzes ist elementar, aber langwierig; wir wollen daher darauf verzichten.

Falls man diesen Satz anwenden kann, läßt sich d also bestimmen, indem man die Nenner der Konvergenten der Kettenbruchentwicklung von e/N bestimmt und jeweils durch Ausprobieren nachprüft, ob für einen Zufallsblock a die gewünschte Beziehung $a^{de} \equiv a \bmod N$ gilt.

Eine einfache Abschätzung zeigt, daß er für p und q von etwa gleicher Größe anwendbar ist, sofern d höchstens die Größenordnung von etwa $\sqrt[4]{N}$ hat; neuere, etwas aufwendigere Untersuchungen zeigen, daß auch man d auch noch für $d < N^{0,289}$ rekonstruieren kann; Fachleute erwarten, daß möglicherweise sogar alle $d < \sqrt{N}$ unsicher sind.

Private Exponenten müssen also immer groß sein. Falls man von einem vorgegebenen öffentlichen Exponenten ausgeht, ist das für realistische N mit an Sicherheit grenzender Wahrscheinlichkeit erfüllt; Vorsicht ist nur geboten, wenn man mit dem privaten Exponenten startet.

§7: Literatur

RSA ist immer noch das gebräuchlichste Verfahren der *public key* Kryptographie; von daher gibt es kaum ein nach etwa 1980 erschienenes Lehrbuch der Kryptologie, das kein Kapitel darüber enthält. Beispiele sind die bereits zitierten Werke

Johannes Buchmann: *Einführung in die Kryptograohie,* Springer, 1999

und

Jan C.A. van der Lubbe: *Basic Methods of cryptography,* Cambridge University Press, 1998

Ein Lehrbuch, das sich vor allem mit asymmetrischer Kryptographie beschäftigt und RSAS natürlich ausführlich behandelt, ist

Neal Koblitz: *A Course in Number Theory and Cryptography,* Graduate Texts in Mathematics **114**, Springer ^{2}1994

Dazu kommen eine ganze Reihe von Lehrbüchern der algorithmischen Zahlentheorie, die RSA behandeln, dabei aber ihr Hauptaugenmerk auf Primzahlen und auch Faktorisierung legen. Ein relativ altes Buch dieser Art ist

Hans Riesel: *Prime Numbers and Computer Methods for Factorization,* Birkhäuser, 1985

Richard Crandall, Carl Pomerance: *Prime numbers – A Computational Perspective,* Springer, 2001

Samuel Wagstaff: *Cryptanalysis of Number Theoretic Ciphers,* Chapman & Hall/CRC, ^{2}2003

Mit Implementierungsfragen beschäftigt sich

Michael Welschenbach: *Kryptographie in C und C++,* Springer, 1998

§6 dieser Ausarbeitung folgt weitgehend dem Artikel

Dan Boneh: *Twenty years of Attacks on the RSA Cryptosystem,* Notices of the AMS, February 1999; auch online verfügbar unter `www.ams.org/notices/199902/boneh.pdf` .

Einen Eindruck über die Diskussionen, die zu den jährlichen Empfehlungen über „Geeignete Algorithmen“ führen gibt der Vortrag

Schabhüser: *How to find the "‘socially accepted"’ minimal Keylength for Digital Signature Algorithms,* Vortrag ECC 2002, Essen, `www.exp-math.uni-essen.de/w̃eng/schabhueser_ECC_2002.`$\left\{\begin{array}{l}\texttt{PDF}\\ \texttt{ppt}\end{array}\right.$

Kapitel 6
Verfahren mit diskreten Logarithmen

Wie wir gesehen haben, ist die Schlüsselverteilung bei RSA zwar deutlich einfacher als bei symmetrischen Verfahren, aber es gibt immer noch Probleme, dem Adressaten einer Nachricht sicher den korrekten Schlüssel zuzuordnen, was oft nur über eine Zentrale möglich ist.

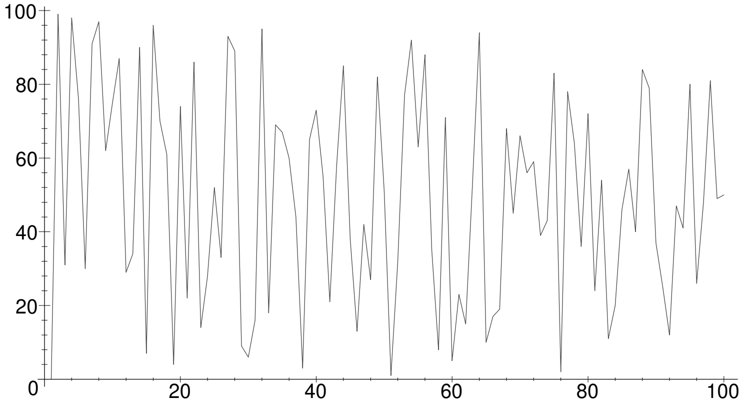
Für die bloße Übertragung einer Leihkartennummer an eine Bibliothek oder kleinere Einkäufe im Internet lohnt es sich kaum, sich bei einer Zentrale registrieren zu lassen (und dafür zu bezahlen); dafür braucht man alternative Verfahren, die ohne den Austausch geheimer Schlüsselinformation auskommen. Typischerweise arbeiten diese mit diskreten Logarithmen.

§1: Diskrete Logarithmen

In $\mathbb{R}$ ist der Logarithmus zur Basis a die Umkehrfunktion der Funktion $x \mapsto a^x$; genauso definieren wir ihn auch für endliche Körper:

$$y = a^x \Longrightarrow x = \log_a y\,.$$

Trotz dieser formalen Übereinstimmung gibt es es allerdings große Unterschiede zwischen reellen Logarithmen und ihren Analoga in endlichen Körpern: Während reelle Logarithmen sanft ansteigende stetige Funktionen sind, die man leicht mit beliebig guter Genauigkeit annähern kann, sieht der diskrete Logarithmus typischerweise so aus wie in der Abbildung zu sehen ist. Auch ist im Reellen der Logarithmus zur Basis $a > 1$ für jede positve Zahl definiert; in endlichen Körpern ist es viel schwerer zu entscheiden, ob ein bestimmter Logarithmus existiert: Modulo sieben etwa sind $2, 4$ und 1 die einzigen Zweierpotenzen, so daß $3, 5$ und 6 keine Zweierlogarithmen haben. Ein Satz aus der Algebra besagt allerdings, daß es stets Elemente a gibt, für die a^x jeden Wert außer der Null annimmt, die sogenannten primitiven Wurzeln. In $\mathbb{F}_7$ wären dies etwa drei und fünf.



Die Funktion $\log_{51} x$ in $\mathbb{F}_{101}$

Die Berechnung der Potenzfunktion durch sukzessives Quadrieren und Multiplizieren ist auch in endlichen Körpern einfach, für ihre Umkehrfunktion, den diskreten Logarithmus gibt es aber derzeit nur deutlich schlechtere Verfahren. Die derzeit besten Verfahren zur Berechnung von diskreten Logarithmen in Körpern mit N Elementen erfordern etwa denselben Aufwand wie die Faktorisierung eines RSA-Moduls der Größenordnung N; diese Diskrepanz zwischen Potenfunktion und Logarithmen kann kryptologisch ausgenutzt werden.

Als Körper verwendet man entweder Körper von Zweipotenzordnung, da man in diesen gut rechnen kann, oder Körper von Primzahlordnung. Da es für viele interessante Körper von Zweipotenzordnung bereits Chips gibt, die dort diskrete Logarithmen berechnen, dürften Körper von Primzahlordnung bei ungefähr gleicher Elementanzahl wohl etwas sicherer sein: Es gibt einfach viel mehr Primzahlen als Zweierpotenzen, und jeder Fall erfordert einen neue Hardwareentwurf. Falls man die

Primzahlen hinreichend häufig wechselt, dürfte sich dieser Aufwand für kaum einen Gegner lohnen.

Da Körper von Primzahlordnung auch einfacher sind als solche von Primzahlpotenzordnung, wollen wir uns hier auf die ersteren beschränken; wer mit endlichen Körpern vertraut ist, sollte keine Schwierigkeiten haben, alles auf beliebige endliche Körper zu verallgemeinern.

§ 2: Diffie-Hellmann und ElGamal-Verfahren

Beim Diffie-Hellmann-Verfahren, dem ältesten auf der Grundlage diskreter Logarithmen, geht es darum, wie zwei Teilnehmer, die weder über gemeinsame Schlüsselinformation noch über eine sichere Leitung verfügen, einen Schlüssel vereinbaren können.

Nach DIFFIE-HELLMANN einigen sie sich zunächst (über die unsichere Leitung) auf eine Primzahl p und eine natürliche Zahl a derart, daß die Potenzfunktion $x \mapsto a^x$ möglichst viele Werte annimmt.

Als nächstes wählt Teilnehmer A eine Zufallszahl $x < p$ und B entsprechend $y < p$; A schickt $u = a^x \bmod p$ an B und erhält dafür $v = a^y \bmod p$.

Sodann berechnet A die Zahl

$$v^x \bmod p = \left(a^y\right)^x \bmod p = a^{xy} \bmod p$$

und B entsprechend

$$u^y \bmod p = \left(a^x\right)^y \bmod p = a^{xy} \bmod p\,;$$

beide haben also auf verschiedene Weise dieselbe Zahl berechnet, die sie nun als Schlüssel in einem klassischen Kryptosystem verwenden können: Beispielsweise könnten die letzten 128 Bit der Zahl als AES-Schlüssel dienen.

Ein Gegner, der den Datenaustausch abgehört hat, kennt die Zahlen p, a, u und v; um $a^{xy} \bmod p$ zu finden, muß er den diskreten Logarithmus von u oder v berechnen.

Das Verfahren von ELGAMAL ist eine einfache Variante, die daraus ein Verschlüsselungsverfahren machet: Nachrichten werden durch ganze Zahlen zwischen 0 und $p-1$ dargestellt und vor der Übertragung mit a^{xy} modulo p multipliziert; der Chiffretext zum Klartext m ist also

$$a^{xy} m \bmod p\,.$$

Da Inversenbildung im Körper mit p Elementen mittels des EUKLIDischen Algorithmus leicht möglich ist, kann der Empfänger durch Multiplikation mit dem inversen Element zu $a^{xy} \bmod p$ den Klartext rekonstruieren.

In dieser Form ist das Verfahren nur für die Übertragung eines Blocks geeignet, denn da die Verschlüsselungsfunktion einfach eine Multiplikation ist, gilt für die Chiffretexte c_1, c_2 zu zwei Klartexten m_1, m_2

$$c_1^{-1} c_2 = m_1^{-1} m_2\,;$$

ein Lauscher, der einen Klartextblock kennt oder errät (Briefkopf, Schlußformel, ...), kann mithin auch alle anderen Blöcke entschlüsseln. Falls also A an B sendet, muß a für jeden Block m_i eine neue Zufallszahl x_i erzeugen und $u_i = a^{x_i} \bmod p$ berechnen; tatsächlich übertragen werden also die beiden Blöcke

$$u_i \quad \text{und} \quad a^{x_i y} m_i\,.$$

Der Chiffretext ist damit doppelt so lang wie der Klartext, so daß das Verfahren für lange Texte nicht attraktiv ist.

§ 3: Das Verfahren von Massey-Omura

Auch bei diesem Verfahren geht es um Nachrichtenaustausch zwischen zwei Partnern A und B, die über keinerlei Schlüsselinformation verfügen. Die beiden einigen sich auf eine Primzahl p, und jeder erzeugt sich einen (geheimzuhaltenden) Exponenten e_A *bzw.* e_B, der prim ist zu $p-1$. Dazu berechnet er nach dem erweiterten EUKLIDischen Algorithmus ein (ebenfalls geheimzuhaltendes) Inverses modulo $p-1$; diese Inversen seien d_A und d_B. Nach dem kleinen Satz von FERMAT ist somit für jedes $m \in \mathbb{Z}$

$$m^{e_A d_A} \equiv m^{e_B d_B} \equiv m \bmod p\,.$$

Will nun A eine Nachricht m verschlüsselt an B schicken, so schickt er m^{e_A} mod p. Damit kann natürlich weder B noch ein etwaiger Lauscher etwas anfangen: Da niemand außer A den Verschlüsselungsexponenten e_A und den Entschlüsselungsexponenten d_A kennt, ist das einfach *irgendeine* Potenz zu *irgendeiner* Basis. Selbst ein BAYESscher Gegner, der alle Kombinationen (M,e) mit $M^e \equiv m^{e_A}$ mod p durchprobieren kann, wird dort für große p eine Fülle von potentiellen Klartexten finden, die alle ungefähr gleich wahrscheinlich sind.

Der Empfänger schickt die Nachricht daher gleich wieder zurück, potenziert sie aber vorher mit seinem Verschlüsselungsexponenten e_B. Was A erhält, ist also $m^{e_A e_B}$, eine Nachricht die niemand entschlüsseln kann.

A potenziert diese Nachricht mit seinem Entschlüsselungsexponenten d_A und erhält

$$\left(m^{e_A e_B}\right)^{d_A} = m^{e_A e_B d_A} = m^{e_A d_A e_B} = \left(m^{e_A d_A}\right)^{e_B} \equiv m^{e_B} \bmod p\,.$$

Diese Nachricht schickt er an B, der nun mit seinem Entschlüsselungsexponenten d_B leicht den Klartext ermitteln kann.

Auch die Sicherheit dieses Verfahrens hängt an diskreten Logarithmen: Ein etwaiger Lauscher kennt die Zahlen

$$m^{e_A} \bmod p, \quad m^{e_A e_B} = \left(m^{e_A}\right)^{e_B} \quad \text{und} \quad m^{e_B} = \left(m^{e_A e_B}\right)^{d_A}\,;$$

falls er in der Lage ist, diskrete Logarithmen modulo p zu berechnen, kann er also e_B bestimmen als den diskreten Logarithmus von $m^{e_A e_B}$ zur Basis m^{e_A} und d_A als diskreten Logarithmus von $\left(m^{e_A e_B}\right)^{d_A}$ zur Basis $m^{e_A e_B}$. Damit kann auch er m berechnen, indem er beispielsweise m^{e_A} modulo p mit d_A potenziert. Die Primzahl p muß also auch bei diesem Verfahren so groß sein, daß die Berechnung diskreter Logarithmen modulo p zumindest praktisch undurchführbar ist.

§4: Strategien zur Berechnung diskreter Logarithmen

Genau wie es zahlreiche Ansätze gibt, ganze Zahlen auf mehr oder weniger effiziente Weise zu faktorisieren, gibt es auch die verschiedensten Methoden, den diskreten Logarithmus x einer Zahl u zur Basis a modulo p zu berechnen.

Am einfachsten und langwierigsten ist Probieren: Man berechnet einfach systematisch alle Potenzen von a modulo p, bis man das Ergebnis u erhält; im schlimmsten Fall sind dies p Multiplikationen, im Durchschnitt $p/2$.

Besser ist das *baby step – giant step* Verfahren, mit dem sich der Aufwand immerhin auf etwa $\sqrt{p}$ reduzieren läßt. Dazu wählt man eine natürliche Zahl m die ungefähr gleich $\sqrt{p}\cdot\log_2 p$ ist, und berechnet zunächst die Potenzen a^i mod p für $i \le m$; das sind m sogenannte *baby steps.*

Bei den dann folgenden *giant steps* berechnet man, um den diskreten Logarithmus von x zu erhalten, die Elemente xa^{-mj} mod p für $j = 1, 2, \ldots$ und vergleicht sie mit den Potenzen aus dem ersten Teil. Ein solcher Vergleich kann etwa über eine binäre Suche oder eine *hash*-Tabelle implementiert werden und hat einen Aufwand von $\log_2 p$.

Sobald man einen Wert xa^{-mj} gefunden hat, der mit einer der in den *baby steps* berechneten Potenzen a^i übereinstimmt, hat man die Gleichung

$$xa^{-mj} = a^i \bmod p \qquad \text{oder} \qquad x = a^{mj+i} \bmod p\,,$$

der diskrete Logarithmus von x zur Basis a ist also $mj+i$. Die notwendige Anzahl von *giant steps* liegt im schlimmsten Fall bei $p/m \approx \sqrt{p}$; im Durchschnitt ist sie halb so groß.

Diese Reduktion des Rechenaufwands auf die Quadratwurzel des naiven Werts erinnert an die allererste Reduktion bei den Faktorisierungsalgorithmen für eine natürliche Zahl N: Auch hier kommt man auf einen Aufwand von „nur“ $\sqrt{N}$, wenn man sich auf das Durchprobieren potentieller Teiler bis $\sqrt{N}$ beschränkt.

Allgemein zeigt die bisherige Erfahrung, daß es zu jedem Faktorisierungsalgorithmus einen Algorithmus zur Berechnung diskreter Logarithmen gibt, der die gleiche Komplexität hat – auch wenn bislang niemand beweisen konnte, daß es einen solchen Zusammenhang gibt und auch kein Grund erkennbar ist, warum es ihn geben sollte.

Die derzeit besten Faktorisierungsalgorithmen beruhen auf dem quadratischen und dem Zahlkörpersieb; für beide wurden bald nach ihrer

Einführung ähnliche Siebalgorithmen gefunden, die zur Berechnung diskreter Logarithmen führen. Wir beschränken uns hier, wie auch schon bei der Faktorisierung, auf das quadratische Sieb, dessen Variante für diskrete Logarithmen als *Indexkalül* bezeichnet wird, und auch hier beschränken wir uns auf eine einfache Variante speziell für Primkörper $\mathbb{F}_p$.

Wie beim quadratischen Sieb wird eine Schranke B festgelegt und damit eine Faktorbasis $\mathcal{B}$ definiert; diese besteht hier aus *allen* Primzahlen $q \le B$. Der Algorithmus besteht aus zwei Teilen:

Im ersten Teil berechnet man die diskreten Logarithmen aller Primzahlen q aus der Faktorbasis zur gegebenen Basis a modulo p. Dies mag auf den ersten Blick unsinnig erscheinen, denn schließlich suchen wir den diskreten Logarithmus *einer* Zahl und beginnen dazu mit der Berechnung der diskreten Logarithmen vieler Zahlen.

Die Logarithmen der Primzahlen lassen sich aber simultan wie folgt berechnen: Man berechne viele Potenzen $a^y \bmod p$ und suche diejenigen, die eine Primfaktorzerlegung mit lauter Faktoren aus $\mathcal{B}$ haben. Ist

$$a^y \bmod p = q_1^{e_1} \cdots q_r^{e_r} ,$$

so ist

$$x \equiv e_1 \log_a q_1 + \cdots + e_r \log_a q_r \bmod m ,$$

wobei m die kleinste natürliche Zahl ist, für die $a^m \equiv 1 \bmod p$. Für eine primitive Wurzel a modulo p ist $m = p - 1$, ansonsten kann m auch ein echter Teiler davon sein.

Mit genügend vielen Gleichungen dieser Form hat man ein lineares Gleichungssystem für die Logarithmen der $q \in \mathcal{B}$, allerdings leider nicht über einem Körper, sondern modulo der im allgemeinen zusammengesetzten Zahl m. Falls m Produkt von Primzahlen ist, löst man das Gleichungssystem modulo jeder dieser Primzahlen und setzt die Lösungen nach dem chinesischen Restesatz zusammen; wenn auch echte Primzahlpotenzen P^s in m stecken, schreibt man die e_i und die linken Seiten y im Zahlensystem zur Basis P und erhält dann für jede Ziffer ein lineares Gleichungssystem über dem Körper mit P Elementen, aus denen man die Lösung modulo P^s zusammensetzen kann. Dieser erste Schritt ist offensichtlich völlig unabhängig vom Element x, dessen Logarithmus wir suchen; er kann für eine gegebene Basis a und Primzahl p ein für allemal im voraus durchgeführt werden.

Im zweiten Schritt betrachtet man für zufällig gewählte Exponenten y die Elemente $a^y x \bmod p$, bis man eines findet, das nur durch Primzahlen aus der Faktorbasis teilbar ist. Falls etwa

$$a^y x \bmod p = q_1^{f_1} \cdots q_s^{f_s}$$

ist, muß

$$\log_a x = f_1 \log_a q_1 + \cdots + f_s \log_a q_s - y \bmod m$$

sein.

§5: Elliptische Kurven

Wie wir in den vorangegangen Paragraphen gesehen haben, kommen Kryptoverfahren, die auf diskreten Logarithmen beruhen, ohne vorher bekannte Schlüssel aus, so daß sie gerade für sichere Kommunikation im Internet sehr attraktiv sind. Problematisch sind allerdings die großen Zahlen, mit denen man hier arbeiten muß: Im Augenblick 1024 Bit, ab 2005 sogar 2048. Für einen PC ist das zwar nicht sonderlich problematisch, für den WAP-Browser eines Mobiltelephons aber ist es ein fast unüberwindliches Hindernis. Auch beim elektronischen Bargeld, egal ob auf der Basis von RSA oder in der hier nicht behandelten Variante mit diskreten Logarithmen sind die langen Schlüssel ein Problem, da die Bank alle Seriennummern bereits eingelöster „Banknoten“ speichern muß.

Zum Glück lassen sich diskrete Logarithmen nicht nur für die Multiplikation ganzer Zahlern modulo einer Primzahl p definieren, sondern in jeder Situation mit einer Rechenoperation hat, die sich iterieren läßt.

Beispielsweise könnten wir eine $n \times n$-Matrix A über einem endlichen Körper hernehmen und die Umkehrfunktion von $x \mapsto A^x$ betrachten; das diskrete Logarithmenproblem bestünde also darin, zu einer Matrix B eine ganze Zahl $x \ge 0$ zu finden, so daß $B = A^x$ ist. Das wäre allerdings nicht sonderlich sinnvoll, denn der Aufwand bei der Berechnung von A^x steigt mit wachsendem n stark an, wohingegen das diskrete Logarithmenproblem über die JORDAN-Zerlegung von A und B leicht auf das im Grundkörper zurückgeführt werden kann – wenn nicht gar der nilpotente Anteil noch zusätzliche Hinweise liefert.

Für kryptographische Anwendungen ideal wäre eine Operation, die nicht sehr viel aufwendiger ist als die Multiplikation modulo p, für die aber die Berechnung des darauf beruhenden diskreten Logarithmus deutlich komplexer ist. Solche Operationen liefert die algebraische Geometrie; der einfachste Fall (und auch der einzige, der bislang in der Praxis eine gewisse Rolle spielt) ist der der *elliptischen Kurven.*

Elliptische Kurven haben nur wenig mit Ellipsen zu tun; der historische Zusammenhang, der ihnen den Namen gab, besteht darin, daß elliptische Kurven über den komplexen Zahlen durch Funktionen parametrisiert werden können, die bei der Berechnung der Länge von Ellipsenbögen eine Rolle spielen. Für die Kryptologie, wo es um die Verschlüsselung endlicher Nachrichten geht, interessieren natürlich keine elliptischen Kurven über den komplexen Zahlen, sondern solche über endlichen Körpern. Trotzdem ist es vielleicht ganz nützlich, sich im folgenden der größeren Anschaulichkeit werden unter k zunächst den Körper der reellen Zahlen vorzustellen.

Eine affine elliptische Kurve über einem Körper k ist eine nichtleere Teilmenge der Ebenen k^2, die durch das Verschwinden eines Polynoms F vom Grad drei gegeben ist, d.h.

$$E^* = \left\{(x,y) \in k^2 \;\middle|\; F(x,y) = 0\right\}$$

mit

$$F(X,Y) = aX^3+bX^2Y+cXY^2+dY^3+eX^2+fXY+gY^2+hX+iY+j\,.$$

Von F verlangen wir, daß die partiellen Ableitungen F_X und F_Y und F selbst keine gemeinsame Nullstelle (x,y) haben, weder in k^2 noch in K^2 für irgendeinen Erweiterungskörper K von k. Geometrisch bedeutet diese Bedingung, daß es in jedem Punkt (egal ob seine Koordinaten in k liegen oder in einem Erweiterungskörper) genau eine Tangente gibt; die Kurve kann sich daher nicht selbst überkreuzen und auch keine „Spitzen“ haben.

Genau wie bei den klassischen diskreten Logarithmen sind auch bei ihrer Übertragung auf elliptische Kurven vor allem zwei Klassen von endlichen Körpern wichtig: Körper von Zweipotenzordnung und die Körper $\mathbb{F}_p$. Da elliptische Kurven über Körpern von Zwei- oder Dreipotenzordnung etwas schwieriger zu behandeln sind als solche über den Körpern $\mathbb{F}_p$ mit $p \geq 5$, wollen wir uns im folgenden auf letztere beschränken. Zumindest in den nächsten Abschnitten ist es allerdings immer noch nützlich, sich auch den Fall $k = \mathbb{R}$ vorzustellen.

Nachdem wir nun nur noch Körper betrachten, in denen man sowohl durch zwei als auch durch drei dividieren kann, läßt sich ohne allzugroße Schwierigkeiten mit Techniken wie quadratischer Ergänzung und deren kubischem Analogon ein neues Koordinatensystem für k^2 finden, in dem die Kurve durch eine Gleichung der Form

$$Y^2 = X^3 - aX - b$$

beschrieben werden kann, die sogenannte WEIERSTRASS-Normalform. Die Bedingung an die Ableitungen ist hier äquivalent dazu, daß das Polynom auf der rechten Seite keine mehrfachen Nullstellen hat; genauere Rechnung zeigt, daß dies äquivalent ist zum Nichtverschwinden der sogenannten Diskriminante $\Delta = 4a^3 - 27b^2$.

Schneiden wir diese Kurve mit der Geraden

$$Y = mX + c\,,$$

so muß die X-Koordinate jedes Schnittpunkts die Gleichung

$$(mx+c)^2 = x^3 - ax - b \quad\text{oder}\quad x^3 - m^2x^2 - (2mc+a)x - b - c^2$$

erfüllen, also eine echte kubische Gleichung. Diese hat, mit Vielfachheiten gezählt, genau drei Lösungen; da die y-Koordinate eines Schnittpunkts als $y = mx + c$ eindeutig durch die x-Koordinate bestimmt ist, gibt es also (mit Vielfachheiten gezählt) drei Schnittpunkte – wenn auch nicht notwendigerweise mit Koordianten aus dem Körper k.

Wenn allerdings die Gerade $Y = mX + b$ durch zwei Punkte (x_1, y_1) und (x_2, y_2) der elliptischen Kurve geht, deren Koordinaten in k liegen, sind x_1 und x_2 Nullstellen der kubischen Gleichung, die in k liegen; dividiert man sie durch $(X - x_1)(X - x_2)$ bleibt eine lineare Gleichung mit Koeffizienten aus k übrig, so daß auch der dritte Schnittpunkt in k liegt.

Damit folgt aber noch nicht, daß die Gerade durch zwei Kurvenpunkte mit Koordinaten in k ein dritter Kurvenpunkt mit Koordinaten in k ist: Zwar haben *fast* alle ebenen Geraden ein Gleichung der Form $Y = mX + b$, aber es gibt auch noch die zur y-Achse parallen Geraden, die nur durch Gleichungen der Form $x = d$ beschrieben werden können. Für diese wird die Gleichung für die y-Koordinate der Schnittpunkte zu

$$y^2 = d^3 - ad - b\,,$$

es gibt also nur zwei Schnittpunkte, denn die x-Koordinate ist nun ja auf d fixiert.

Das ist analog zur gewohnten Situation aus der klassischen Geometrie: Zwei ebene Geraden schneiden sich fast immer, aber es gibt eben auch den Ausnahmefall paralleler Geraden.

Zur Verallgemeinerung des diskreten Logarithmus auf elliptischen Kurven brauchen aber wir eine Vorschrift, die zwei Punkten einer elliptischen Kurve einen dritten zuordnet; das ist für zwei Punkte, die dieselbe x-Koordinate haben, unmöglich, da die Gerade durch diese beiden Punkte keinen weiteren Schnittpunkt mit der Kurve hat.

Um auch hier einen dritten Schnittpunkt zu bekommen, ergänzen wir die Kurve um einen sogenannten „unendlichfernen“ Punkt O, der auf allen zur y-Achse parallelen Geraden und natürlich auch auf der Kurve liegen soll, d.h. wir betrachten die Menge

$$E = \left\{(x, y) \in k^2 \;\middle|\; y^2 = x^3 - ax - b\right\} \cup \left\{O\right\}$$

als elliptische Kurve und vereinbaren, daß der zusätzliche Punkt O auf jeder der Geraden $x = c$ liegen soll.

(Auch wenn das Konzept eines unendlichfernen Punkts auf den ersten Blick seltsam und unpräzise erscheinen mag, sind diese Punkte doch in der Geometrie oftmals sehr nützlich; die projektive Geometrie liefert einen Formalismus, der diese Punkte ununterscheidbar von gewöhnlichen Punkten macht und präzisen Umgang mit ihnen erlaubt. Es lohnt sich aber nicht, wegen des einen unendlichfernen Punkts einer elliptischen Kurve diese ganze Theorie zu entwickeln.)

Damit können wir je zwei Punkten auf E einen dritten Zuordnen, allerdings ist das noch keine Verknüpfung, mit der wir vernünftig rechnen können: Bezeichnen wir diese Verknüpfung mit $\oplus$, so ist für die drei Schnittpunkte P, Q, R einer Geraden mit der elliptischen Kurve einerseits $P \oplus Q = R$, andererseits aber auch $P \oplus R = Q$, also auch

$$P \oplus (P \oplus R) = R\,.$$

Wenn wir von den üblichen Rechenregeln ausgehen, würde hieraus folgen, daß $P \oplus P$ für jeden Punkt P null wäre, womit wir ganz sicher keine Chance für kryptographisch brauchbare diskrete Logarithmen haben.

Wir definieren daher stattdessen eine andere Verknüpfung + durch die Vorschrift, daß O das Neutralelement sein soll, d.h.

$$O + P = P + O \quad \text{für alle } P \in E\,,$$

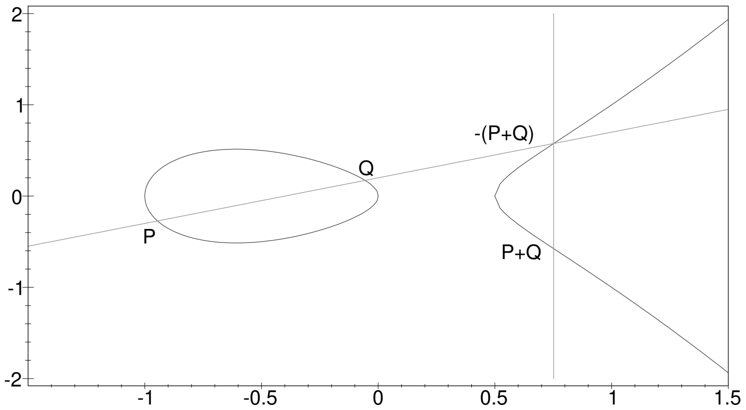
und daß $P + Q + R = O$ sein soll, wenn P, Q und R auf einer Geraden liegen. Damit ist insbesondere für zwei Punkte P und Q mit gleicher x-Koordinate

$$P + Q = P + Q + O = O\,,$$

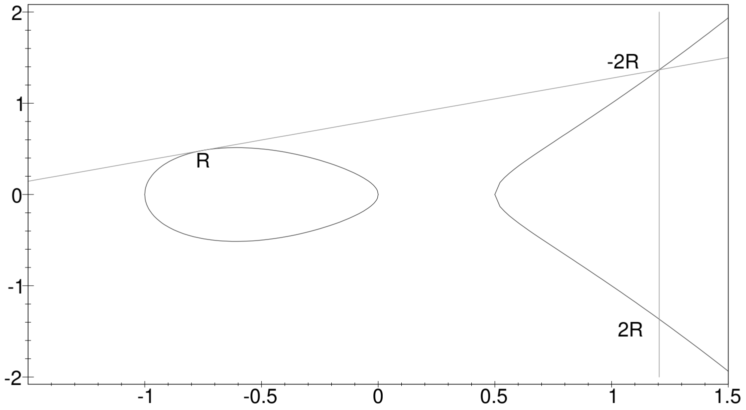
das inverse Element bezüglich von P bezüglich dieser Addition zum Punkt (x, y) ist also einfach $(x, -y)$.

Damit ist auch klar, wie man Summen berechnen kann: Zu zwei Punkten P, Q konstruiert man zunächst den dritten Schnittpunkt R der Geraden durch P und Q mit E und spiegelt diesen dann an der x-Achse. Im Fall $P = Q$ ist die Gerade durch P und Q natürlich die Tangente an E im Punkt P, von der wir bei der Definition einer elliptischen Kurve vorausgesetzt haben, daß sie immer existiert und eindeutig bestimmt ist.

Es ist klar, daß diese Addition kommutativ ist und O als neutrales Element hat; auch die Existenz inverser Elemente ist problemlos. Schwieriger ist das Assoziativgesetz: Es gilt zwar, aber der Beweis ist so aufwendig, daß wir hier darauf verzichten wollen.



Die Summe zweier verschiedener Punkte



Das Doppelte eines Punktes

Für eine natürliche Zahl n und einen Punkt $P \in E$ läßt sich nun leicht der Punkt

$$Q = nP = \underbrace{P + \cdots + P}_{n \text{ mal}}$$

definieren, und damit haben wir auch ein diskretes Logarithmenproblem: Man berechne umgekehrt zu gegebenem Q und P die natürliche Zahl n.

Zu zwei beliebigen Punkten $P, R \in E$ muß es natürlich kein $m \in \mathbb{Z}$ geben, so daß $R = mP$ ist; im allgemeinen wird es üuberhaupt keinen Punkt $P \in E$ geben, so daß alle anderen Punkte Vielfache von P sind; ein Kryptosystem auf der Basis elliptischer Kurve muß sich also beschränken auf die Teilmenge aller Vielfachen eines festen Punkts P, und genau wie es bei den klassischen Systemen auf der Grundlage diskreter Logarithmen wichtig ist, eine Basis a zu finden, die modulo p möglichst viele verschiedene Potenzen hat, ist es hier wichtig, einen Kurvenpunkt $P \in E$ zu finden mit möglichst vielen verschiedenen Vielfachen. Dies ist zum Glück immer möglich, denn es gibt immer Punkte P, deren Vielfache eine Gruppe bilden, deren Elementanzahl nicht viel kleiner ist als die Gesamtzahl der Kurvenpunkte.

Die Verallgemeinerung der vorgestellten Kryptoverfahren mit diskreten Logarithmen auf elliptische Kurven ist problemlos: Bei DIFFIE-HELLMANN etwa einigen sich die beiden Partner auf einen endlichen Körper k, eine elliptische Kurve E über k und einen Punkt $P \in E$ mit Koordinaten in k. Dann wählt A eine geheimzuhaltende natürliche Zahl x, die kleiner ist als die Anzahl verschiedener Vielfachen von P, und berechnet den Punkt $U = xP$, den er an B schickt; genauso wählt B eine Zahl y und schickt $V = yP$ an A. Beide sind dann in der Lage den Punkt

$$S = xV = yU = (xy)P$$

zu berechnen, dessen x-Koordinate als Schlüssel eines Kryptosystems benutzt werden kann. Man kann sich leicht überlegen, daß es tatsächlich ausreicht, anstelle der Punkte jeweils nur ihre x-Koordinaten zu übertragen.

Ähnlich lassen sich auch das ELGAMAL-Verfahren und das MASSEY-OMURA-Verfahren auf elliptische Kurven verallgemeinern.

Natürlich sind die Versionen mit elliptischen Kurven logisch und rechnerisch komplizierter als die klassischen Verfahren: Das n-fache eines Punktes P läßt sich zwar durch Additionen und Verdoppelungen im wesentlichen genau auf die gleiche Art berechnen wie eine Potenz durch

Multiplikationen und Quadratbildung, aber die Berechnung der Summe zweier Punkte sowie auch die Verdopplung eines Punktes erfordern deutlich mehr Aufwand als die bloße Multiplikation zweier Zahlen: Je nach der verwendeten Gleichung für die elliptische Kurve (die WEIERSTRASS-Gleichung ist nicht optimal) liegt der Aufwand etwa beim Fünf- bis Zehnfachen. Von daher sind Verfahren mit elliptischen Kurven deutlich aufwendiger als ihre klassischen Analoga.

Auch der Aufwand des Gegners steigt um denselben Faktor, falls er versucht, den diskreten Logarithmen auf einer elliptischen Kurve durch Probieren oder mit der *baby step – giant step* Methode zu finden: Beide Methoden lassen sich problemlos übertragen.

Anders sieht es aus beim Indexkalkül: Für diesen ist ganz wesentlich, daß die Potenzen von a modulo p nicht nur Elemente des Körpers $\mathbb{F}_p$ sind, sondern auch durch nichtnegative ganze Zahlen beschrieben werden können; diese wiederum können in ihre Primfaktoren zerlegt werden. Im Körper $\mathbb{F}_p$ selbst gibt es natürlich keine Primfaktorzerlegung: Jedes Element außer der Null ist invertierbar.

Auch auf einer elliptischen Kurve gibt es nichts, was den Primzahlen in $\mathbb{Z}$ entspräche; man kann zwar die Gleichung der elliptischen Kurve auch über den ganzen Zahlen betrachten und dort die Koordinaten der Punkte in ihre Primfaktoren zerlegen, aber bei der Addition von Punkten haben die Primfaktorzerlegungen der Koordinaten der Summanden und der Summe nichts miteinander zu tun.

Auch sonst ist es bislang noch niemandem gelungen, ein Verfahren zur Berechnung diskreter Logarithmen auf elliptischen Kurven zu finden, das in seinem Aufwand auch nur in die Nähe der Größenordnung des Indexkalküls käme; ein Gegner muß also mit deutlich schlechteren Verfahren arbeiten und sein Aufwand steigt um deutlich mehr als den Faktor fünf bis zehn, mit dem der Anwender des Verfahrens rechnen muß.

Aus diesem Grund kann man bei Verfahren mit elliptischen Kurven mit deutlich kleineren Körpern arbeiten als bei den klassischen Verfahren mit diskreten Logarithmen; die Richtlinien der Regulierungsbehörde für Telekommunikation und Post und des Bundesamts für Sicherheit in der Informationstechnik gehen derzeit davon aus, daß die Verfahren auch bis Ende 2006 noch sicher sind, wenn die Ordnung q des Punkts P mindestens 160 Bit hat; bis Ende 2008 werden 180 Bit gefordert, und dann wird auch gefordert, daß die Elementanzahl des Körpers mindestens 192 Bit hat; bis 2006 gibt es darüber noch keine Vorschrift, d.h. etwa 160 Bit reichen. (Darunter gibt es im allgemeinen keine Punkt hinreichend hoher Ordnung.) Bei klassischen diskreten Logarithmen werden bis Ende 2007 mindestens 1024 Bit verlangt, danach 1280, und schon jetzt sind 2048 empfohlen.

Bedenkt man nun, daß Aufwand sowohl für eine Potenzierung als auch für die Berechnung eines Vielfachen eines Kurvenpunkts jeweils proportional zur dritten Potenz der Ziffernzahl ist, sieht man klar die Vorteile elliptischer Kurven: Das Verhältnis zwischen 1024 und 160 ist 6,4 mit dritter Potenz knapp über 262. Auch wenn die Grundoperationen bei elliptischen Kurven um den Faktor fünf bis zehn aufwendiger sind, sind sie in Anbetracht der deutlich kleineren Zahlen also insgesamt doch noch um einen Faktor 25 bis 50 schneller. Hinzu kommt, daß man mit wesentlich kürzeren Nachrichtenblöcken arbeiten kann, was gerade beim elektronischen Geld ein großer Vorteil ist.

Mathematisch sind die Verfahren mit elliptischen Kurven deutlich komplizierter als die klassischen; die Theorie der elliptischen Kurven und der elliptischen Integrale, mit denen alles anfing, stammt aus der Zeit EULERS und war seither immer ein aktives Gebiet mathematischer Forschung. Deren Ergebnisse haben zum Teil auch Auswirkungen auf die kryptographische Anwendung elliptischer Verfahren, so daß die Auswahl einer kryptographisch guten elliptischen Kurve einiges an Kenntnissen über elliptische Kurven erfordert. Auch in den offiziellen Empfehlungen gibt es nicht nur die Vorschrift, daß die Ordnung der zyklischen Untergruppe mindestens 160 Bit haben muß; im Vorwort der 1999er-Empfehlungen steht auch, daß die wichtigste Änderung gegenüber dem Vorjahr darin bestehe, daß die Hauptordnung des Endomorphismenrings der elliptischen Kurve mindestens Klassenzahl zweihundert haben müsse. (Bis 1998 reichte einhundert, nach 1999 gab es bislang noch keine Änderung.)

Wer nicht versteht, was das bedeutet und vor allem auch, was es soll, sollte kein Kryptosystem auf der Basis elliptischer Kurven implementieren, ohne einen Fachmann zu konsultieren.

§6: Literatur

Diskrete Logarithmensysteme über endlichen Körper werden in denselben Büchern behandelt wie RSA; hierfür sei daher auf die Literaturangaben zum vorigen Kapitel verwiesen. Bücher, die auch Verfahren auf der Grundlage elliptischer und (teilweise) hyperelliptischer Kurven betrachten sind

NEAL KOBLITZ: *A Course in Number Theory and Cryptography,* Graduate Texts in Mathematics **114**, Springer ^{2}1994

und

NEAL KOBLITZ: *Algebraic Aspects of Cryptography,* Springer, 1998

Speziell mit elliptischen Kurven beschäftigt sich das Buch

ANNETTE WERNER: *Elliptische Kurven in der Kryptographie,* Springer, 2002,

in dem es vor allem um eine erste Einführung in die Theorie elliptischer Kurven unter dem Gesichtspunkt der Kryptographie geht. Beweise sind nicht immer vollständig, und anspruchsvollere Algorithmen werden nur sehr kurz behandelt. Das Buch

A.J. MENEZES: *Elliptic curve public key cryptosystems,* Kluwer, 1993

bringt eine auf dem Stand von 1993 ziemlich vollständige Diskussion aller kryptologisch relevanter Aspekte elliptischer Kurve, verweist aber für Beweise praktisch immer auf die Literatur.

Das wohl mathematischste Buch über Kryptographie mit elliptischen Kurven ist

IAN BLAKE, GADIEL SEROUSSI, NIGEL SMART: *Elliptic Curves in Cryptography,* London Mathematical Society Lecture Notes Series **265**, Cambridge University Press, 1999

Das bislang einzige Buch speziell über Kryptographie mit hyperelliptischen Kurven ist

ANDREAS ENGE: *Hyperelliptic Cryptosystems – Efficency and Subexponential Attacks,* Dissertation Universität Augsburg 2000, Books on Demand GmbH

Mit Implementierungsfragen beschäftigt sich

MICHAEL ROSING: *Implementing Elliptic Curve Cryptography,* Manning, 1999

Kapitel 7
SHA und DSA

§1: Kryptographisch sichere Hashfunktionen

Elektronische Unterschriften sind ungefähr so aufwendig wie die Verschlüsselung eines Dokuments mit einem asymmetrischen Verfahren. Solche Verfahren werden allerdings praktisch nie zur Verschlüsselung längerer Dokumente benutzt; man verwendet sie im allgemeinen nur, um damit einen Schlüssel für eines der erheblich schnelleren symmetrischen Verfahren zu übermitteln.

Entsprechend möchte man ein längeres Dokument nicht Block für Block unterschreiben, sondern möchte den Unterschriftsalgorithmus möglichst nur einmal anwenden. Dies wird dadurch möglich, daß man nicht die Nachricht selbst unterschreibt, sondern nur einen daraus berechneten Hashwert.

Im Falle einer der üblichen Hashfunktionen, die man etwa für Suchalgorithmen benutzt, bietet ein solcher Ansatz allerdings keinerlei Sicherheit, denn diese Hashfunktionen sind meist recht einfach aufgebaut, typischerweise eine Art von Quersumme, und damit läßt sich zu einem Text leicht ein zweiter finden, der zwar völlig anderen Inhalt, aber denselben Hashwert hat. Sobald jemand den Hashwert zum ersten Text unterschrieben hat, hätte er damit automatisch auch den zweiten unterschrieben, was natürlich jede Unterschrift wertlos macht.

Von einer kryptographisch brauchbaren Hashfunktion müssen wir daher fordern, daß es rechnerisch nicht mit vertretbarem Aufwand möglich ist, zu einem Hashwert einen Text zu konstruieren. Die Hashfunktion muß

also, genau wie ein symmetrisches Kryptoverfahren, mit Konfusion und Diffusion arbeiten, so daß möglichst jedes Bit des Texts jedes Bit des Hashwerts in einer schwer durchschaubaren Weise beeinflußt.

Dies legt es nahe, den Hashwert über ein symmetrisches Kryptoverfahren zu berechnen: Man nimmt etwa den ersten Block als Schlüssel, verschlüsselt damit den zweiten, nimmt das Ergebnis als Schlüssel zur Verschlüsselung des dritten und so weiter; die Verschlüsselung des letzten Blocks ist dann der Hashwert.

Ein solches Verfahren ist allerdings gleichzeitig zu aufwendig und zu unsicher: Da ohnehin jeder Block der Nachricht zum Endergebnis beiträgt, brauchen wir nicht so viele oder nicht so aufwendige Runden pro Block wie bei der Verschlüsselung eines einzelnen Blocks, dafür aber brauchen wir, wie der nächste Paragraph zeigen wird, bei gleicher Sicherheit eine doppelt so große Blocklänge wie bei einem Verschlüsselungsverfahren.

§2: Das Geburtstagsparadoxon

Der Grund dafür liegt am sogenannten „Geburtstagsparadoxon“: Angenommen, in einem Raum befinden sich n Personen. Wie groß ist die Wahrscheinlichkeit dafür, daß zwei davon am gleichen Tag Geburtstag haben?

Um diese Frage wirklich beantworten zu können, müßte man die (recht inhomogene) Verteilung der Geburtstage über das Jahr kennen; wir beschränken uns stattdessen auf ein grob vereinfachtes Modell ohne Schaltjahre mit 365 gleich wahrscheinlichen Geburtstagen. Dann ist die Wahrscheinlichkeit dafür, daß von n Personen keine zwei am gleichen Tag Geburtstage haben,

$$\prod_{k=0}^{n-1}\left(1-\frac{k}{365}\right) ,$$

denn für eine Person ist das überhaupt keine Bedingung, und jede weitere Person muß die Geburtstage der schon betrachteten Personen vermeiden. (Da der Faktor mit $k = 365$ verschwindet, wird die Wahrscheinlichkeit für $n > 365$ zu null, wie es nach dem DIRICHLETschen Schubfachprinzip auch sein muß.)

Nachrechnen ergibt für $n = 23$ ungefähr den Wert 0,4927; bei 23 Personen liegt also die Wahrscheinlichkeit für zwei gleiche Geburtstage bei 50,7%. Tatsächlich dürfte sie noch deutlich höher liegen, denn bei Geburtstagen ist die Annahme einer Gleichverteilung sicherlich falsch.

Bei einer guten Hashfunktion allerdings sollten die Hashwerte in sehr guter Näherung gleichverteilt sein; falls es N mögliche Hashwerte gibt, liegt die Wahrscheinlichkeit dafür, daß unter n Nachrichten zwei zum selben führen daher bei

$$p_n = \prod_{k=0}^{n-1}\left(1-\frac{k}{N}\right).$$

Da wir uns für große Werte von N interessieren, können wir davon ausgehen, daß

$$\left(1-\frac{1}{N}\right)^N \approx e \quad \text{und} \quad \left(1-\frac{1}{N}\right) \approx e^{-1/N}$$

ist; für nicht zu große Werte von k ist dann auch

$$\left(1-\frac{k}{N}\right) \approx e^{-k/N}$$

und für nicht zu große Werte von n gilt

$$p_n = \prod_{k=0}^{n-1}\left(1-\frac{k}{N}\right) \approx \prod_{k=0}^{n-1} e^{-k/N} = e^{-\frac{1}{N}\sum_{k=0}^{n-1} k} = e^{-\frac{n(n-1)}{2N}}.$$

Für $N = 365$ etwa ergibt dies den Näherungswert $p_{23} \approx 0{,}499998$ für den korrekten Wert 0,4927.

Wenn wir im Exponenten noch den Term $n(n-1)$ durch n^2 approximieren, können wir abschätzen, für welches n die Wahrscheinlichkeit p_n einen vorgegebenen Wert erreicht:

$$e^{-\frac{n^2}{2N}} = p \iff \frac{n^2}{2N} = -\ln p \iff n = \sqrt{-2N\ln p}\,.$$

Damit liegt p_n bei etwa 50%, falls $n \approx \sqrt{2N\ln 2} \approx 1{,}177\sqrt{N}$ ist; für $N = 365$ ergibt dies die immer noch recht gute Näherung 22,494.

Für $p = 1/1000$ ergibt sich $n \approx 3{,}717\sqrt{N}$, für $p = 999/1000$ entsprechend $n \approx 0{,}0447\sqrt{N}$. Die Wahrscheinlichkeit dafür, daß es unter n Nachrichten zwei mit demselben Hashwert gibt, wechselt also bei der Größenordnung $n \approx \sqrt{N}$ von sehr unwahrscheinlich zu sehr wahrscheinlich.

Damit ist klar, daß bei einem kryptographisch brauchbares Hashverfahren die Zahl N der möglichen Hashwerte so groß sein muß, daß die Erzeugung von $\sqrt{N}$ verschiedenen Nachrichten rechnerisch unmöglich ist. Ansonsten könnte nämlich ein Gegner zwei verschiedene Texte a und b erzeugen, von denen a so ist, daß ihn das Opfer unterschreibt, b aber für dieses außerst nachteilig ist. Dann kann er jeweils etwa $\sqrt{N}$ sinngleiche Modifikationen a_i und b_j erzeugen, indem er beispielsweile unabhängig voneinander an jedem Zeilenenden entweder ein Leerzeichen einfügt oder auch nicht, was bei z Zeilen bereits 2^z Varianten ergibt; dann kann er Kombinationen aus Leerzeichen und Rücktaste einfügen oder auch nicht, Tabulatoren durch Leerzeichen ersetzen oder auch nicht, und so weiter. Wie wir gerade gesehen haben, hat er dann eine gute Chance, daß eine a_i denselben Hashwert hat wie ein b_j; legt er seinem Opfer a_i zur Unterschrift vor, hat er damit auch ein unterschriebenes Dokument b_j.

Die Schlüssellänge K eines symmetrischen Kryptoverfahrens wird so gewählt, daß die 2^K Schlüssel nicht mit realistischem Aufwand durchprobiert werden können. Bei einem kryptographisch sicheren Hashverfahren muß dementsprechend die Länge L des Hashwerts so gewählt werden, daß niemand mit realistischem Aufwand $\sqrt{2^L} = 2^{L/2}$ Nachrichten erzeugen kann, d.h. bei gleichen Sicherheitsanforderungen muß ein Hashwert etwa doppelt so lang sein wie ein Schlüssel eines symmetrischen Kryptosystems. Da AES mit Schlüssellängen 128, 192 und 256 arbeitet, sollte man also entsprechend mit Hashwerten der Länge 256, 384 und 512 arbeiten; Hashwerte mit nur 128 Bit sind genau wie Kryptoverfahren mit 64 Bit-Schlüsseln heute nicht mehr sicher.

§3: Die Familie der SHA-Algorithmen

Da es mit kryptographisch sicheren Hashfunktionen deutlich weniger Erfahrung gibt als mit Verschlüsselung, liest sich die bisherige Geschichte solcher Funktionen eher enttäuschend: Zu den meisten Verfahren wurden über kurz oder lang Angriffe gefunden. Die Regulierungsbehörde für Telekommunikation und Post läßt derzeit zwei Algorithmen zu: RIPEMD-160 und SHA-1, die beide mit 160 Bit Hashwerten arbeiten. Ich möchte mich hier auf die SHA-1 beschränken, da es für diesen Algorithmus inzwischen auch Varianten mit längeren Hashwerten gibt.

Der *Secure Hash Algorithm* SHA wurde im Januar 1992 veröffentlicht und am 11. Mai 1993 als amerikanischer Standard FIPS 180 verkündet. Wegen einer (nie publizierten) „technischen Schwäche“ mußte auch dieser Algorithmus alsbald nachgebessert werden; am 11. Juli 1994 wurde die Modifikation SHA-1 als Nachfolgestandard FIPS 180-1 eingesetzt. Dessen Nachfolger FIPS 180-2 vom 1. August 2002 lies SHA-1 unangetastet und fügte lediglich drei neue, ähnlich aufgebaute Algorithmen SHA-256, SHA-384 und SHA-512 mit längeren Hashwerten dazu. (SHA-1 produziert 160 Bit lange Werte, die drei anderen Algorithmen ihrem Namen entsprechend.) Der aktuelle Standard ist unter

`http://csrc.nist.gov/CryptoToolkit/tkhash.html`

zu finden.

Die vier Algorithmen sind sehr ähnlich aufgebaut; SHA-1 und SHA-256 arbeiten mit 32-Bit-Wörtern und Blöcken der Länge 512, bei SHA-384 und SHA-512 wird mit den doppelten Längen gerechnet. Die Nachrichten, die SHA-1 und SHA-256 verarbeiten kann, müssen kürzer als 2^{64} Bit, also etwa zwei Millionen Terabyte, sein; für SHA-384 und SHA-512 liegt die Grenze bei 2^{128} Bit.

Diese Maximallänge spielt eine Rolle bei der Vorverarbeitung der Nachricht: Ist M die zu verarbeitende Nachricht, bestehend aus ℓ Bit, so wird am Ende ein Bit „1“ eingefügt, dann k Nullen und schließlich noch die Zahl ℓ als Block von 64 *bzw.* 128 Bit. Die Zahl k wird als kleinste nichtnegative ganze Zahl gewählt, für die $\ell+1+k+64$ durch 512 teilbar ist für SHA-1 und SHA-256 *bzw.* für die $\ell + 1 + k + 128$ durch 1024 teilbar ist für SHA-1 und SHA-256, so daß in jedem Fall die Länge der Nachricht ein ganzzahliges Vielfaches der Blocklänge ist.

Damit läßt sich die Nachricht nun aufteilen in Blöcke $M^{(1)}, M^{(2)}, \ldots$; jeder dieser Blöcke wiederum wird aufgeteilt in Wörter $M_0^{(i)}, \ldots, M_{15}^{(i)}$. (Man beachte, daß bei jedem der vier Algorithmen die Blocklänge gleich der 16-fachen Wortlänge ist.)

Jeder der Algorithmen startet mit seinem eigenen Anfangshashwert. Bei SHA-1 besteht dieser aus den willkürlich *(?)* festgelegten fünf (hexadezimal angegebenen) Wörtern

$$H_0^{(0)} = 67452301, \quad H_1^{(0)} = EFCDAB89, \quad H_2^{(0)} = 98BADCFE,$$
$$H_3^{(0)} = 10325476, \quad H_4^{(0)} = C3D2E1F0\,.$$

Bei den drei neueren Standards zeigt sich der Trend zu nachvollziehbaren mathematischen Definitionen: Hier gibt es jeweils acht Wörter $H_0^{(0)}, \ldots, H_7^{(0)}$, die über die hexadezimal geschriebenen gebrochenen Anteile der Quadratwurzeln von Primzahlen p definiert sind; es geht also um die hexadezimal geschriebenen ganzen Zahlen $\left[2^r\{\sqrt{p}\}\right]$, wobei $r = 32$ *bzw.* 64 die Wortlänge des jeweiligen Algorithmus ist. Für SHA-256 und SHA-512 nimmt man die erste bis achte Primzahl, für SHA-384 die neunte bis sechzehnte. In der nachstehenden Tabelle sind diese Werte zu finden, wobei für SHA-256 natürlich nur die linke Hälfte relevant ist.

Daneben gibt es noch Konstanten, die auch bei SHA-1 wieder willkürlich *(?)* festgelegt sind als

$$K_t = \begin{cases} 5A827999 & \text{für } \ 0 \le t \le 19 \\ 6ED9EBAL & \text{für } 20 \le t \le 39 \\ 8FLBBCDC & \text{für } 40 \le t \le 59 \\ CA62C1D6 & \text{für } 60 \le t \le 99 \end{cases},$$

und für die drei anderen Algorithmen gebenen sind durch die ersten Bits der gebrochenen Anteile der Kubikwurzeln der ersten Primzahlen. Für SHA-256 nimmt man die ersten 32 Bit und die ersten 64 Primzahlen, für SHA-384 und SHA-512 sind es die ersten 64 Bit und die ersten achtzig Primzahlen. In jedem Fall hat man also eine Folge von Wörtern

i	p_i	$\left[2^r\{\sqrt{p_i}\}\right]$
1	2	6A09E667 F3BCC908
2	3	BB67AE85 84CAA73B
3	5	3C6EF372 FE94F82B
4	7	A54FF53A 5F1D36F1
5	11	510E527F ADE682D1
6	13	9B05688C 2B3E6C1F
7	17	1F83D9AB FB41BD6B
8	19	5BE0CD19 137E2179
9	23	CBBB9D5D C1059ED8
10	29	629A292A 367CD507
11	31	9159015A 3070DD17
12	37	152FECD8 F70E5939
13	41	67332667 FFC00B31
14	43	8EB44A87 68581511
15	47	DB0C2E0D 64F98FA7
16	53	47B5481D BEFA4FA4

$K_0, K_1, \ldots$ bis K_{63} *bzw.* K_{79}. Die Hexadezimalentwicklungen der gebrochenen Anteile der Kubikwurzeln der ersten achtzig Primzahlen sind unten in der Tabelle zu finden.

Außer diesen Konstanten sind für die vier Algorithmen noch Funktionen definiert, die später im Konfusionsschritt eingesetzt werden; sie verknüpfen jeweils drei Wörter zu einem vierten, indem die angegebenen logischen Operationen bitweise angewendet werden.

Die Funktion

$$Ch(x, y, z) = (x \wedge y) \oplus (\neg x \wedge z)\,,$$

in der $\oplus$ für die Addition in $\mathbb{F}_2$ oder (äquivalent) das exklusive Oder steht, hat für wahres x den Wahrheitswert $y \oplus$ *falsch*, also denselben Wert wie y. Ist x dagegen falsch, erhalten wir *falsch* $\oplus\, z$, also den Wahrheitswert von z.

$Parity(x, y, z) = x \oplus y \oplus z$ ist einfach zu verstehen: Ein Ergebnisbit ist genau dann gleich eins, wenn von den entsprechenden Eingabebits eine ungerade Anzahl gleich eins ist.

i	p_i	$\left[2^r\{\sqrt[3]{p_i}\}\right]$	i	ip_i	$\left[2^r\{\sqrt[3]{p_i}\}\right]$
1	2	428A2F98 D728AE22	41	179	A2BFE8A14 CF10364
2	3	71374491 23EF65CD	42	181	A81A664BB C423001
3	5	B5C0FBCF EC4D3B2F	43	191	C24B8B70D 0F89791
4	7	E9B5DBA5 8189DBBC	44	193	C76C51A30 654BE30
5	11	3956C25B F348B538	45	197	D192E819D 6EF5218
6	13	59F111F1 B605D019	46	199	D69906245 565A910
7	17	923F82A4 AF194F9B	47	211	F40E35855 771202A
8	19	AB1C5ED5 DA6D8118	48	223	106AA0703 2BBD1B8
9	23	D807AA98 A3030242	49	227	19A4C116B 8D2D0C8
10	29	12835B01 45706FBE	50	229	1E376C085 141AB53
11	31	243185BE 4EE4B28C	51	233	2748774CD F8EEB99
12	37	550C7DC3 D5FFB4E2	52	239	34B0BCB5E 19B48A8
13	41	72BE5D74 F27B896F	53	241	391C0CB3C 5C95A63
14	43	80DEB1FE 3B1696B1	54	251	4ED8AA4AE 3418ACB
15	47	9BDC06A7 25C71235	55	257	5B9CCA4F7 763E373
16	53	C19BF174 CF692694	56	263	682E6FF3D 6B2B8A3
17	59	E49B69C1 9EF14AD2	57	269	748F82EE5 DEFB2FC
18	61	EFBE4786 384F25E3	58	271	78A5636F4 3172F60
19	67	FC19DC68 2B8CD5B5	59	277	84C87814A 1F0AB72
20	71	240CA1CC 77AC9C65	60	281	8CC702081 A6439EC
21	73	2DE92C6F 592B0275	61	283	90BEFFFA2 3631E28
22	79	4A7484AA 6EA6E483	62	293	A4506CEBD E82BDE9
23	83	5CB0A9DC BD41FBD4	63	307	BEF9A3F7B 2C67915
24	89	76F988DA 831153B5	64	311	C67178F2E 372532B
25	97	983E5152 EE66DFAB	65	313	CA273ECEE A26619C
26	101	A831C66 D2DB43210	66	317	D186B8C7 21C0C207
27	103	B00327C 898FB213F	67	331	EADA7DD6 CDE0EB1E
28	107	BF597FC 7BEEF0EE4	68	337	F57D4F7F EE6ED178
29	109	C6E00BF 33DA88FC2	69	347	6F067AA7 2176FBA
30	113	D5A7914 7930AA725	70	349	A637DC5 A2C898A6
31	127	6CA6351 E003826F	71	353	113F9804B EF90DAE
32	131	1429296 70A0E6E70	72	359	1B710B35 131C471B
33	137	27B70A8 546D22FFC	73	367	28DB77F5 23047D84
34	139	2E1B213 85C26C926	74	373	32CAAB7B 40C72493
35	149	4D2C6DF C5AC42AED	75	379	3C9EBE0A 15C9BEBC
36	151	53380D1 39D95B3DF	76	383	431D67C4 9C100D4C
37	157	650A735 48BAF63DE	77	389	4CC5D4BE CB3E42B6
38	163	766A0AB B3C77B2A8	78	397	597F299C FC657E2A
39	167	81C2C92 E47EDAEE6	79	401	5FCB6FAB 3AD6FAEC
40	173	92722C8 51482353B	80	409	6C44198C 4A475817

Auch mit der Mehrheitsfunktion $Maj(x, y, z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z)$ gibt es keine Schwierigkeiten: Falls genau zwei der Eingabebits gesetzt sind, ist genau eine der drei Klammern eins, also auch das Ergebnis. Sind alle drei Eingabebits gesetzt, erhalten wir eine Summe von drei Einsen, also ebenfalls eins. In allen anderen Fällen sind alle drei Summanden null, also auch das Ergebnis.

Auërdem werden zur Diffusion noch die zyklischen Verschiebungen ROTR nach links und ROTR nach rechts verwendet, sowie die entsprechenden nichtzyklischen Verschiebungen SHL und SHR. Die Operatoren verschieben um jeweils ein Bit; für weitere Verschiebungen sorgen ihre Potenzen.

Der eigentliche Algorithmus SHA-1 arbeitet dann folgendermaßen: Die Blöcke $M^{(1)}, M^{(2)}, \dots$ gehen nacheinander durch die folgenden Verarbeitungsschritte:

1. Setze

$$W_t = \begin{cases} M_t^{(i)} & \text{für } 0 \le t \le 15 \\ \text{ROTL}(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) & \text{für } 16 \le t \le 79 \end{cases}.$$

Dies ist offensichtlich ein Diffusionsschritt.

2. Initialisiere die fünf internen Variablen mit den fünf Hashwerten der vorigen Runde (für die erste Runde wurden die Hashwerte $H_j^{(0)}$ oben definiert):

$$a \to H_0^{(i-1)}, \quad b \to H_1^{(i-1)}, \quad c \to H_2^{(i-1)},$$
$$d \to H_3^{(i-1)}, \quad e \to H_4^{(i-1)}$$

3. Der Konfusionsschritt: Führe für $t = 0$ bis $t = 79$ die folgenden Anweisungen aus:

$$T \to \overset{5}{\text{ROTL}}(a) + f_t(b, c, d) + K_t + W_t, \quad e \to d, \quad d \to c,$$
$$c \to \overset{30}{\text{ROTL}}(b), \quad b \to a, \quad a \to T\,,$$

wobei $f_t = \begin{cases} Ch & \text{für } \quad \le t \le 19 \\ Maj & \text{für } 40 \le t \le 59 \\ Parity & \text{sonst} \end{cases}$ ist. (Die Konstanten K_t wurden oben definiert.)

4. Berechne den Hashwert der Runde:

$$H_0^{(i)} \to a + H_0^{(i-1)}, \quad H_1^{(i)} \to b + H_1^{(i-1)}, \quad H_2^{(i)} \to c + H_2^{(i-1)},$$
$$H_3^{(i)} \to d + H_3^{(i-1)}, \quad H_4^{(i)} \to e + H_4^{(i-1)}$$

Ergebnis des Algorithmus sind die aneinandergesetzten Hashwerte der letzten Runde.

Die Algorithmen SHA-256, SHA-382 und SHA-512 funktionieren nach demselben Schema mit anderen Konstanten und etwas modifizerten Berechnungen in den einzelnen Schritten.

§4: DSA

DSA steht für *Digital Signature Algorithm,* ein Algorithmus der im *Digital Signature Standard* DSS der USA festgelegt ist und neben RSA auch zu den von der Regulierungsbehörde für Telekommunikation und Post empfohlenen „Geeigneten Algorithmen“ zählt. Seine Sicherheit beruht auf diskreten Logarithmen, allerdings wird das klassische Verfahren dadurch modifiziert, daß die Sicherheit zwar auf dem diskreten Logarithmenproblem in einem großen Körper beruht, die Rechenoperationen bei der Anwendung des Algorithmus aber nur eine deutlich kleinere Untergruppe verwenden.

Für diese kleine Untergruppe wählt man eine Primzahl q, die nach den derzeitigen Empfehlungen der Regulierungsbehörde eine Länge von 160 Bit haben sollte – aus diesem Grund sind in den Empfehlungen auch nur 160 Bit Hashfunktionen betrachtet.

Zu dieser Primzahl q sucht man eine Primzahl $p \equiv 1 \bmod q$, für deren Länge die Regulierungsbehörde bis Ende 2007 mindestens 1024 Bit vorschreibt, bis Ende 2008 mindestens 1280. „Empfohlen“ sind auch hier 2048 Bit.

Als nächstes muß ein Element g gefunden werden, dessen Potenzen im Körper $\mathbb{F}_p$ eine Gruppe der Ordnung q bilden. Das ist einfach: Man starte mit irgendeinem Element $g_0 \in \mathbb{F}_p \setminus \{0\}$ und berechne seine $(p-1)/q$-te Potenz. Falls diese ungleich eins ist, muß sie wegen $g_0^{p-1} = 1$ die Ordnung q haben; andernfalls muß man ein neues g_0 betrachten.

Die so bestimmten Zahlen q, p und g werden veröffentlicht und können auch in einem ganzen Netzwerk global eingesetzt werden. Geheimer

Schlüssel jedes Teilnehmers ist eine Zahl x zwischen eins und $q-1$; der zugehörige öffentliche Schlüssel ist $y = g^x \bmod p$.

Unterschreiben lassen sich mit diesem Verfahren Nachrichtenblöcke m mit $0 \le m < q$, insbesondere also 160 Bit lange Hashwerte. Dazu wählt man für jede Nachricht eine Zufallszahl k mit $0 < k < q$ und berechnet

$$r = (g^k \bmod p) \bmod q\,.$$

Da q eine Primzahl ist, hat k ein multiplikatives Inverses modulo q; man kann also durch k dividieren und erhält eine Zahl s, für die

$$sk \equiv m + xr \bmod q$$

ist; die Unterschrift unter die Nachricht m besteht dann aus den beiden 160 Bit lagen Zahlen r und s. Sie kann nur berechnet werden von jemanden, der den geheimen Schlüssel x kennt.

Überprüfen kann die Unterschrift allerdings jeder: Ist t das multiplikative Inverse zu s modulo q, so ist

$$k \equiv tsk \equiv tm + xtr \bmod q\,,$$

also, da g die Ordnung q hat,

$$r \equiv g^k \equiv g^{tm} g^{xtr} \equiv g^{tm} y^{tr} \bmod p\,.$$

in dieser Gleichung sind die linke wie auch die rechte Seite *modulo q* öffentlich bekannt, die Gleichung kann also modulo q überprüft werden. Die Unterschrift wird anerkannt, wenn beide Seiten modulo q gleich sind.

Ein Angreifer müßte sich x aus y verschaffen, müßte also ein diskretes Logarithmenproblem modulo der großen Primzahl p lösen.

§5: DSA mit elliptischen Kurven

Wenn auch beim DSA die Ergebnisse nur 160 Bit lang sind, wird doch intern mit Zahlen gerechnet, die mindestens 1024 Bit haben. Damit bietet sich auch hier an, das Verfahren so zu modifizieren, daß es statt in der multiplikativen Gruppe eines Körpers in der Gruppe der Punkte einer elliptischen Kurve arbeitet. Dazu geht man folgendermaßen vor:

Man wählt eine elliptische Kurve E über einem Körper $\mathbb{F}_p$ und darauf einen Punkt P. Dessen Ordnung sein q mit einer von p verschiedenen Primzahl q. Dann kann man wie oben einen privaten Schlüssel $0 < x < q$ wählen und $Q = xP$ als öffentlichen Schlüssel publizieren; die Unterschrift besteht aus der x-Koordinate r des Punkts $R = kP$ und der daraus wie oben berechneten Zahl s.

Zum Überprüfen der Unterschrift braucht man wieder zunächst ein multiplikatives Inverses t von s; die Bedingung ist dann, daß der Punkt $tmP + trQ$ die x-Koordinate r hat. Die Rechnung geht genau wie oben, da auch die von P erzeugte Untergruppe der elliptischen Kurve die Ordnung q hat.

§6: Literatur

SHA-1 und DSA sind in den meisten hinreichend neuen Lehrbüchern der Kryptologie behandelt, z.B. in

Douglas R. Stinson: Cryptography – Theory and Practice, Chapman & Hall/CRC, ^{2}2002

Für die genaue Beschreibung von SHA-256 bis SHA-512 ist wohl die Originaldokumentation unter

`http://csrc.nist.gov/CryptoToolkit/tkhash.html`

am besten geeignet. DSA mit elliptischen Kurven ist beispielsweise in

Michael Rosing: *Implementing Elliptic Curve Cryptography,* Manning, 1999

dargestellt.

Kapitel 8
Kryptographische Protokolle

Bislang hatten wir Kryptologie nur im Zusammenhang mit Verschlüsselung und mit elektronischen Unterschriften betrachtet; in diesem Kapitel sollen einige darüber hinausgehende Aspekte betrachtet werden.

Eine wichtige solche Anwendung ist beispielsweise die Identitätsfestellung: Der Gebrauch einer Geldkarte (oder auch eines Mobiltelephons) hängt wesentlich davon ab, daß die Auszahlung *bzw.* die Gesprächsgebühren dem richtigen Konto belastet werden können.

Bei Geldkarten wird dies heute so realisiert, daß er Benutzer eine Geheimzahl eingeben muß, die in verschlüsselter Form im Magnetstreifen der Karte kodiert ist. Die Verschlüsselung hängt nicht nur ab von der Geheimzahl, sondern auch von den Kontodaten des Inhabers, so daß keine Bijektion zwischen den nur knapp zehn Tausend verschiedenen Geheimzahlen und irgendwelchen Feldern auf dem Magnetstreifen besteht. Zur Verschlüsselung wird ein Triple DES benutzt, dessen Schlüssel im gesamten System konstant ist und der daher sehr sorgfältig geheimgehalten werde muß; er ist nur den Computern der Clearingstellen des Systems bekannt.

Geldautomaten oder *point of sale* Terminals müssen daher sowohl die eingetippte Geheimzahl als auch die Information auf dem Magnetstreifen an so eine Clearingstelle übermitteln; dort wird beides verglichen und die Zahlung entweder autorisiert oder auch nicht.

Die dabei verwendeten Terminals funktionieren so, daß ein Händler die übermittelte Kundendaten nicht zu Gesicht bekommt; allerdings ist natürlich denkbar, daß ein betrügerischer Händler Geräte so manipuliert, daß sie sowohl eine Kopie des Magnetstreifens als auch die eingetippte Geheimzahl in einer ihm zugänglichen Weise speichern. Mit diesen Informationen kann er sich dann gegenüber Dritten als Karteninhaber ausgeben und beliebig über dessen Konto verfügen.

Besser wäre ein Verfahren, bei dem der Händler zwar sicher sein kann, daß er den legitimen Karteninhaber vor sich hat, bei dem aber auch ein betrügerischer Händler keine Informationen erhält, mit denen er sich anschließend als der Kunde ausgeben kann.

Einen möglichen Ansatz dazu liefern elektronische Unterschriften: Wenn der Händler dem Kunden einen zufällig gewählten Text zur elektronischen Unterschrift vorlegt

§x): Quadratische Reste

Definition: Ein Element $a \in \mathbb{F}_p$ heißt *quadratischer Rest* modulo p, wenn es ein $x \in \mathbb{F}_p$ gibt mit $x^2 = a$; andernfalls heißt a *quadratischer Nichtrest*. Entsprechend heißt $a \in \mathbb{Z}$ quadratischer Rest oder Nichtrest modulo p, wenn $a \bmod p \in \mathbb{F}_p$ diese Eigenschaft hat.

Offensichtlich sind Null und Eins modulo jeder Primzahl quadratische Reste; -1 ist beispielsweise quadratischer Nichtrest modulo drei, aber quadratischer Rest modulo fünf.

Da es modulo zwei nur zwei Elemente gibt, die beide quadratische Reste sind, **beschränken wir uns im folgenden auf ungerade Primzahlen** p. Für diese gilt:

Lemma: Unter den $p-1$ Elementen von $\mathbb{F}_p^\times = \mathbb{F}_p \setminus \{0\}$ gibt es jeweils $\frac{p-1}{2}$ quadratische Rest und quadratische Nichtreste.

Beweis: Wir betrachten die Abbildung

$$q\colon \begin{cases} \mathbb{F}_p^\times \to \mathbb{F}_p^\times \\ x \mapsto x^2 \end{cases}.$$

Offensichtlich sind für jedes $x \in \mathbb{F}_p^\times$ die Bilder $Q(x)$ und $Q(-x)$ gleich, und das sind auch schon die einzigen Elemente mit gleichem Bild; denn

da $\mathbb{F}_p$ ein Körper ist, hat das quadratische Polynom $X^2 - a$ für jedes $a \in \mathbb{F}_p$ höchstens zwei Nullstellen. Somit haben die $p-1$ Elemente von $\mathbb{F}_p^\times$ genau $\frac{p-1}{2}$ verschiedene Bilder, und das sind genau die quadratischen Reste.
■

Um genauere Aussagen über quadratische Reste und Nichtreste machen zu können, definieren wir das LEGENDRE-Symbol

$$\left(\frac{a}{p}\right) = \begin{cases} 0 & \text{falls } p|a \\ 1 & \text{falls } a \text{ quadratischer Rest modulo } p \\ -1 & \text{falls } a \text{ quadratischer Nichtrest modulo } p \end{cases} ,$$

wobei a entweder eine ganze Zahl oder ein Element von $\mathbb{F}_p$ sein kann; im letzeren Fall ist $\left(\frac{a}{p}\right) = 0$ genau dann, wenn $a = 0$ ist.

Lemma: Für $a, b \in \mathbb{F}_p$ oder $a, b \in \mathbb{Z}$ gilt

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

Beweis: Es genügt, $a, b \in \mathbb{F}_p$ zu betrachten. ab verschwindet genau dann, wenn a oder b verschwindet, und damit verschwindet die linke Seite der zu beweisenden Gleichung genau dann, wenn die rechte verschwindet. Im folgenden seien daher $a, b \in \mathbb{F}_p^\times$.

Falls $a = x^2$ und $b = y^2$ quadratische Reste sind, ist auch $ab = (xy)^2$ quadratischer Rest; für $\left(\frac{a}{p}\right) = \left(\frac{b}{p}\right) = 1$ stimmt die Formel also auch.

Ist $a = x^2$ quadratischer Rest, b aber nicht, so kann auch ab kein quadratischer Rest sein; denn wäre $ab = z^2$, so auch $\frac{b=(z}{x)^2}$. Entsprechend läßt sich argumentieren, wenn b quadratischer Rest ist, a aber nicht; falls rechts einmal +1 und einmal -1 steht, stimmt die Formel also auch.

Fehlt noch der Fall, daß a und b quadratische Nichtreste sind. Da es $\frac{p-1}{2}$ quadratische Reste und genauso viele quadratische Nichtreste gibt, sind die Elemente ax^2 mit $x \in \mathbb{F}_p^\times$ genau die quadratischen Nichtreste; insbesondere gibt es also ein $x \in \mathbb{F}_p^\times$ mit $b = ax^2$. Dann ist $ab = aax^2 = (ax)^2$ quadratischer Rest, womit auch dieser Fall bewiesen wäre.
■

Kriterium von Euler: Ist p ungerade Primzahl und kein Teiler von $a \in \mathbb{Z}$, so ist

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

Beweis: Nach dem kleinen Satz von FERMAT ist p Teiler von

$$a^{p-1} - 1 = \left(a^{(p-1)/2} + 1\right)\left(a^{(p-1)/2} - 1\right) ;$$

damit ist auf jeden Fall

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

Für einen quadratischen Rest $a \equiv x^2 \bmod p$ ist, wieder nach dem kleinen Satz von FERMAT,

$$a^{(p-1)/2} \equiv x^{p-1} \equiv 1 \bmod p ,$$

und da es einerseits genau $\frac{p-1}{2}$ von null verschiedene quadratische Reste modulo p gibt, andererseits das Polynom $X^{(p-1)/2} - 1$ in $\mathbb{F}_p$ höchstens $\frac{p-1}{2}$ verschiedene Nullstellen hat, gilt diese Formel *genau* für die quadratischen Reste. Damit muß für quadratische Nichtreste

$$a^{(p-1)/2} \equiv -1 \bmod p$$

sein, womit das Lemma bewiesen ist.
■

Das Kriterium von EULER ist nicht die effizienteste Art, das LEGENDRE-Symbol zu berechnen; schneller geht es zumindest für große p nach dem *quadratischen Reziprozitätsgesetz.* Dieses besagt, daß für zwei ungerade Primzahlen p, q gilt

$$\left(\frac{q}{p}\right)\left(\frac{p}{q}\right) = (-1)^{(\frac{p-1}{2})(\frac{q-1}{2})} = \begin{cases} +1 & \text{falls } p \equiv q \bmod 4 \\ -1 & \text{sonst} \end{cases} .$$

Erweitert man das LEGENDRE-Symbol zm sogenannten JACOBI-Symbol, bei dem der Nenner“ keine Primzahl mehr sein muß durch die Definition

$$\left(\frac{a}{\prod p_i}\right) = \prod \left(\frac{a}{p_i}\right)$$

und verallgemeinert das quadratischen Reziprozitätsgesetz so, daß es auch für das JACOBI-Symbol gilt, erhält man eine alternative Berechnungsweise für das JACOBI-Symbol, die im wesentlichen wie der EUKLIDische Algorithmus vorgeht und etwas effizienter ist als das Kriterium von EULER. Andererseits ist die Potenzierung modulo p, die wir zur Anwendung des EULERschen Kriteriums durchführen müssen, genau das, was wir in der Kryptographie ohnehin ständig anwenden, so daß dieser Rechenweg zumindest für kryptographische Zwecke völlig ausreicht.

Wir wollen daher auf einen Beweis des quadratischen Reziprozitätsgesetzes verzichten (Interessenten finden in fast jedem Lehrbuch der Zahlentheorie mindestens einen der zahlreichen Beweise), brauchen aber für das folgende die beiden sogenannten *Ergänzungssätze* dazu. Der erste folgt sofort aus dem EULERschen Kriterium und gibt an, wann -1 quadratischer Rest ist:

Erster Ergänzungssatz zum quadratischen Reziprozitätsgesetz: Für eine ungerade Primzahl p ist

$$\left(\frac{-1}{p}\right) = (-1)^{(p-1)/2} = \begin{cases} +1 & \text{falls } p \equiv 1 \bmod 4 \\ -1 & \text{falls } p \equiv 3 \bmod 4 \end{cases}.$$

Der zweite Ergänzungssatz gibt an, wann Zwei quadratischer Rest ist; dazu benötigen wir zunächst eine weitere, eher theoretisch interessante Methode zur Berechnung des LEGENDRE-Symbols:

Lemma von Gauß: p sei eine ungerade Primzahl, die a nicht teilt, und m sei die Anzahl der Zahlen

$$a, \quad 2a, \quad \dots, \quad \frac{p-1}{2}a\,,$$

deren Divisionsrest modulo p mindestens $\frac{p+1}{2}$ ist. Dann ist

$$\left(\frac{a}{p}\right) = (-1)^m\,.$$

Beweis: $\ell = \frac{p-1}{2} - m$ der Reste sind höchstens gleich $\frac{p-1}{2}$; diese Reste seien $a_1, \dots, a_\ell$, und die übrigen seien $b_1, \dots, b_m$. Dann ist

$$\prod_{i=1}^{\ell} a_i \prod_{j=1}^{m} b_j \equiv \prod_{k=1}^{(p-1)/2} (kn) = \left(\frac{p-1}{2}\right)! \cdot n^{(p-1)/2}\,.$$

Die Zahlen $p - b_j$ liegen allesamt zwischen eins und $\frac{p-1}{2}$ und sind paarweise verschieden; sie sind ebenfalls verschieden von den a_i, denn wäre $a_i = p - b_j$ für $a_i = \lambda a \bmod p$ und $b_j = \mu a \bmod p$, so wäre

$$\lambda a \equiv p - \mu a \equiv -\mu a \bmod p \Longrightarrow \lambda + \mu \equiv 0 \bmod p\,.$$

Das ist aber unmöglich, denn da λ und μ zwischen eins und $\frac{p-1}{2}$ liegen, liegt ihre Summe zwischen zwei und $p - 1$.

Damit sind die $\frac{p-1}{2}$ Zahlen a_i und $p - b_j$ allesamt verschieden; außerdem liegen sie alle zwischen eins und $\frac{p-1}{2}$, also sind es gerade die sämtlichen Zahlen zwischen eins und $\frac{p-1}{2}$. Damit ist

$$\begin{aligned} \left(\frac{p-1}{2}\right)! &= \prod_{i=1}^{\ell} a_i \prod_{j=1}^{m} (p - b_j) \equiv (-1)^m \prod_{i=1}^{\ell} a_i \prod_{j=1}^{m} b_j \\ &\equiv (-1)^m \left(\frac{p-1}{2}\right)!\ a^{(p-1)/2} \bmod p \end{aligned}$$

nach obiger Formel; Kürzen mit der modulo p invertierbaren Zahl $\left(\frac{p-1}{2}\right)!$ zeigt zusammen mit dem EULERschen Kriterium, daß

$$1 = (-1)^m a^{(p-1)/2} \equiv (-1)^m \left(\frac{a}{p}\right) \bmod p$$

und

$$\left(\frac{a}{p}\right) \equiv (-1)^m \bmod p$$

ist, wie behauptet. ■

Zweiter Ergänzungssatz zum quadratischen Reziprozitätsgesetz: Für eine ungerade Primzahl p ist

$$\left(\frac{2}{p}\right) = (-1)^{(p^2-1)/8} = \begin{cases} +1 & \text{für } p \equiv \pm 1 \bmod 8 \\ -1 & \text{für } p \equiv \pm 3 \bmod 8 \end{cases}.$$

Beweis: Die Zahlen $2, 2\cdot 2, \ldots \frac{p-1}{2}\cdot 2$ sind allesamt kleiner als p und somit ihre eigenen Divisionsreste modulo p. Eine solche Zahl $2r$ liegt genau dann zwischen $\frac{p+1}{2}$ und $p-1$, wenn

$$\frac{p}{2} < 2r < p \qquad \text{oder} \qquad \left[\frac{p}{4}\right] < r \le \left[\frac{p}{2}\right]$$

ist; es gibt also

$$m = \left[\frac{p}{2}\right] - \left[\frac{p}{4}\right]$$

solcher Zahlen. Wir schreiben $p = 8r+i$ mit $i \in \{1,3,5,7\}$; dann haben die einzelnen Terme in dieser Formel die folgenden Werte:

i	1	3	5	7
$\left[\frac{p}{2}\right]$	$4r$	$4r+1$	$4r+2$	$4r+3$
$\left[\frac{p}{4}\right]$	$2r$	$2r$	$2r+1$	$2r+1$
m	$2r$	$2r+1$	$2r+1$	$2r+2$

Also ist m gerade für $i = 1$ oder 7, d.h. für $p \equiv \pm 1 \bmod 8$, und ungerade sonst. Damit folgt die Behauptung aus dem Lemma von GAUSS. ∎

§y): Quadratwurzeln modulo einer Primzahl

Am einfachsten lassen sich Quadratwurzeln modulo zwei ziehen, denn modulo zwei ist jede Zahl ihre eigene Quadratwurzel. Im folgenden sei daher p eine ungerade Primzahl; wir zerlegen

$$p - 1 = 2^e \cdot q$$

in eine Zweierpotenz 2^e und eine ungerade Zahl q.

Da die multiplikative Gruppe $\mathbb{F}_p^\times$ modulo p zyklisch ist, gibt es ein Element $g \in \mathbb{F}_p^\times$, so daß

$$\mathbb{F}_p^\times = \{g, g^2, \ldots, g^{p-1} = 1\}$$

genau aus den Potenzen von g besteht. Damit ist $p-1$ der kleinste Exponent n mit der Eigenschaft, daß $g^n = 1$ ist, oder wie wir auch sagen werden, g hat die *Ordnung* $p-1$:

Definition: Ein Element $a \in G$ einer Gruppe g hat die Ordnung $n \in \mathbb{N}$, wenn n die kleinste natürliche Zahl ist mit der Eigenschaft $a^n = 1$.

Die Ordnung eines Elements g^r läßt sich leicht berechnen: Da g^{rn} genau dann zu eins wird, wenn $p-1$ den Exponenten rn teilt, ist rn für die Ordnung n das kleinste gemeinsame Vielfache von r und $p-1$. Das kleinste gemeinsame Vielfache ist bekanntlich gleich dem Produkt, diviert durch den größten gemeinsamen Teiler. Damit ist die Ordnung

$$n = \frac{p-1}{\operatorname{ggT}(r, p-1)}\,.$$

Speziell für die beiden Elemente

$$y = g^{2^e} \quad \text{und} \quad z = g^q$$

folgt, daß y die Ordnung q hat und z die Ordnung 2^e.

Da q und 2^e teilerfremd sind, gibt es nach dem erweiterten EUKLIDischen Algorithmus ganze Zahlen u, v, so daß

$$2^e u + qv = 1$$

ist. Hierbei muß v offensichtlich eine ungerade Zahl sein, denn sonst wäre die Summe auf der linken Seite eine gerade Zahl.

Damit ist

$$g = g^{qu+2^e v} = g^{qu} g^{2^e v} = y^u z^v$$

und entsprechend für jedes r

$$g^r = y^{ur} z^{vr}\,.$$

Da es bei Potenzen von y nur auf den Exponenten modulo q ankommt und bei solchen von z nur auf den Exponenten modulo 2^e, heißt dies, daß sich jedes Element von $\mathbb{F}_p^\times$ in der Form

$$a = y^\alpha z^\beta \quad \text{mit} \quad 0 \le \alpha < q \quad \text{und} \quad 0 \le \beta < 2^e$$

schreiben läßt.

$a = g^r$ ist genau dann ein quadratischer Rest, wenn r eine gerade Zahl ist; die beiden Quadratwurzeln sind dann $\pm r^{r/2}$. Falls wir nur a kennen, ist dies allerdings nicht sonderlich nützlich zur Berechnung dieser Wurzeln, denn erstens kennen wir im allgemeinen das Element g nicht explizit, und zweitens kennen wir auch den Exponenten r nicht. Ersteres wäre kein großes Problem, denn es gibt effiziente probabilistische Algorithmen, um sich mögliche Werte für g zu verschaffen, letzteres aber ist ein diskretes Logarithmenproblem modulo p, also nur dann effizient lösbar, wenn die Primzahl p kryptographisch völlig uninteressant ist.

Auch der etwas komplizierteren (und bislang ebenfalls nicht explizit bekannten) Form $a = y^\alpha z^\beta$ können wir ansehen, wann a quadratischer Rest ist: Da v eine ungerade Zahl ist, ist r genau dann gerade, wenn auch vr gerade ist, und das wiederum ist äquivalent dazu, daß $\beta = vr \bmod 2^e$ eine gerade Zahl ist.

$a^q = y^{\alpha q} z^{betaq} = z^{\beta q}$ ist eine Potenz von z; es gibt also eine ganze Zahl k zwischen null und $2^e - 1$, so daß $a^q z^k = 1$ ist, nämlich

$$k = -\beta q \bmod 2^e \in \{0, 1, 2, \ldots, 2^e - 1\} ,$$

und auch diese Zahl ist genau dann gerade, wenn a quadratischer Rest ist. Mit dieser Zahl k sind dann

$$x = \pm a^{(q+1)/2} z^{k/2}$$

die beiden Quadratwurzeln des quadratischen Rests a, denn

$$x^2 = a^{q+1} z^k = a(a^q z^k) = a .$$

Sobald wir den Wert $z^{k/2}$ kennen, können wir also die Quadratwurzeln von a bestimmen, und wir wir gleich sehen werden, läßt sich dieser Wert erheblich schneller berechnen als ein diskreter Logarithmus.

Zumindest für die „Hälfte“ aller Primzahlen haben wir damit überhaupt kein Problem: Eine ungerade Zahl hat bei Division durch vier offensichtlich entweder Rest eins oder Rest drei; wir betrachten zunächst den Fall, daß $p \equiv 3 \bmod 4$. (Solche Primzahlen werden in der Kryptologie gelegentlich als BLUMsche Primzahlen bezeichnet.)

Ist $p \equiv 3 \bmod 4$, so ist $p - 1 \equiv 2 \bmod 4$, d.h. $p - 1$ ist zwar durch zwei, nicht aber durch vier teilbar. Mithin ist

$$p - 1 = 2q \quad \text{mit einer ungeraden Zahl } q ,$$

d.h. der oben definierte Exponent e ist eins. Damit kommen für k nur die Werte $k = 0$ und $k = 1$ in Frage, und für einen quadratischen Rest a muß $k = 0$ sein. Dann sind sowohl z^k als auch $z^{k/2}$ gleich eins, also sind die beiden Wurzeln aus a einfach

$$x_{1/2} = \pm a^{(q+1)/2} = \pm a^{(\frac{p-1}{2}+1)/2} = \pm a^{(p+1)/4} ,$$

was sich leicht berechnen läßt. Die Richtigkeit dieser Formel läßt sich auch leicht direkt nachprüfen, denn

$$x_{1/2}^2 = a^{(p+1)/2} = a \cdot a^{(p-1)/2} = a \cdot \left(\frac{a}{p}\right)$$

nach der EULERschen Formel. Ist also a ein quadratischer Rest, so ist $x_{1/2}^2 = a$; wenden man die Formel fälschlicherweise auf einen quadratischen Nichtrest an, ist $x_{1/2}^2 = -a$.

Ist $p \equiv 1 \bmod 4$, so ist entweder $p \equiv 1 \bmod 8$ oder $p \equiv 5 \bmod 8$. Zumindest im letzteren Fall können wir wieder eine explizite Formel für die Quadratwurzeln aufstellen: In diesem Fall ist $p - 1 \equiv 4 \bmod 8$, d.h. $p - 1$ ist zwar durch vier, nicht aber durch acht teilbar. Damit ist

$$p - 1 = 4q = 2^2 q \quad \text{mit einer ungeraden Zahl } q ,$$

d.h. $e = 2$. Damit kann k nun die vier Werte $0, 1, 2, 3$ annehmen; für quadratische Reste gibt es die beiden Möglichkeiten $k = 0$ und $k = 2$.

Im Fall $k = 0$ können wir wie oben argumentieren: Dann ist

$$x_{1/2} = \pm a^{(q+1)/2} = \pm a^{(\frac{p-1}{4}+1)/2} = \pm a^{(p+3)/8} .$$

Für $k = 2$ sind die Wurzeln $\pm a^{(q+1)/2} z$, wobei z wie oben definiert und nicht explizit bekannt ist. z ist nach Definition q-te Potenz eines primitiven Elements, und das gilt offenbar für jedes Element, dessen Ordnung gleich 2^e ist. (Hier ist natürlich $e = 2$, aber das folgende Argument gilt für jedes e.)

Wenn wir die q-te Potenz *irgendeines* Elements aus $\mathbb{F}_p^\times$ berechnen, erhalten wir ein Element, dessen Ordnung eine Zweierpotenz ist, die 2^e teilt. Falls sie nicht gleich 2^e ist, muß das Element Quadrat eines anderen sein; die Elemente von Zweipotenzordnung, die genaue Ordnung 2^e haben, sind daher genau die quadratischen Nichtreste. Einen solchen können wir uns verschaffen, wenn wir irgendeinen quadratischen Nichtrest modulo p kennen: Da q ungerade ist, ist dann auch dessen q-te Potenz quadratischer Nichtrest, und wegen $p - 1 = 2^e q$ muß diese Potenz Zweipotenzordnung haben. Um z zu finden, reicht es also, *irgendeinen* quadratischen Nichtrest modulo p zu finden.

Letzteres ist im Fall $p \equiv 5 \bmod 8$ einfach: Hier ist zwei nach dem zweiten Ergänzungssatz zum quadratischen Reziprozitätsgesetz quadratischer Nichtrest; daher können wir

$$z = 2^q = 2^{(p-1)/4}$$

setzen. Für $k = 2$ ist somit

$$x_{1/2} = \pm a^{(p+3)/8} \cdot 2^{(p-1)/4} .$$

Um das Ganze wirklich explizit zu machen, müssen wir nun nur noch die beiden Fälle $k = 0$ und $k = 2$ algorithmisch voneinander unterscheiden, aber das ist einfach: Wir berechnen zunächst

$$w = a^{(p+3)/8} ;$$

falls $u^2 = a$ ist, sind $x_{1/2} = \pm w$ die beiden Quadratwurzeln. Andernfalls multiplizieren wir w mit $2^{(p-1)/4}$; falls das Quadrat dieses Elements von $\mathbb{F}_p$ gleich a ist, sind die Quadratwurzeln $\pm 2^{(p-1)/4} w$; andernfalls ist a quadratischer Nichtrest.

Auch dies läßt sich wieder direkt nachrechnen:

$$w^4 = a^{(p+3)/4} = a^2 \cdot a^{(p-1)/2} = a^2 \left(\frac{a}{p}\right) = a^2$$

nach EULER, falls a quadratischer Rest modulo p ist. Damit ist

$$w^2 = \pm a .$$

Falls $w^2 = a$, sind die beiden Wurzeln $\pm w$; andernfalls ist $w^2 = -a$. Da zwei quadratischer Nichtrest ist, ist nach EULER $2^{(p-1)/2} = -1$, also hat $w \cdot 2^{(p-1)/4}$ das Quadrat a.

Bleibt noch der Fall, daß $p \equiv 1 \bmod 8$. Dies ist der schwerste und allgemeinste Fall, denn während $p \equiv 3 \bmod 4$ äquivalent ist zu $e = 1$ und $p \equiv 5 \bmod 8$ zu $e = 2$ kann e hier jeden Wert größer oder gleich drei annehmen. Damit ist auch die Anzahl 2^e der möglichen Werte von k nicht mehr nicht mehr beschränkt; die Suche nach dem richtigen Exponenten k wir also aufwendiger.

Auch z selbst ist im allgemeinen Fall schwerer zu finden: Während wir für $p \equiv 5 \bmod 8$ wissen, daß zwei ein quadratischer Nichtrest ist, gibt es für $p \equiv 1 \bmod 8$ keine entsprechende Wahl; hier ist $\left(\frac{2}{p}\right) = 1$ und wir wissen nur, daß der kleinste quadratische Rest *irgendeine* Zahl zwischen eins und $1 + \sqrt{p}$ ist. Bei kryptographisch interessanten Primzahlen ist dies ein viel zu großes Intervall zum systematischen Durchsuchen, selbst wenn die Primzahlen bei kryptographischen Protokollen, die sich im Sekundenbereich abspielen, deutlich kleiner sein kann als etwa bei elektronischen Unterschriften.

Leider gibt es keinen effizienten deterministischen Algorithmus zur Bestimmung eines quadratischen Nichtrests modulo einer beliebigen Primzahl; da wir aber wissen, daß die Hälfte aller Restklassen modulo p quadratische Nichtreste sind, kann man in der Praxis leicht welche finden, indem man einfach Zufallszahlen erzeugt und beispielsweise nach der EULERschen Formel das LEGENDRE-Symbol ausrechnet. Die Wahrscheinlichkeit mehr als zehn Versuche zu brauchen liegt dann bei 1 : 1024, die für mehr als zwanzig Versuche ist kleiner als eins zu einer Million, *usw.*

Der folgende Algorithmus von SHANKS funktioniert für beliebige ungerade Primzahlen p; für $p \equiv 3 \bmod 4$ und $p \equiv 5 \bmod 8$ ist es aber natürlich effizienter, die obigen Varianten zu benutzen.

p sei also eine beliebige ungerade Primzahl, und $a \in \mathbb{F}_p^\times$ sei ein quadratischer Rest; der Algorithmus bestimmt unter dieser Voraussetzung ein Element $x \in \mathbb{F}_p^\times$ mit der Eigenschaft, daß x und $-x$ Quadrat a haben.

Indem wir $p - 1$ so lange wie möglich durch zwei dividieren (oder die hinteren Bits betrachten) bestimmen wir zunächst die Zerlegung $p - 1 = 2^e q$ mit einer ungeraden Zahl q.

Im *ersten Schritt* wird dann ein quadratischer Nichtrest modulo p bestimmt, indem wir so lange Zufallszahlen $n \bmod p$ erzeugen, bis

$\left(\frac{n}{p}\right) = -1$ ist. Dann berecnen wir $z = n^q \bmod p$ und wissen, daß diese Restklasse genau die Ordnung 2^e hat.

Im *zweiten Schritt* werden einige Variablen initialisiert; wir setzen

$$y \to z, \quad r \to e, \quad u \to a^{(q-1)/2}, \quad b \to au^2, \quad x \to au\,,$$

wobei alle Berechnungen modulo p durchgeführt werden. Danach ist

$$ab = x^2, y^{2^{r-1}} = -1 \quad \text{und} \quad b^{2^{r-1}} = 1\,,$$

denn $ab = a^2u^2 = x^2$, und die beiden hinteren Gleichungen bedeuten nach EULER einfach, daß $y = z$ quadratischer Nichtrest ist, a aber (nach Voraussetzung) quadratischer Rest.

Diese drei Gleichungen werden als Schleifeninvarianten durch den gesamten Algorithmus beibehalten; für $b = 1$ besagen sie, daß x eine Quadratwurzel aus a ist.

Im *dritten Schritt* testen wir daher, ob $b = 1$ ist; falls ja, endet der Algorithmus mit den Lösungen $\pm x$. Andernfalls suchen wir die kleinste natürliche Zahl m, für die $b^{2^m} = 1$ ist; die dritte Schleifeninvariante zeigt, daß es ein solches m gibt und $m \le r - 1$ ist.

Im *vierten Schritt* setzen wir

$$t \to y^{2^{r-m-1}}, \quad y \to t^2, \quad r \to m, \quad x \to xt, \quad b \to by$$

und gehen zurück zum dritten Schritt.

Die obigen Schleifeninvarianten gelten auch wieder nach den Zuweisungen im vierten Schritt: Für $ab = x^2$ kommt das einfach daher, daß das neue x gleich dem alten mal t ist, wohingegen das neue b aus dem Altgen durch Multiplikation mit $y = t^2$ entsteht. Die Gleichung $y^{2^r} = -1$ gilt weiterhin, weil das neue y die 2^{r-m}-te Potenz des alten ist, so daß seine m-te Potenz gleich dem alten Wert von y^{2^r} ist; da das neue r gleich m ist, folgt die Behauptung. Daß auch die dritte Schleifeninvariante erhalten bleibt, folgt aus der Gültigkeit der zweiten.

Kapitel 9
Spekulationen über künftige Entwicklungen

Auch ohne Spekulation ist klar, daß der Wettlauf zwischen Kryptographie und Kryptanalyse weitergehen wird, daß die Schlüssellängen sowohl der symmetrischen als auch der asymmetrischen Kryptoverfahren weiterhin ansteigen müssen, und daß immer wieder neue Ideen sowohl zu besseren Kryptoverfahren als zu effizienterer Kryptanalyse bekannter Verfahren führen werden.

Daneben gab es in den letzten Jahren eine Reihe von Entwicklungen, die möglicherweise unser gesamtes Bild der Kryptologie verändern könnten, auch wenn sie teilweise noch sehr spekulativ sind. Diese Ansätze sollen hier im Schlußkapitel kurz vorgestellt werden.

§1: Quantenkryptographie

Die Quantenkryptographie ist von den drei hier behandelten Themen das am wenigsten spekulative; in ersten Versuchen wurde sie bereits erfolgreich getestet.

Von allen Kryptosystemen, die wir bislang kennengelernt haben, hat nur der *one time pad* beweisbare absolute Sicherheit. Das liegt nicht nur daran, daß wir nur relativ wenige Kryptosysteme kennen: Wenn man fordert, daß ein Gegner aus dem Kryptogramm *keinerlei* Information ziehen kann außer der Tatsache, daß eine Nachricht einer bestimmten Länge versandt wurde, zeigt ein Satz von SHANNON (s. etwa **[W]**, §7.3), daß ein in diesem Sinne sicheres Kryptosystem notwendigerweise mit einem Schlüssel arbeiten muß, der *mindestens* so lang ist wie die zu verschlüsselnde Nachricht; er kann also nicht kürzer sein als beim *one time pad.*

Perfekte Sicherheit von Kryptosystemen erfordert somit einen sehr hohen Aufwand an Schlüsselinformation; hinzu kommt, daß der Schlüsselaustausch ohnehin ein großes Problem aller symmetrischer Kryptoverfahren ist: Jede Schlüsselvereinbarung erfordert entweder ein persönliches Treffen der Beteiligten oder einen vertrauenswürdigen Boten oder aber ein anderes Kryptosystem, von dessen Sicherheit dann *jede* künftige Kommunikation abhängt.

Asymmetrische Kryptoverfahren werden zwar genau dazu verwendet, sie sind aber wertlos, wenn wir *perfekte* Sicherheit wollen: Die Sicherheit aller derzeit gebräuchlicher asymmetrischer Kryptoverfahren beruht nur auf *erfahrungsgemäß* schwer lösbaren Problemen; ein BAYESscher Gegner könnte jedes *public key* System sofort brechen. Im nächsten Paragraphen werden wir im übrigen Computer kennenlernen, die dies auch können, und von denen wir nur *annehmen* können, daß sie bislang und in den nächsten paar Jahren von niemandem realisiert werden können.

Die Quantenkryptographie gestattet es zwei räumlich getrennten Partnern, einen Schlüssel (beispielsweise für den *one time pad*) so zu vereinbaren, daß ein Gegner mit einer beliebig nahe bei eins liegenden vorgegebenen Wahrscheinlichkeit kein einziges Bit an Information erhalten kann.

a) Informationsübertragung mit einzelnen Photonen

Die Grundidee des Verfahrens besteht darin, die Bits durch einzelne Photonen zu kodieren. Genau wie einen ganzen Lichtstrahl kann man auch ein einzelnes Photon *polarisieren,* d.h. man kann seine Schwingungsebene festlegen und beispielsweise vereinbaren, daß eine horizontale Schwingungsebene für eine Eins und eine vertikale für eine Null stehen soll. Der Empfänger stellt dann sein Polarisationsfilter horizontal; falls er dahinter ein Photon mißt, wurde eine Null gesendet, ansonsten eine Eins.

Praktisch werden die Photonen meist approximiert durch sehr kurze Lichtblitze, die mit hoher Wahrscheinlichkeit nicht mehr als ein Photon

enthalten, weil sie beispielsweise so kurz sind, daß sie nur mir Wahrscheinlichkeit $1/10$ überhaupt ein Photon enthalten. Die Wahrscheinlichkeit für zwei oder mehr Photonen in einem nichtleeren Lichtblitz liegt dann bei etwa 6%, was tolerierbar ist.

Da drehbare Polarisationsfilter nur langsam auf eine neue Richtung eingestellt werden können, verwendet man in der Praxis andere Methoden: Beispielsweise läßt sich der POCKELS-Effekt aus der nichtlinearen Optik ausnutzen, wonach gewisse anisotrope Kristalle beim Anlegen einer Spannung ihren Brechungsindex ändern, oder aber man verwendet (da die Spannung beim POCKELS-Effekt typischerweise in der Größenordnung einiger hundert Volt liegen muß, was sich auch nicht so schnell schalten läßt) für jede gewünschte Polarisationsrichtung eine eigene Laserdiode oder sonstige geeeignete Lichtquelle mit dahinterstehendem Polarisationsfilter und vereinigt anschließend die Strahlengänge. Auf diese Weise lassen sich Lichtblitze mit einer Frequenz von etwa 1 MHz erzeugen.

Da nicht jeder Puls ein Photon enthält und auch während der Übertragung Photonen verloren gehen, muß der Empfänger zunächst wissen, *ob* ein Photon angekommen ist; außerdem muß er dann dessen Polarisationsrichtung bestimmen. Dazu läßt er das Photon durch einen doppelbrechenden Kristall (z.B. einen Kalkspat) gehen; je nachdem ob es parallel oder senkrecht zu dessen optischer Achse polarisiert ist, verläßt es den Kristall an einer von zwei wohldefinierten Stellen, hinter denen Photomultiplikatoren und Detektoren stehen, so daß für jede der beiden Polarisationsrichtungen ein Detektor anspricht. Natürlich müssen Sender und Empfänger synchronisiert werden; dies geschieht beispielsweise dadurch, daß eine festgelegte Zeitspanne vor jedem Puls ein heller Lichtblitz gesendet wird, der garantiert ankommt.

Entsprechende Apparaturen wurden experimentell getestet, erstmals im Oktober 1989 über eine Entfernung von damals nur 32 cm. Inzwischen ist man bei Glasfaserkabeln einer Länge von knapp 50 km angelangt, sowohl bei aufgerollter Glasfaser im Labor als auch bei „echten“ Glasfaserverbindungen wie etwa der der Swiss Telekom von Genf nach Nyon (22,8 km) unter dem Genfer See. Für eine Vernetzung innerhalb einer Stadt ist Quantenkryptographie also bereits heute praktikabel. Entsprechende Strecken über Tausende von Kilometern erscheinen nach derzeitigem Stand der Technik unwahrscheinlich: In **[ZGHMT]** wird die maximal erreichbare Distanz auf etwa 100 km geschätzt. Der Grund ist vor allem die Absorption in Glasfaserkabeln: Bei den für Telekommunikation typischen Wellenlängen um 1300 und 1550 nm hat man eine Dämpfung von 0,35 beziehungsweise 0,2 dB/km, so daß nach knapp 30 beziehungsweise 50 km nur noch ein Zehntel der abgeschickten Photonen übriggeblieben ist. Für Photonen mit nur 800 nm Wellenlänge, die sich einfacher detektieren lassen, beträgt die Dämpfung sogar 2 dB/km, man hat also schon nach fünf Kilometern neun von zehn Photonen verloren.

Verglichen mit anderen Kryptosystemen, die auch für die drahtlose Übermittlung von Nachrichten benutzt werden können, ist die Notwendigkeit einer Glasfaserverbindung überhaupt ein Nachteil: Besser wäre es, wenn man die Photonen kabellos übertragen könnte.

Das Problem hierbei ist natürlich, daß es in der Luft, vor allem bei Tageslicht, bereits ziemlich viele Photonen gibt. Das ist allerdings nicht ganz so schlimm, wie es auf den ersten Blick aussieht, denn bei hinreichender Sorgfalt kann man den Ort, die Zeit und die Wellenlänge der übermittelten Photonen sehr genau festlegen. Bei einer Wellenlänge von etwa 770 nm (im infraroten Bereich also) ist die Atmosphäre auch so durchlässig, daß sich die Verlustrate in Grenzen hält.

Wissenschaftler der National Laboratories in Los Alomos berichteten in *Nature* im Juli 2000 **[BHLMNP]**, daß sie am 13. August 1999 von 9^{30} bis 11^{30} Uhr unter dem wolkenlosen blauen Himmel von New Mexico drahtlose Quantenkryptographie über eine Entfernung von 1,6 km erfolgreich testen konnten. Fernziel ist die magische Grenze von 30 km, die es erlauben würde, Photonen an einen Satelliten zu schicken. Ein solcher Satellit ist zwar von einem festen Punkt der Erde aus nur etwa acht Minuten pro Tag in direkter Sichtverbindung; diese Zeit würde aber, bei den angestrebten Datenraten, ausreichen, um einen Schlüssel zu vereinbaren, mit dem sich deutlich mehr als die typischerweise in einem Unternehmen innerhalb von 24 Stunden anfallenden Nachrichten vom und zum Satelliten verschlüsseln lassen.

b) Protokolle zur Quantenkryptographie

Natürlich wurde bei keinem der vorgestellten Experimente genau das oben skizzierte Verfahren getestet, denn so wie beschrieben bietet es keinerlei kryptographische Sicherheit: Ein Gegner könnte einfach alle Photonen abfangen und neue auf die Reise schicken; falls er diese genauso polarisiert, wie die abgefangenen, wird weder der Sender noch der Empfänger etwas bemerken.

Hier kommt nun die Quantennatur des Lichts ins Spiel: Licht kann nicht nur vertikal oder horizontal polarisiert werden, sondern in jeder beliebigen Richtung (oder auch zirkulär). Geht es dann durch ein auf eine andere Richtung eingestelltes Polarisationsfilter, nimmt die Intensität mit dem Cosinusquadrat der Winkeldifferenz zwischen den beiden Richtungen ab, bis sie bei aufeinander senkrecht stehenden Richtungen verschwindet. Das austretende Licht ist jeweils in Richtung des analysierenden Filters polarisiert.

Im Falle eines einzigen Photons kann die Intensität natürlich nicht abnehmen; das Photon wird entweder durchgelassen oder nicht. Falls die beiden Richtungen weder gleich sind noch aufeinander senkrecht stehen, ist allerdings nicht mehr deterministisch festgelegt, welcher der beiden Fälle eintritt: Bei einer Winkeldifferenz δ zwischen den beiden Polarisationsrichtungen wird das Photon mit Wahrscheinlichkeit $\cos^2\delta$ durchgelassen und mit Wahrscheinlichkeit $\sin^2\delta$ absorbiert. Speziell für $\delta = 45°$ wird sind beide Wahrscheinlichkeiten gleich ein halb.

Der erste Ansatz, dies für kryptographische Zwecke auszunutzen, stammt von CHARLES BENNET und GILLES BRASSARD aus dem Jahr 1984; er wird heute kurz als das BB84-Protokoll bezeichnet.

Bei diesem Protokoll entscheidet sich der Sender vor dem Senden eines jeden Photons zufällig für ein Referenzsystem, bestehend entweder aus den Richtungen 0° und 90°, oder aus den Richtungen 45° und 135°. Danach entscheidet er sich, wiederum zufällig, für einen der beiden Bitwerte 0 oder 1 und kodiert beispielsweise die Eins durch Polarisationsrichtung 0° oder 45°, die Null durch 90° oder 135°. Dazu braucht er entweder zwei POCKELS-Zellen oder vier Lichtquellen mit entsprechend eingestellten Polarisationsfiltern dahinter.

Praktisch würden die Zufallsentscheidungen beispielsweise durch Digitalisieren von weißem Rauschen erfolgen oder aber indem man polarisierte Photonen durch ein um 45° gedrehtes Filter schießt und mißt, welche transmittiert werden.

Auch der Empfänger wählt für jedes Bit zufällig eines der beiden Referenzsysteme; falls er dasselbe System gewählt hat wie der Absender, kann er das von diesem abgeschickte Bit messen, andernfalls erhält er ein Zufallsergebnis, was im Mittel in der Hälfte aller Fälle vorkommen wird.

Die folgende Tabelle faßt die verschiedenen Möglichkeiten für Sender und Empfänger noch einmal zusammen:

gesendet wird:	0°	0°	45°	45°	90°	90°	135°	135°
empfangen mit:	+	×	+	×	+	×	+	×
Meßergebnis:	0°	**?**	**?**	45°	90°	**?**	**?**	135°

Hier bedeutet „+“, daß der Empfänger seinen Doppelspat so orientiert, daß er 0° und 90°-Polarisierung messen kann, während „×“ entsprechend für das um 45° gedrehte Bezugssystem steht. Ein „**?**“ in der letzten Zeile soll besagen, daß hier das Meßergebnis nur vom Zufall abhängt und somit keine sinnvolle Information enthält.

Um festzustellen, welche Hälfte der gemessenen Bits sinnvolle Information enthält, informiert der Empfänger den Sender anschließend über einen gewöhnlichen, nicht abhörsicheren Kanal, welche Photonen angekommen sind und mit welchem Referenzsystem er sie gemessen hat. Der Sender teilt ihm dazu jeweils mit, ob dies die richtige Wahl war oder nicht. Dieser Austausch soll im folgenden kurz als die „öffentliche“ Diskussion bezeichnet werden.

Für die Photonen, bei denen beide mit demselben Referenzsystem gearbeitet haben, notiert sich der Sender den gesendeten und der Empfänger den gemessenen Bitwert; bis auf allfällige Übertragungsfehler sollten diese somit beide dieselbe Bitfolge notieren. Der Lauscher, der nachträglich zwar das korrekte Referenzsystem erfahren hat, nicht aber den übertragenen Bitwert, kann zumindest aus der „öffentlichen“ Diskussion nichts darüber erfahren. Seine sonstigen Möglichkeiten sollen weiter unten erörtert werden.

Zunächst aber soll hier noch das 1992 von BENNET vorgeschlagene logisch und technisch etwas einfachere B92-Protokoll vorgestellt werden. Hier verwendet der Sender nur die beiden Polarisationsrichtungen 0° für Null und 45° für Eins; der Empfänger nur 135° für Null und 90° für Eins. Insbesondere braucht der Empfänger also keinen doppelbrechenden Kristall mehr mit zwei Detektoren, sondern nur noch eine POCKELS-Zelle mit *einem* nachgeschalteten Detektor. Die möglichen Meßergebnisse sind in folgender Tabelle zusammengefaßt:

gesendet wird	0°	0°	45°	45°
gemessen wird mit	90°	135°	90°	135°
Photon wird detektiert?	nein	**?**	**?**	nein

Das „**?**“ in der letzten Zeile soll dabei bedeuten, daß der Empfänger mit gleicher Wahrscheinlichkeit ein Photon mißt oder auch nicht.

Es gibt also nur zwei Kombinationen, bei denen der Empfänger überhaupt ein Photon finden kann: Falls der Empfänger mit 0° gesendet und der Empfänger mit 135° gemessen hat, oder wenn der Empfänger mit 45° gesendet und der Empfänger mit 90° gemessen hat. Falls ein Photon gemessen wird, weiß der Empfänger also, mit welcher Polarisationsrichtung gesendet wurde und kann je nachdem eine Null oder Eins notieren.

Im anschließenden „öffentlichen“ Dialog muß er dem Sender dann nur mitteilen, in welchen Positionen er Photonen gemessen hat, so daß auch der sich die Bitwerte dafür notieren kann. Man beachte, daß bei diesem Protokoll nur jedes vierte übermittelte Photon zu einem Bitwert führt.

c) Elimination der gegnerischen Information

Sowohl beim BB84- als auch beim B92-Protokoll kennen Sender und Empfänger am Ende eine Bitfolge, über die ein Gegner zumindest durch Abhören der „öffentlichen“ Diskussion nichts in Erfahrung bringen kann. Er hat aber natürlich auch noch die Möglichkeit, sich in den anfänglichen Photonenaustausch einzuschalten.

Hier wird die Quantentheorie wichtig: Ein Lauscher hat keine Möglichkeit, ein Photon zu messen, ohne es zu verändern. Falls er beispielsweise ein mit 0°-Polarisierung gesendetes Photon mit einem Doppelspat im schrägen Referenzsystem mißt, hat er anschließend mit gleicher Wahrscheinlichkeit ein mit 45° oder 135° polarisiertes Photon; er weiß aber nicht, ob das Photon wirklich mit dieser Polarisierung ankam, oder ob es eine der beiden Polarisierungsrichtungen 0° und 90° hatte.

Hier ist extrem wichtig, daß mit einzelnen Photonen gearbeitet wird: Falls zwei Photonen im selben Zustand übertragen werden, kann man sie durch einen Strahlteiler trennen und jedes in einem anderen Referenzsystem messen; beim BB84-Protokoll wird dann in der „öffentlichen“ Diskussion klar, welche der Messungen zum richtigen Bitwert führte, beim B92-Protokoll ist er in drei Viertel aller Fälle sofort klar, da eine Polarisationsrichtung von 90° im +-System nur gemessen werden kann, wenn das Photon im ×-System übertragen wurde, d.h. mit Polarisationsrichtung 45°. Entsprechend kann 135° im ×-System nur gemessen werden kann, wenn das Photon mit 0° polarisiert war. Lediglich bei der Kombination (0°, 45°) ist nicht klar, welches Bit der Sender übermitteln wollte.

Mit nur einem Photon kann der Gegner jedoch nie sicher sein, welchen Zustand das gesendete Photon hatte und muß sich überlegen, was er als nächster tun will. Er hat zwei Möglichkeiten:

Als erstes könnte er das Photon einfach verschwinden lassen. Das ist aus seiner Sicht nicht sinnvoll: Falls es ihm nur darum geht, die Kommunikation unmöglich zu machen, hat er einfachere Möglichkeiten. Er muß also wieder irgendein Photon senden.

Am sichersten wäre es, ein Photon zu senden, das genau dieselbe Polarisationsrichtung hat wie das abgefangene; in diesem Fall bliebe sein Lauschen unbemerkt. Diese Möglichkeit wird aber durch die Quantentheorie ausgeschlossen, da er den Zustand des Photons nicht vollständig bestimmen kann.

Am wenigsten verfälscht er beim BB84-Protokoll, wenn er ein Photon losschickt, das die von ihm gemessene Polarisationsrichtung hat: Falls er im richtigen Referenzsystem maß, bleibt sein Eingriff bei diesem Photon unbemerkt, andernfalls gibt es immerhin noch eine 50%-ige Wahrscheinlichkeit, daß der Empfänger, falls er im richtigen System mißt, den korrekten gesendeten Wert mißt. (Falls auch der Empfänger im falschen System mißt, wird das Bit nicht verwendet, so daß es auf seinen Wert nicht ankommt.) In etwa einem Viertel aller Fälle sorgt der Lauscher durch seinen Eingriff dafür, daß der Empfänger trotz gleichen Referenzsystems einen anderen Bitwert mißt als den ursprünglich gesendeten.

Beim B92-Protokoll kann ein Lauscher, der wie der Empfänger beim B84-Protokoll mit Doppelbrechung arbeitet, jedes zweite Bit messen: Falls er im +-System 90° mißt, muß das Photon mit 45° polarisiert gewesen sein, wenn er im ×-System 135° mißt, mit 45°. In diesem Fall kann er ein entsprechendes Ersatzphoton schicken. In der anderen Hälfte der Fälle muß er aber raten und verfälscht somit auch hier wieder insgesamt rund ein Viertel aller übertragener Bitwerte.

Sender und Empfänger einigen sich daher in der „öffentlichen“ Diskussion auf eine Zufallsstichprobe der notierten Bits und vergleichen diese; auf diese Weise können sie die Fehlerrate der Übertragung feststellen und durch Vergleich mit der (bekannten) Fehlerrate bei ungestörter Kommunikation abschätzen, wieviel Information der Gegner in Erfahrung bringen konnte.

Falls sie einen Abhörversuch feststellen, haben sie die Möglichkeit, die Kommunikation abzubrechen und zu einem späteren Zeitpunkt, eventuell nach genauer Untersuchung der Verbindungsstrecke, einen neuen Versuch zu wagen. Sie können aber auch versuchen, die vom Gegner gewonnene Information aus ihrer gemeinsamen Bitfolge zu eliminieren. Dies geschieht in drei Schritten:

Zunächst wird natürlich die Stichprobe aus der Bitfolge eliminiert: Diese Bits wurden in der „öffentlichen“ Diskussion verglichen und sind somit dem Gegner bekannt.

Im zweiten geht es darum, überhaupt eine gemeinsame Bitfolge zu bekommen: Durch den Vergleich einer Stichprobe kann schließlich nur die Fehler*rate* ermittelt werden, nicht aber die Stellen, an denen sich außerhalb der Stichprobe Fehler befinden. Dieser zweite Schritt ist unabhängig von der Aktivität eines Gegners notwendig, da auch bei einer nicht abgehörten Übertragung Photonen ihren Zustand ändern können, Detektoren gelegentlich auch nicht existente Photonen melden *usw.*

Die Grundidee ist dieselbe wir in der klassischen Kodierungstheorie: Bei jeder Übertragung von Information über einen Kanal gibt es Störungen, die zur Verfälschung eines Teils der Information führen. Um diese zu eliminieren, überträgt man zusätzlich zur eigentlichen Information noch zusätzliche Prüfbits, mit deren Hilfe man eine (vom verwendeten Code abhängige) Anzahl von Bitfehlern pro Codewort korrigieren kann.

Nehmen wir etwa an, die vereinbarte Bitfolge sei aus Sicht des des Senders der Vektor $\mathfrak{v}$, aus Sicht des Empfängers aber $\mathfrak{v}' = \mathfrak{v} + \mathfrak{u}$, wobei $\mathfrak{u}$ der Fehlervektor ist.

Falls sich $\mathfrak{v}$ als Folge von Worten aus einem fehlerkorrigierenden Code auffassen ließe, könnte der Empfänger einfach die Dekodierungsabbildung dieses Codes auf $\mathfrak{v}'$ anwenden, um $\mathfrak{v}$ zu erhalten. Da aber nur ein Bruchteil der Blitze zu Komponenten von $\mathfrak{v}$ führt, hat der Sender keine Möglichkeit, $\mathfrak{v}$ entsprechend zu präparieren. Er behilft sich daher mit einem Trick: Er wählt einen zufälligen Bitvektor und kodiert diesen mit einem fehlerkorrigierenden Code, der für die ermittelte Fehlerrate ausreicht; die Länge dieses Vektors sei so gewählt, daß der enstehende Codevektor $\mathfrak{w}$ dieselbe Länge wie $\mathfrak{v}$ hat. Sodann berechnet er $\mathfrak{v} + \mathfrak{w}$ und überträgt diesen Vektor über die ungesicherte Leitung – je nach deren Qualität eventuell wiederum gesichert durch einen fehlerkorrigierenden Code. Auch die Art der verwendeten Codes wird dem Empfänger über die ungesicherte Leitung mitgeteilt.

Der Empfänger kennt dann sowohl $\mathfrak{v} + \mathfrak{w}$ als auch $\mathfrak{v}' = \mathfrak{v} + \mathfrak{u}$, er kann also deren Differenz $\mathfrak{v} + \mathfrak{w} - \mathfrak{v}' = \mathfrak{w} - \mathfrak{u} = \mathfrak{w} + \mathfrak{u}$ berechnen. (Da wir mit Bitvektoren rechnen, gibt es keinen Unterschied zwischen plus und minus.) Das ist aber das mit dem Fehler $\mathfrak{u}$ gestörte Codewort $\mathfrak{w}$, die Dekodierungsfunktion des fehlerkorrigierenden Codes erlaubt also die Rekonstruktion von $\mathfrak{w}$. Damit ist dann aber auch $\mathfrak{u}$ berechenbar und somit auch $\mathfrak{v}$.

Der Lauscher weiß weniger über $\mathfrak{v}$ als der Empfänger; er kann zwar

eventuell auch etwas von der Fehlerkorrektur im zweiten Schritt profitieren, hat aber immer noch keine vollständige Information über $\mathfrak{v}$.

Dieses unterschiedliche Wissen über $\mathfrak{v}$ wird nun im dritten Schritt, der sogenannten *privacy amplification*, ausgenutzt, um $\mathfrak{v}$ auf einen Vektor $\mathfrak{z}$ zu verkürzen, über den der Lauscher mit hoher Wahrscheinlichkeit so gut wie nichts weiß. Die Längendifferenz zwischen $\mathfrak{v}$ und $\mathfrak{z}$ entspricht dabei der Information, die der Lauscher über $\mathfrak{v}$ gewonnen hat; diese kann leicht aus der Fehlerrate berechnet werden.

Zur Berechnung von $\mathfrak{z}$ einigen sich Sender und Empfänger wieder in „öffentlicher“ Diskussion, über die (zufällige) Auswahl einer Hashfunktion φ aus einer großen Anzahl vorher vereinbarter solcher Funktionen. Dies darf erst *nach* Übertragung der Photonen geschehen, denn wenn der Lauscher diese Funktion kennt, kann er seine Abhörstrategie darauf abstimmen und sie praktisch wirkungslos machen; insbesondere gibt es also immer ein kleines Restrisiko, daß der Lauscher durch vorheriges Erraten der richtigen Funktion durch deren Anwendung keine Information verliert.

Bei einem hinreichend großen Raum von Hashfunktionen ist dieses Restrisiko jedoch verschwindend klein, und eine genauere informationstheoretische Analyse zeigt, daß die Information, die der Lauscher über $\varphi(\mathfrak{v})$ hat, mit sehr hoher Wahrscheinlichkeit sehr nahe bei Null liegt. Die Behandlung der Einzelheiten dieser Analyse und des dahinterstehenden Begriffsapparats würde hier zu weit führen; interessierte Leser seien auf die Originalarbeit **[BBR]** verwiesen, vor allen den dortigen Paragraphen 4.2.

Zum Schluß sei nach angemerkt, daß auch die Sätze aus **[BBR]** noch nicht beweisen, daß Quantenkryptographie wirklich sicher ist – auch nicht bis auf das erwähnte Restrisiko. Wir sind nämlich immer nur davon ausgegangen, daß der Lauscher sich darauf beschränkt, einzelne Photonen zu messen und in geeigneter Weise zu ersetzen. Tatsächlich könnte er stattdessen auch irgendwelche Interferenzerscheinungen oder andere Daten über aus vielen Photonen zusammengesetzte Zustände messen und daraus Schlußfolgerungen ziehen.

Realistischerweise muß man zur Abschätzung der Sicherheit des Verfahrens dem Gegner erlauben, alle Messungen durchzuführen, die die Quantentheorie nicht ausdrücklich verbietet, darunter auch solche, an die bislang noch niemand gedacht hat.

Unter diesen Umständen erfordert die Analyse seiner Möglichkeiten erheblich tiefere Methoden aus der Quantentheorie, als die einfache Darstellung in diesem Seminar bieten kann. Mit solchen Methoden kann man dann allerdings beweisen, daß Quantenkryptographie auch gegenüber einem Gegner mit unbeschränkten Ressourcen im erwähnten Sinne sicher ist; siehe dazu etwa **[SP]**.

d) Literaturhinweise

Eine erste Übersicht gibt der Artikel

CHARLES H. BENNET, GILLES BRASSARD, ARTHUR K. EKERT: Quantenkryptographie, *Spektrum der Wissenschaft,* Dezember 1992, S. 96–104

Neuere Entwicklungen findet man in den Kapiteln über Quantenkryptographie der Bücher

MICHAEL BROOKS [Hrsg.]: *Quantum computing and communications,* Springer, 1999 *und*

COLIN P. WILLIAMS, SCOTT H. CLEARWATER: *Explorations in quantum computing,* Springer Telos, 1997

Das erste der beiden Bücher besteht aus ziemlich kurzen und wenig technischen Darstellungen, die sich vor allem an Laien wenden; im zweiten wird auch auf die Mathematik und die Physik hinter den Verfahren eingegangen. Außerdem enthält dieses Buch eine CD mit Mathematica *notebooks* zur Simulation von Quantenprozessen.

Hier im Text zitiert wurden

[BBR] CHARLSES BENNETT, GILLES BRASSARD, JEAN MARC ROBERT: Privacy amplification by public discussion, *SIAM J. Comp.* **17** (1988), 210–229
[BHLMNP] W.T. BUTTLER, R.J. HUGHES, S.K. LAMOREAUX, G.L. MORGAN, J.E. NORDHOLT, C.G. PETERSON: Daylight quantum key distribution over 1.6 km, *Phys. Rev. Letters* **84***,* 24 *(12. Juni 2000),* 5652–5255
[SP] PETER W. SHOR, JOHN PRESKILL: Simple proof of security of the BB84 quantum key distribution protocol, *Phys. Rev. Letters* **85**,2 *(10. Juli 2000),*441–444
[W] DOMINIC WELSH: *Codes und Kryptographie,* VCH Weinheim, 1991
[ZGHMT] H. ZBINDEN, N. GISIN, B. HUTTNER, A. MULLER, W. TITTEL: Practical Aspects of Quantum Cryptographic Key Distribution, *J. Cryptology* **13** (2000), S. 207–220

§ 2: Quantencomputer

Die Quantentheorie ist nicht nur in der Lage, Kryptographie sicherer zu machen, sie stellt potentiell auch eine ernste Bedrohung existierender Kryptoverfahren dar, die mit einem sogenannten Quantencomputer möglicherweise leicht gebrochen werden können. Besonders gefährdet sind die asymmetrischen oder *public key*-Verfahren, mit denen wir uns in den letzten beiden Kapiteln beschäftigten.

a) Quantenmechanische Grundlagen

Zum Verständnis eines Quantencomputers müssen wir uns etwas stärker mit dem formalen Apparat der Quantenmechanik befassen als dies für die Quantenkryptographie notwendig war.

Grundlage der Beschreibung eines quantenmechanischen Systems ist immer sein *Zustandsraum,* ein Vektorraum über dem Körper $\mathbb{C}$ der komplexen Zahlen mit einem HERMITEschen Skalarprodukt. Dieses Produkt wird in der Form $\langle u|v\rangle$ geschrieben, und von daher hat sich die DIRACsche Konvention eingebürgert, die Elemente von V in der Form $|v\rangle$ zu schreiben. Da $\langle u|v\rangle$ eine *bracket* ist, wird $|v\rangle$ allein meist kurz als ein *ket* bezeichnet.

Diese Schreibweise mag einem Mathematiker auf den ersten Blick verwirrend erscheinen; er sollte sich aber klarmachen, daß es hier nur um ein Symbol geht und sich an der Mathematik natürlich nichts ändert, egal ob man einen Vektor als $\mathfrak{v}$, $\vec{v}$ oder eben $|v\rangle$ schreibt.

Für Leser, die HERMITEsche Skalarprodukte nicht aus der Linearen Algebra kennen, sei hier die Definition kurz wiederholt: Grundsätzlich soll sich ein HERMITEsches Skalarprodukt genauso verhalten, wie das klassische Skalarprodukt im $\mathbb{R}^n$, jedoch gibt es damit das Problem, das nach der üblichen Definition etwa

$$\begin{pmatrix}1\\i\end{pmatrix}\cdot\begin{pmatrix}1\\i\end{pmatrix} = 1^2 + i^2 = 1 - 1 = 0$$

wäre; wenn wir das Produkt eines Vektors mit sich selbst in der üblichen Weise als Länge interpretieren, hätte also ein deutlich vom Nullvektor verschiedener Vektor die Länge null, was natürlich allen unseren Vorstellungen von Länge widerspricht.

Deshalb werden die Komponenten des zweiten Vektors komplex konjugiert; wir setzten also für Vektoren aus $\mathbb{C}^n$

$$\left\langle \begin{pmatrix}z_1\\\vdots\\z_n\end{pmatrix} \middle| \begin{pmatrix}w_1\\\vdots\\w_n\end{pmatrix} \right\rangle = z_1\overline{w}_1 + \cdots + z_n\overline{w}_n\ ,$$

so daß etwa

$$\left\langle \begin{pmatrix}1\\i\end{pmatrix} \middle| \begin{pmatrix}1\\i\end{pmatrix} \right\rangle = 1^2 + i\overline{i} = 2$$

ist. Dadurch geht natürlich die Symmetrie des Produkts verloren und auch die Linearität im zweiten Argument: Offensichtlich ist nun

$$\langle v|\lambda w\rangle = \overline{\lambda}\ \langle v|w\rangle\ .$$

Die formale Definition eines HERMITEschen Skalarprodukts ist

Definition: Ein HERMITEsches Skalarprodukt auf einem komplexen Vektorraum V ist eine Abbildung

$$\langle\cdot|\cdot\rangle : V\times V \to \mathbb{C};\qquad \big(|v\rangle\,,|w\rangle\big) \mapsto \langle v|w\rangle$$

mit folgenden Eigenschaften:

1. $\langle\lambda v_1 + \mu v_2|w\rangle = \lambda\,\langle v_1|w\rangle + \mu\,\langle v_2|w\rangle$
2. $\langle v|w\rangle = \overline{\langle w|v\rangle}$
3. $\langle v|v\rangle \geq 0$ und $\langle v|v\rangle = 0$ genau dann, wenn $|v\rangle$ der Nullvektor ist.

Die Vektoren aus V beschreiben die möglichen Zustände des Systems, wobei zueinander proportionale Vektoren denselben Zustand beschreiben; der tatsächliche Zustandsraum ist also der projektive Raum $\mathbb{P}(V)$. Da es somit auf die Länge des Zustandsvektors nicht ankommt, wird diese vielfach auf eins normiert.

Meßbare physikalische Größen werden durch HERMITEsche Operatoren $A\colon V \to V$ beschrieben, d.h. durch Operatoren, für die gilt

$$\langle Av|Aw\rangle = \langle v|w\rangle$$

für alle $|v\rangle\,,|w\rangle \in V$. Schreibt man den Operator A bezüglich einer Orthonormalbasis als Matrix, so ist das äquivalent zur Gleichung

$${}^tA = \overline{A}\ .$$

Ist das System vor der Messung im Zustand, der durch den Einheitsvektor $|v\rangle$ beschrieben wird, so ist sein Zustand nach der Messung ein Eigenvektor von A und der Meßwert ist der zugehörige Eigenwert.

Genau wie es zu einer symmetrischen reellen Matrix immer eine Orthonormalbasis des $\mathbb{R}^n$ aus Eigenvektoren der Matrix A gibt, gibt es zu einem HERMITEschen Operator immer eine Orthonormalbasis $\mathfrak{B}$ von V aus Eigenvektoren von A; falls die zugehörigen Eigenwerte alle verschieden sind, ist die Wahrscheinlichkeit für den Eigenwert λ als Meßergebnis gleich dem Betragsquadrat des Produkts von $|v\rangle$ und $|w\rangle$, wobei $|w\rangle$ der Eigenvektor der Länge eins zum Eigenwert λ ist.

Betrachten wir als Beispiel die Photonen aus dem vorigen Kapitel. Der Zustandsraum für ein Photon ist ein zweidimensionaler Vektorraum, in dem die Polarisationsrichtung φ etwa durch den Vektor $\binom{\cos\varphi}{\sin\varphi}$ repräsentiert wird; im Falle eines reellen Vektorraums wäre jeder Vektor proportional zu einem solchen Einheitsvektor. Daß wir komplexe Vektorräume betrachten, bedeutet, daß zum Zustand auch noch ein Phasenfaktor gehört, der für die vorgestellten Protokolle zur Quantenkryptographie aber keine Rolle spielt.

Die Messung der Polarisationsrichtung mit einem Polarisationsfilter mit Durchlaßrichtung ψ wird durch den Operator

$$A_\psi = \begin{pmatrix}\cos 2\psi & \sin 2\psi\\ \sin 2\psi & -\cos 2\psi\end{pmatrix} = \begin{pmatrix}\cos^2\psi - \sin^2\psi & 2\sin\psi\cos\psi\\ 2\sin\psi\cos\psi & \sin^2\psi - \cos^2\psi\end{pmatrix}$$

modelliert; seine Eigenwerte sind ±1 mit Eigenvektoren

$$|v_1\rangle = \begin{pmatrix}\cos\psi\\ \sin\psi\end{pmatrix} \quad\text{und}\quad |v_{-1}\rangle = \begin{pmatrix}-\sin\psi\\ \cos\psi\end{pmatrix}.$$

Bei einem in Richtung φ polarisierten Photon liegt also die Wahrscheinlichkeit für +1 beim Quadrat von

$$\begin{pmatrix}\cos\varphi\\ \sin\varphi\end{pmatrix}\begin{pmatrix}\cos\psi\\ \sin\psi\end{pmatrix} = \cos\varphi\cos\psi + \sin\varphi\sin\psi = \cos(\varphi-\psi)\,,$$

die für -1 entsprechend bei $\sin^2(\varphi-\psi)$.

b) Quantenregister und QBits

In einem klassischen Computer werden Bits dargestellt durch die beiden Zustände eines bistabilen Schaltelements wie etwa eines Flipflops oder durch eine Magnetisierungsrichtung oder etwas ähnliches. Unabhängig von der physikalisch-technischen Realisierung hat man immer ein Element, das sich stets in einem von zwei wohldefinierten Zuständen befindet; diese werden traditionell als 0 und 1 bezeichnet.

Angenommen, wir verwenden stattdessen ein Photon. Das ist beim heutigen Stand der Technologie zwar völlig unrealistisch, aber es ist wohl die anschaulichste Art, das Prinzip eines Quantencomputers zu verstehen; weiter unten werden wir auch realitätsnähere Ansätze diskutieren.

Zur Kodierung eines Bits können wir etwa vereinbaren, daß die horizontale Polarisation einer Null und die vertikale einer Eins entsprechen soll; mit einem Polarisationsfilter lassen sich Photonen in jedem der beiden Zustände unschwer produzieren.

Wir können aber auch anders vorgehen: Wir produzieren ein Photon mit Polarisationsrichtung 45° und lassen es so auf einen Doppelspat fallen, daß der eine der beiden austretenden Strahlen horizontal polarisiert ist, der andere vertikal.

Da nur ein Photon in den Kristall eintritt, kann auch nur eines austreten; indem wir zwei Detektoren in die beiden austretenden Strahlengänge stellen, können wir leicht feststellen, in welchem der beiden Strahlen sich das Photon befindet.

Dieses Aufstellen von Detektoren ist allerdings eine Messung, der Zustandsvektor des Photons wird also auf einen Eigenraum eines Operators projiziert. Was wir messen, ist daher nicht wirklich das aus dem Kristall austretende Photon. Dessen Zustand ist nach der experimentell sehr gut bestätigten Lehre der Quantentheorie, solange es keiner Messung unterworfen wird, eine Überlagerung der beiden möglichen Meßergebnisse: Bezeichnen wir die beiden möglichen Zustände *nach* der Messung mit $|0\rangle$ und $|1\rangle$, so befindet sich das Photon *vor* der Messung in einem Zustand, der durch den (auf Länge eins normierten) Vektor

$$\frac{\sqrt{2}}{2}\,|0\rangle + \frac{\sqrt{2}}{2}\,|1\rangle$$

beschrieben wird. Hier sind also die beiden Zustände $|0\rangle$ und $|1\rangle$ überlagert, und falls der Winkel 45° durch einen anderen ersetzt wird, lassen sich auch die Koeffizienten vor $|0\rangle$ und $|1\rangle$ beliebig verändern.

Ein solches quantenmechanisches System mit einem zweidimensionalen Zustandsraum bezeichnen wir als ein *QBit.* Im Gegensatz zu einem gewöhnlichen Bit, das nur die Werte 0 und 1 annehmen kann, sind für ein QBit also beliebige Werte der Form

$$\alpha\,|0\rangle + \beta\,|1\rangle \quad\text{mit}\quad \alpha,\beta\in\mathbb{C}$$

möglich. ($\alpha^2+\beta^2=1$ bei Normierung auf Länge eins)

Setzt man mehrere QBits zusammen, entsteht ein Quantenregister. Wichtig ist dabei, daß dieses Quantenregister ein einziges quantenmechanisches System ist, es ist also beispielsweise durchaus möglich, daß der Zustand „alle QBits sind $|0\rangle$“ mit dem Zustand „alle QBits sind $|1\rangle$“ überlagert ist. In diesem Fall wüßten wir zwar nicht im voraus, welches Ergebnis eine Messung eines QBits aus dem Register hätte, wir könnten uns aber sicher sein, daß nach der Messung des ersten QBits jede Messung eines weiteren QBits zum selben Ergebnis führte.

In DIRAC-Notation schreibt man

$$|\alpha_1\alpha_2\ldots\alpha_n\rangle$$

für den Zustand, in dem das i-te QBit den Wert $\alpha_i\in\{0,1\}$ hat; die 2^n Vektoren, die man auf diese Weise erhält, bilden offenbar eine Basis des Zustandsraums für das Quantenregister. (Mathematisch betrachtet ist dieser das Tensorprodukt $V_1 \otimes V_2 \otimes \cdots \otimes V_n$ der Zustandsräume V_i der einzelnen QBits.) Der Zustand eines Quantenregisters wird also beschrieben durch eine Linearkombination von Vektoren der obigen Form wie etwa

$$\frac{\sqrt{2}}{2}\,|00\ldots0\rangle + \frac{\sqrt{2}}{2}\,|11\ldots1\rangle$$

für das Beispiel aus dem vorigen Absatz.

c) Quantencomputer

Genau wie ein klassischer Computer den Inhalt klassischer Register manipuliert, manipuliert ein Quantencomputer den Inhalt von Quantenregistern.

Das fundamentale Grundgesetz der Quantenmechanik, die SCHRÖDINGER-Gleichung, sagt aus, wie sich deren Inhalt verändern kann: Ist $|\psi(t)\rangle$ der Zustandsvektor zur Zeit t und wird das System durch keine äußeren Einflüsse gestört, so ist

$$i\hbar\frac{\partial\,|\psi(t)\rangle}{\partial t} = H\,|\psi(t)\rangle\ ,$$

wobei H den HAMILTONschen Operator des Systems bezeichnet, jenen HERMITEschen Operator also, der die Gesamtenergie des Systems beschreibt, und $\hbar = h/2\pi \approx 1{,}054589 \times 10^{-34}$ Js das durch 2π dividierte PLANCKsche Wirkungsquantum.

Die SCHRÖDINGER-Gleichung ist ein System linearer Differentialgleichungen mit konstanten Koeffizienten; seine Lösung läßt sich mit Hilfe der Matrixexponentialfunktion sofort hinschreiben:

$$|\psi(t)\rangle = e^{-iHt/\hbar}\,|\psi(0)\rangle = U(t)\,|\psi(0)\rangle \quad \text{mit} \quad U(t) = e^{-iHt/\hbar}\ .$$

Dabei ist

$$U(t)\cdot\overline{U(t)}^T = e^{-iHt/\hbar}\cdot e^{i\overline{H}^T t/\hbar} = E$$

die Einheitsmatrix, denn $\overline{H}^T = H$ nach Definition eines HERMITEschen Operators. Somit ist der Operator $U(t)$, der die zeitliche Entwicklung des Systems beschreibt, unitär und insbesondere auch invertierbar.

Diese Invertierbarkeit ist ein großer Unterschied zu klassischen Computern: Die Addition $3 + 5 = 8$ etwa läßt sich nicht invertieren, denn das Ergebnis „8“ enthält keine Information mehr darüber, wie es zustandegekommen ist. Langfristig werden aber möglicherweise auch klassische Computer bei immer weitergehender Miniaturisierung der Bauelemente mit reversibler Logik rechnen müssen, denn die einzige Stelle beim Rechnen, bei der Energieverbrauch aus physikalischen Gründen nicht vermieden werden kann, ist die Vernichtung von Information.

Von den klassischen Logikoperationen, mit denen heutige Computer arbeiten, ist nur die Negation reversibel und natürlich auch unitär: Bezüglich der Basis $\{|0\rangle, |1\rangle\}$ des Zustandsraums eines Photons wird sie durch die Matrix

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

beschrieben. Konjunktion und Disjunktion sind aus demselben Grund wie die Addition nicht reversibel: Für eine reversible Logikoperation muß die Anzahl der Ausgangsbits gleich der der Eingangsbits sein. Ein Beispiel einer reversiblen Operation mit zwei Eingangsbits ist die kontrollierte Negation

$$(x, y) \mapsto \begin{cases} (x, \neg y) & \text{falls } x = 1 \\ (x, y) & \text{falls } x = 0 \end{cases} = (x, x \oplus y)\ ,$$

wobei $\oplus$ die Addition modulo 2 bezeichnet.

Bezüglich der Basis $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$ wird die kontrollierte Negation durch die unitäre Matrix

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

beschrieben, und sie ist auch quantenmechanisch realisierbar.

Wichtiger ist das TOFFOLI-Gate, das auf drei QBits operiert:

$$(x, y, z) \mapsto T(x, y, z) \underset{\text{def}}{=} \big((x, y, z \oplus (x \wedge y)\big)\ ,$$

denn mit seiner Hilfe lassen sich auf Kosten eines zusätzlichen Bits „und“ und „oder“ realisieren:

$$T(x, y, |0\rangle) = (x, y, x \wedge y) \text{ und } T(\neg x, \neg y, |1\rangle) = (\neg x, \neg y, x \vee y)\,.$$

In der Standardbasis des Zustandsraums für drei QBits vertauscht das TOFFOLI-Gate einfach die beiden Vektoren $|110\rangle$ und $|111\rangle$, es wird also durch eine unitäre Matrix beschrieben. Seine quantenmechanische Realisierung *ohne* Phasenverschiebung ist etwas trickreich, aber möglich. Insbesondere kann es als eine Folge von Operationen mit jeweils höchstens zwei Eingangsvariablen realisiert werden, was für die technische Realisierung von großer Bedeutung ist: Interaktionen zwischen drei Quantenbits gleichzeitig wären zu weit jenseits des derzeit Realisierbaren.

Ein Quantencomputer kann also mit der Negation und dem TOFFOLI-Gate alle logischen Berechnungen durchführen. Arithmetische Operationen können nach den klassischen Regeln der Schaltalgebra auf logische zurückgeführt werden; auch sie sind somit in einem Quantencomputer realisierbar.

d) Der Algorithmus von Shor

Als Beispiel für die kryptographische Relevanz eines Quantencomputers wollen wir die Faktorisierung einer ganzen Zahl N betrachten.

Der dümmste Ansatz zur Faktorisierung von Hand beseht darin, daß wir x und y unabhängig voneinander die Zahlen von 2 bis N durchlaufen lassen und jeweils das Produkt xy berechnen; falls dieses gleich N ist, haben wir eine Zerlegung von N gefunden.

Für eine etwa hundertstellige Zahl N (mit deren Faktorisierung ein heutiger Computer keine größeren Schwierigkeiten hat) wären hierzu rund 10^{200} Multiplikationen notwendig, mit klassischen Computern ein Ding der Unmöglichkeit: Selbst wenn alle heutigen Computer seit Beginn des Universums daran gerechnet hätten, wäre erst ein verschwindend kleiner Bruchteil der Produkte berechnet.

Für einen Quantencomputer dagegen sind diese 10^{200} Multiplikationen überhaupt kein Problem: Da $10^{100} \approx 2^{332}$ ist, nehmen wir zwei Quantenregister x, y aus etwa 350 QBits und bringen beide in den Zustand, in dem alle Basisvektoren denselben Koeffizienten haben. Sodann berechnen wir das Produkt der beiden Registerinhalte auf reversible Weise, d.h. wir berechnen das Tripel (x, y, xy). Dieses ist in einem Zustand, in dem alle Kombinationen (x, y, xy) mit $0 \le x, y < 2^{350}$ überlagert sind, insbesondere also auch die, für die $xy = N$ ist. Der Quantencomputer kann also all diese Multiplikationen gleichzeitig durchführen.

Damit ist allerdings leider das Faktorisierungsproblem noch nicht gelöst, denn wir müssen das Tripel ja auch noch messen. Dabei kollabiert der überlagerte Zustand und wir erhalten als Ergebnis ein Tripel (x, y, xy) aus natürlichen Zahlen, über das wir keinerlei Kontrolle haben. Insbesondere ist die Wahrscheinlichkeit, daß an dritter Stelle die Zahl N steht, verschwindend gering, so daß dieser sehr einfache Ansatz definitiv nicht zum Ziel führt.

Ein Quantencomputer kann also zwar sehr viele Operationen gleichzeitig durchführen, aber dies läßt sich nur ausnutzen, wenn man das Ziel der Rechnung anschließend auch wirklich messen kann.

Deshalb geht der Algorithmus von SHOR das Faktorisierungsproblem völlig anders an: Wie von FERMAT bis hin zu den modernsten Siebalgorithmen immer wieder ausgenutzt wird, hat man dann eine gute Chancen, eine Zahl zu faktorisieren, wenn man sie selbst oder ein Vielfaches als Differenz zweier Quadrate darstellen kann: Ist

$$kN = x^2 - y^2 = (x - y)(x + y)\,,$$

so kann man hoffen, daß $\operatorname{ggT}(x - y, N)$ und $\operatorname{ggT}(x + y, N)$ echte Teiler von N sind.

Der Algorithmus von SHOR nutzt dies aus, indem er für eine zufällig gewählte Zahl x zwischen 2 und $N - 2$ ihre Ordnung modulo N berechnet, d.h. die kleinste natürliche Zahl r, für die

$$x^r \equiv 1 \bmod N$$

ist. Eine solche Zahl r muß es nicht geben; für $N = 4$ und $x = 2$ beispielsweise gibt es keine. Elementare zahlentheoretische Betrachtungen zeigen, daß die zu N teilerfremde Zahlen bezüglich der Multiplikation modulo N eine zyklische Gruppe bilden; genau für diese Zahlen gibt es also ein solches r, und für alle anderen ist der ggT von x und N ein echter Teiler von N.

Falls r existiert und ungerade ist, hat man Pech gehabt und beginnt noch einmal mit einem neuen x; andernfalls ist

$$(x^{r/2+1} + 1)(x^{r/2} - 1) \equiv 0 \bmod N\,,$$

wir sind also genau in der obigen Situation und können ggTs berechnen. Falls dies keine echten Teiler sind, haben wir ebenfalls Pech gehabt und müssen mit einem neuen x von vorne anfangen. Da r die kleinste Zahl ist, für die $x^r - 1$ durch N teilbar ist, passiert das genau dann, wenn $x^{r/2} + 1$ durch N teilbar ist, also im Fall, daß $x^{r/2} \equiv -1 \bmod N$. Der Algorithmus ist somit genau dann nützlich, wenn dieser Fall nicht allzu oft (oder gar immer) eintritt.

Ist etwa p eine ungerade Primzahl und $N = p^n$, so kann bei geraden r die Primzahl p keinesfalls Teiler *beider* Faktoren

$$x^{r/2+1} + 1 \quad \text{und} \quad x^{r/2} - 1$$

sein, da sich die beiden nur um zwei unterscheiden. Also ist einer der beiden durch p^n teilbar, was natürlich nur der erste sein kann. Somit versagt der Algorithmus von SHOR in diesem Fall für *jedes* x.

Im kryptographisch besonders interessanten Fall, daß $N = pq$ Produkt zweier ungerader Primzahlen ist, sieht es aber – je nach Standpunkt leider oder zum Glück – ganz anders aus: Da modulo einer Primzahl $y \equiv \pm 1$ die beiden einzigen Lösungen der Gleichung $y^2 \equiv 1$ sind, zeigt eine einfache, wenn auch etwas langwierige Argumentation über den chinesischen Restesatz, daß für mindestens die Hälfte aller zu N primen Zahlen die Ordnung r gerade und $x^{r/2}$ nicht kongruent -1 modulo N ist. Hier liegt also die Erfolgswahrscheinlichkeit bei über 50%; wiederholt man die die Rechnung mit einem anderen x, kommt man bereits auf 75%, nach sieben zufällig gewählten Werten von x auf über 99%, nach zehn auf über 99,9%. Falls es also gelingt, r effizient zu berechnen, ist dieses Verfahren extrem gefährlich für das RSA-System.

Die Berechnung von r stellt uns vor ein ähnliches Problem wie die Berechnung eines diskreten Logarithmus, und in der Tat führt der Algorithmus von SHOR zur Bestimmung von r auch auf einen Algorithmus zur Berechnung diskreter Logarithmen; auch die Verallgemeinerung auf entsprechende Probleme für elliptische Kurven stellt keine prinzipiellen Probleme.

SHOR geht folgendermaßen vor: Für $N < 2^L$ braucht er einen Quantencomputer mit zwei Quantenregistern der Längen $2L$ beziehungsweise L. Im ersten speichert er die Überlagerung aller Zahlen a von 0 bis $2^{2L} - 1$, für das zweite berechnet er den Zustand x hoch erstes Register modulo N. Der Aufwand hierfür liegt für die klassische Berechnungsmethode durch fortgesetztes Quadrieren in der Größenordnung von L^3 Operationen.

Der Computer befindet sich dann in einem Zustand, in dem alle Paare $(a, x^a \bmod N)$ mit $0 \le a < 2^{2L}$ überlagert sind. Wird nun der Inhalt des zweiten Registers gemessen, ist die Chance für das Messen der Zahl eins natürlich verschwindend gering; das Meßergebnis wird *irgendeine* Zahl b zwischen 0 und $N - 1$ sein.

Durch die Messung des zweiten Registers hat sich aber der Inhalt des ersten verändert: Der *gesamte* Computer ist *ein* quantenmechanisches System, das in einen Zustand gebracht wurde, in dem der Inhalt des zweiten Registers gleich x hoch dem Inhalt des ersten Registers ist. Die Messung des zweiten Registers kann an dieser Tatsache nichts ändern, und da das zweite Register nach der Messung die Zahl b enthält, kann eine Messung des ersten Registers nun nur noch ein Ergebnis a liefern, für das $x^a \bmod N = b$ ist. Ein solches Ergebnis nützt uns aber nichts, und deshalb dürfen wir das erste Register keinesfalls messen.

Da das erste Register doppelt so lang ist wie das zweite, befindet es sich immer noch in einem überlagerten Zustand, nämlich in der Überlagerung aller Zahlen $0 \le a < 2^{2L}$, für die $x^a \equiv b \bmod N$ ist. Ist a_0 die kleinste solche Zahl, sind dies genau diejenigen Zahlen der Form $a_0 + kr$ mit $k \in \mathbb{N}_0$, die kleiner sind als 2^{2L}; hierbei ist r die gesuchte Ordnung.

Bei den Zahlen im überlagerten Zustand im Register handelt es sich also um eine periodische Folge von $m = \left[\frac{2^{2L} - a_0}{r}\right]$ Zahlen; was uns interessiert, ist die Periode r.

Die Periode eines Gitters läßt sich optisch aus dem Beugungsspektrum des Gitters bestimmen; genauso wollen wir auch hier vorgehen und nicht den *Inhalt* des ersten Registers messen, sondern so etwas wie sein *Beugungsspektrum.* Ein wesentlicher Aspekt der Quantentheorie ist schließlich, daß auch einander ausschließende Zustände eines Teilchens miteinander interferieren: Falls etwa ein Photon zwei mögliche Wege zu einem Punkt zurücklegen kann, tritt auch im Falle eines einzigen Photons Interferenz auf.

Das rechnerische Analogon zum Beugungsspektrum ist die *Quanten-*FOURIER-*Transformation.* Ersteres leitet man bekanntlich aus dem HUYGENSchen Prinzip ab, wonach jeder Punkt, in dem das Gitter durchlässig ist, Ausgangspunkt einer neuen Welle ist; diese wird beschrieben durch eine komplexe Exponentialfunktion, und das Beugungsspektrum ist gegeben durch die Summe aller dieser Exponentialfunktionen.

Ganz entsprechend ordnet die Quanten-FOURIER-Transformation eines Registers der Länge $2L$ ordnet dem Inhalt $|x\rangle$ dieses Registers (aufgefaßt als Zahl zwischen 0 und $2^{2L}-1$ oder als Überlagerung mehrerer solcher Zahlen) den Zustand

$$2^{-L}\sum_{\nu=0}^{2^{2L}-1} e^{2\pi|x\rangle\nu/2^{2L}}\,|\nu\rangle$$

zu. Man überzeugt sich leicht, daß dies eine unitäre Abbildung definiert; dies liegt einfach an der wohlbekannten Formel

$$\sum_{\nu=1}^{m} e^{k\nu/m} = \begin{cases} m & \text{falls } m|k \\ 0 & \text{sonst} \end{cases} ,$$

hinter der die Symmetrie der n-ten Einheitswurzeln zum Nullpunkt der komplexen Zahlenebene steckt – falls $n = m/ggT(m,k)$ größer als eins ist.

Schwieriger ist es, diese Abbildung quantenmechanisch zu realisieren; dazu braucht man die sogenannte schnelle FOURIER-Transformation, die durch geschickte Gruppierung der Summanden nach dem Prinzip *Teile und herrsche* eine deutlich effizientere Berechnung der Abbildung erlaubt und deren einzelne Rechenschritte sich leicht in Quantenprozesse übersetzen lassen, deren Ergebnis jeweils nur von zwei QBits abhängt.

Diese Quanten-FOURIER-Transformation wird nun also auf das erste Register angewandt; dadurch kommt dieses in den Zustand

$$\begin{aligned} &2^{-L}\sum_{\nu=0}^{2^{2L}-1}\frac{1}{\sqrt{m}}\sum_{\mu=1}^{m} e^{2\pi i(a_0+\mu r)\nu/2^{2L}}\,|\nu\rangle \\ &= \frac{1}{2^L\sqrt{m}}\sum_{\nu=0}^{2^{2L}-1} e^{2\pi i a_0}\left(\sum_{\mu=1}^{m} e^{2\pi i r\mu\nu/2^{2L}}\right)|\nu\rangle \end{aligned} .$$

Da $m = \left[\frac{2^{2L}-a_0}{r}\right] \approx 2^{2L}/r$ ist, läßt sich die Klammer approximieren durch

$$\sum_{\mu=1}^{m} e^{2\pi i r\mu\nu/2^{2L}} \approx \sum_{\mu=1}^{m} e^{2\pi i\mu\nu/m} = \begin{cases} m & \text{falls } m|\nu \\ 0 & \text{sonst} \end{cases} .$$

Der Zustand des ersten Registers ist also ungefähr gleich

$$\frac{1}{2^L\sqrt{m}}\sum_{\nu=0}^{m-1} e^{2\pi i a_0} m\,|r\nu\rangle = \frac{e^{2\pi i a_0}\sqrt{m}}{2^L}\sum_{\nu=0}^{m-1}|r\nu\rangle .$$

(Durch Aufsummieren der geometrischen Reihe kann man natürlich auch leicht den genauen Wert der geometrischen Reihe berechnen.)

Falls wir ein Register messen, dessen Inhalt durch die approximative Zustandsfunktion beschrieben wird, erhalten wir mit *Sicherheit* ein Vielfaches von r als Ergebnis, wobei die verschiedenen Vielfachen allerdings gleiche Wahrscheinlichkeit haben.

Die exakte Zustandsfunktion führt zu leichten Abweichungen davon: Die Betragsmaxima der Koeffizienten liegen nun bei den Vielfachen der wirklichen Periode $2^{2L}/r$, aber weiter davon entfernt liegende Werte sind nicht mehr unmöglich, sondern nur noch unwahrscheinlich.

Wir wiederholen deshalb das Experiment mit demselben x mehrfach (eine genauere Analyse zeigt, daß etwa L Wiederholungen mit hoher Wahrscheinlichkeit genügen) und haben dann verschiedene Meßwerte für ν. Für die meisten, wenn nicht gar alle dieser Werte gibt es eine Zahl λ, so daß

$$\nu \approx \lambda \frac{2^{2L}}{r} \qquad \text{oder} \qquad \frac{\nu}{2^{2L}} \approx \frac{\lambda}{r}$$

ist.

In der hinteren Gleichung ist die linke Seite bekannt; von der rechten kennen wir weder Zähler noch Nenner. Wir wissen aber, daß der Nenner r als Ordnung von x modulo N kleiner als N ist und damit sehr viel kleiner als der Nenner 2^{2L} auf der linken Seite, der ja größer als N^2 ist. Zur Bestimmung von r müssen wir also für die verschiedenen Brüche $\lambda/2^{2L}$ Approximationen finden, deren Nenner kleiner ist als N.

Die besten rationalen Approximationen einer reellen Zahl mit Nennern einer vorgegebenen Größenordnung liefert, wie wir in Kap. 6, §6f, gesehen haben, deren Kettenbruchentwicklung; also berechnen wir für jeden Meßwert μ alle Konvergenten des Kettenbruchs zu $\frac{\mu}{2^{2L}}$ mit Nenner kleiner N und suchen dann nach einem Nenner r, der bei möglichst vielen Meßwerten auftritt. Dieser ist ein Kandidat für die gesuchte Ordnung von x, und wir verfahren damit wie im SHORschen Algorithmus angegeben.

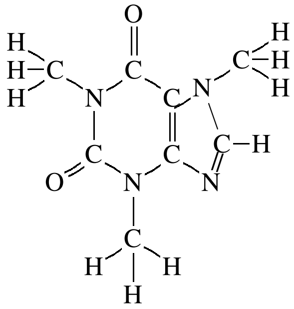
e) Experimentelle Realisierung

Das große Problem beim Bau von Quantencomputern ist die *Dekohärenz:* Ein überlagerter Zustand bleibt nur erhalten, wenn keinerlei Wechselwirkung mit äußeren Systemen eintritt; dies läßt sich auch mit großem Aufwand nur für sehr kurze Zeitspannen sicherstellen. SHOR entwickelte *Quantencodes,* die nach dem Vorbild klassischer fehlerkorrigierender Codes ein gewisses Maß an Dekohärenz verkraften können, aber auch damit läßt sich diese Zeitspanne nur begrenzt ausdehnen. Ein Quantencomputer auf der Basis von Photonen mit ihren extrem kurzen Dekohärenzzeiten erscheint deshalb im Augenblick jenseits jeder technischen Realisierbarkeit.

Am vielversprechendsten ist im Augenblick ein Ansatz, der mit NMR-Spektroskopie arbeitet. Bei dieser handelt es sich um eine schon seit über

dreißig Jahren gebräuchliche chemische Analysemethode, die inzwischen auch für bildgebende Verfahren in der Medizintechnik eingesetzt wird. Sie beruht auf der magnetischen Resonanz gewisser Atomkerne. Zwar zeigen nur wenige Isotopen Kernresonanz, aber es handelt sich dabei um Isotope häufiger und wichtiger Atome: ^{1}H, ^{2}H, ^{10}B, ^{11}B, ^{13}C, ^{14}N, ^{15}N, ^{17}O, ^{19}F, ^{29}Si und ^{31}Si.

Als Beispiel eines für einen ersten Quantencomputer geeigneten Moleküls wurde (nicht unbedingt ganz ernsthaft) das *Koffein* vorgeschlagen, das in der Tat genug solche Atome enthält, um einen kleinen „Supercomputer in der Kaffeetasse“ zu realisieren:



Protonen und Neutronen in einem Atom haben jeweils Spin $\pm 1/2$, wobei sich benachbarte Protonen sowie benachbarte Neutronen jeweils antiparallel ausrichten. Bleibt dabei ein Gesamtspin übrig, richtet sich das Atom in einem umgebenden Magnetfeld der Stärke von etwa 9 bis 15 Tesla in einer von wenigen wohldefinierten Richtungen aus, zwischen denen es durch Aufnahme beziehungsweise Abgabe von Strahlungsquanten wechseln kann. Mit geeigneten Radiowellen läßt sich also zwischen verschiedenen Zuständen hin- und herschalten; durch Aufnahme eines Resonanzspektrums lassen sich die (über viele Atome gemittelten) Zustände ablesen. Dadurch, daß man hier nicht mit Quantenzuständen einzelner Partikel arbeitet, sondern über viele Partikel mittelt, wird auch das Problem der Dekohärenz entschärft. Man rechnet damit, daß ein solcher Computer schon bald die Zahl fünfzehn faktorisieren kann.

QBits in einzelnen Atomen benutzt die um 1995 in Innsbruck konzipierte *Ionenfalle.* Hier werden Ionen durch elektromagnetische Felder in einer linearen Anordnung gehalten; die QBits werden kodiert durch Anregungszustände der Ionen (von denen jedes durch einen eigenen Laser kontrolliert wird) und der Gitterschwingungen (Phononen) zwischen den einzelnen Ionen. Überlagerte Zustände, die mehrere QBits umfassen, werden über die Vibrationsenergie ihres Schwerpunkts kontrolliert.

Am National Institut of Standards in Boulder, Colorado, wurde so die kontrollierte Negation implementiert; Faktorisierung kleiner Zahlen erscheint durch Weiterentwicklung und Vergrößerung der Apparatur möglich, allerdings müssen dazu noch einige technische Probleme gelöst werden.

Für weiteres und insbesondere auch andere physikalische Ansätze sei auf die zitierte Literatur verwiesen.

f) Literaturhinweise

Auch hier sind als erstes die beiden schon im vorigen Kapitel erwähnten und kurz charakterisierten Bücher

MICHAEL BROOKS [Hrsg.]: *Quantum computing and communications,* Springer, 1999 *und*

COLIN P. WILLIAMS, SCOTT H. CLEARWATER: *Explorations in quantum computing,* Springer Telos, 1997

zu nennen. Ein Buch, das speziell auf die algorithmische Seite Bezug nimmt (oft eher anhand von Beispielen als einer klaren Darstellung des Algorithmus) ist

ARTHUR O. PITTENGER: *An introduction to quantum computing algorithms,* Birkhäuser, Boston, 1999

Die Originalarbeit zum Algorithmus von SHOR wurde kürzlich in überarbeiteter Form nachgedruckt:

PETER W. SHOR: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, *SIAM Rev.* **41** (1999), S. 303–332

Einen weiteren Übersichtsartikel, in dem auch auf das hier nicht behandelte Problem der Fehlerkorrektur in Quantencomputern eingegangen wird, ist SHORS Vortrag auf dem internationalen Mathematikerkongreß 1998 in Berlin:

PETER W. SHOR: Quantum Computing, *Proceedings of the international congress of mathematicians Berlin 1998, Documenta Mathematica, Extra volume ICM* 1998 · *I,* DEUTSCHE MATHEMATIKER-VEREINIGUNG, 1998, S. 467–486 *oder*
www.mathematik.uni-bielefeld.de/documenta/xvol-icm/00/00.html

§3: DNS-Computer

Alles Leben beruht auf der Informationsverarbeitung in den Zellen eines Organismus. Diese arbeitet mit komplexen chemischen Reaktionszykeln, die bei weitem noch nicht alle verstanden sind. Das wichtigste Speichermedium ist die Desoxyribonucleinsäure, kurz DNS, der Zelle.

Die Informationsdichte dort ist ungeheuer hoch: In trockenem Zustand benötigt ein Bit gerade einmal ein Volumen von $1\,\mathrm{nm}^3$, in der Zelle etwa $100\,\mathrm{nm}^3$. Für einen Kubikzentimeter trockener DNS kommt man somit auf eine Speicherkapazität von 10^{21} Bit, das ist eine achtel Billion Gigabyte; für DNS in der Zelle kommt man mit einem Kubikzentimeter immerhin noch auch 125 Milliarden Gigabyte. In einem Computer faßt ein Speicherriegel der Größe $15\,\mathrm{cm} \times 3\,\mathrm{cm} \times 1\,\mathrm{cm}$ gerade einmal ein halbes Gigabyte, und auch bei CDs braucht man größenordnungsmäßig fast eine Billion Stück, um die in einem Kubikzentimeter DNS enthaltene Information zu speichern.

Hinzu kommt, daß die Information in der DNS massiv parallel verarbeitet werden kann mit einem Energieaufwand pro Operation, der ungefähr zehn Größenordnungen unter dem heutiger Supercomputer liegt.

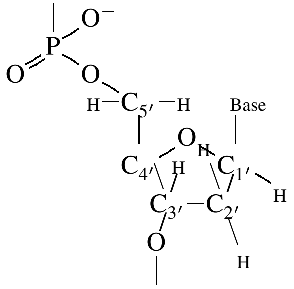
Wenn man dieses Potential für klassische Probleme der wissenschaftlichen oder gewerblichen Informationsverarbeitung nutzbar machen könnte, wäre dies eine Revolution, die auch die Kryptographie massiv verändern würde.

Seit 1994 gibt es einen ersten Hinweis darauf, daß dies vielleicht nicht völlig unrealistisch ist: Damals löste der Mathematiker und Informatiker LEONARD M. ADLEMAN (der hier bereits im Zusammenhang mit dem RSA-Verfahren erwähnt wurde) in einem molekularbiologischen Labor ein (ziemlich einfaches) graphentheoretisches Problem mit Hilfe von DNS.

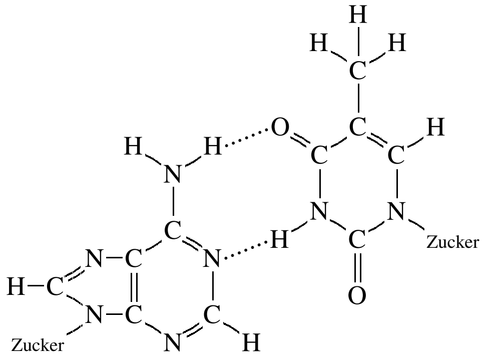
Um seine Vorgehensweise und das Potential der Methode zu verstehen, müssen wir zunächst kurz den Aufbau der DNS betrachten.

a) Die Desoxyribonucleinsäure

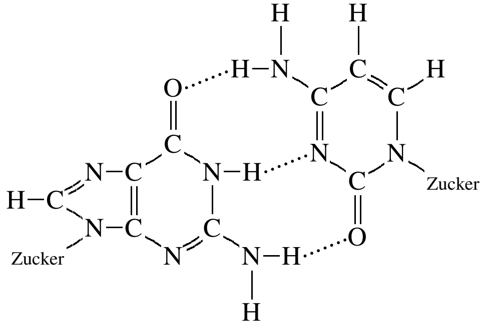
Die Desoxyribonucleinsäure hat ihren Namen vom hier abgebildeten Zuckermolekül Desoxyribose, aus dem ihr Gerüst aufgebaut ist.



Die fünf Kohlenstoffatome werden mit $1'$ bis $5'$ bezeichnet; in der Abbildung ist dies als Index eingetragen. An der Phosphatgruppe von $5'$ und der Hydroxygruppe von $3'$ fehlt jeweils ein Wasserstoffatom: Hier werden die Zucker zu einer Kette zusammengesetzt, wobei stets auf die $5'$-Phosphatgruppe eine $3'$-Hydroxygruppe folgt. Am Kohlenstoffatom $1'$ steht *Base;* hier wird eine jener vier Basen eingebaut, die die eigentlichen Informationsträger der DNS sind: Adenin, Guanin, Thymin und Cytosin.



Adenin Thymin



Guanin Cytosin

Wie man an den Abbildungen sieht, können sich zwischen Adenin und Thymin sowie zwischen Guanin und Cytosin die gestrichelt eingezeichneten Wasserstoffbrücken ausbilden; diese sorgen dafür, daß DNS nur selten in einzelnen Strängen vorkommt: Meist kombinieren sich zwei Stränge zu einem Doppelstrang, wobei jedem Adenin ein Thymin und jedem Guanin ein Cytosin gegenübersteht. Man bezeichnet diese Basen deshalb als zueinander WATSON-CRICK-komplementär.

Aus geometrischen Gründen hat ein DNS-Doppelstrang die berühmte Form der Doppelhelix, die wiederum selbst weitere geometrische Strukturen bildet. Für die Informationsverarbeitung in der Zelle sind alle diese geometrischen Strukturen sehr wichtig; die entsprechenden Mechanismen sind aber noch nicht so gut verstanden, daß man sie auch in Laborexperimenten ausnutzen könnte, so daß diese Strukturen für uns irrelevant sind.

b) Die Polymerase-Kettenreaktion

Ein wesentlicher Bestandteil des molekularen Rechnens ist der Aufbau von DNS-Sequenzen sowie die Vervielfältigung von Sequenzen mit erwünschten Eigenschaften. Das wichtigste Hilfsmittel dafür ist die 1987 von KARY B. MULLIS entwickelte Polymerase-Kettenreaktion.

Polymerasen sind Enzyme, die in der Zelle die Duplizierung der Information aus DNS-Strängen steuern; sie sind je nach Lebewesen verschieden, und auch innerhalb derselben Zelle gibt es oft mehrere Polymerasen mit verschiedenen Aufgaben. Das besonders gut erforschte Bakterium *Escherichia coli* etwa enthält drei DNS-Polymerasen; die am stärksten vertretene DNS-Polymerase I ist eine Kette mit 928 Aminosäuren. Es handelt sich hier also um sehr komplexe Enzyme, die deshalb auch heute noch nicht synthetisiert werden, sondern auch für die Laborarbeit von Bakterien produziert werden.

Polymerasen werden zusammen mit zwei sogenannten *Primern* eingesetzt, d.h. mit zwei Folgen von DNS-Basen; diese sorgen dafür, daß selektiv ein DNS-Strang produziert wird, der WATSON-CRICK-komplementär ist zum Stück zwischen den beiden Primern auf einem vorhandenen DNS-Strang. Insbesondere muß DNS also einsträngig vorliegen, bevor eine Polymerase aktiv werden kann. In der lebenden Zelle sorgen Enzyme dafür, daß benötigte Teilstränge zur Verfügung stehen; im Labor geht man brutaler vor und teilt den ganzen Doppelstrang durch Erhitzen auf eine Temperatur von $90^\circ - 95^\circ$ C.

Da diese Temperatur praktisch alle Enzyme zerstört, arbeitet man heute nicht mehr mit der Polymerase von *Escherichia coli,* sondern meist mit der sogenannten Taq-Polymerase des Bakteriums *Thermus aquaticus,* das in heißem Wasser lebt und daher auch eine hitzebeständige Polymerase hat, die ihre optimale katalytische Wirkung bei etwa 75° C entfaltet.

Zur Vervielfältigung von DNS geht man folgendermaßen vor:

Man gibt die DNS zusammen mit den beiden Primern, der Taq-Polymerase und den vier Nucleotiden, aus denen die DNS aufgebaut ist, in ein Gefäß und denaturiert zunächst die DNS durch Erhitzen, d.h. man zerlegt sie in ihre Einzelstränge. Sodann kühlt man ab auf etwa 55° C; dies führt dazu, daß sich die Primer an die passenden Stellen der DNS-Stränge angliedern. Eine Erhöhung der Temperatur auf etwa 75° C aktiviert die Polymerase, die nun dafür sorgt, daß (ausgehend von den $3'$-Enden der Primer) WATSON-CRICK-komplementäre Nucleotide an die DNS-Stränge angelegt werden.

Damit hat man die gewünschten DNS-Sequenzen verdoppelt, was im allgemeinen nicht ausreicht; deshalb wird das Verfahren etwa 25–35 Mal wiederholt, was zu einer etwa millionenfachen Vermehrung führt. Das Verfahren kann in sogenannten Thermocyclern automatisch durchgeführt werden, da es nur auf die zyklische Temperatursteuerung ankommt.

c) Adlemans Experiment

Gegeben seien eine gewisse Anzahl von Städten sowie Straßen, die Städte gewissen anderen Städten verbinden. Man finde eine Straßenverbindung, die eine vorgegebene Stadt mit einem vorgegebenen Ziel so verbindet, daß jede Stadt genau einmal besucht wird.

Dies ist eine von vielen Varianten eines graphentheoretischen Problems, das für große Anzahlen von Städten und Verbindungsstraßen rechnerisch nur mit großem Aufwand gelöst werden kann. (Es ist NP-vollständig.) ADLEMAN beschränkte sich allerdings auf nur sieben Städte und fand heraus, daß der durchschnittliche menschliche Betrachter etwa 54 Sekunden braucht um sein spezielles Problem zu lösen. Interessanter ist aber die Art und Weise, wie er diese Lösung statt durch Hinschauen durch sieben Tage Arbeit im Labor produzierte.

Die Grundidee ist folgende: Jede Stadt wird durch eine Folge von zwanzig DNS-Basen kodiert; als Lösung soll jene Folge von 140 DNS-Basen produziert werden, die den (in ADLEMANs Beispiel einzigen) Lösungsweg angibt.

Die Herstellung kurzer DNS-Sequenzen mit vorgegebener Basenfolge gehört heutzutage zu den Standardtechniken der Molekularbiologie; große Labors haben Automaten, die solche Sequenzen (über die Polymerase-Kettenreaktion) erzeugen, kleinere kaufen sie von den großen: Für 25$ kann man innerhalb von wenigen Tagen ein Reagenzglas bekommen, das etwa 10^{18} DNS-Stränge mit einer vorgegebenen Folge von zwanzig Basen enthält.

Für jede der sieben Städte wurde also ein solches Reagenzglas benötigt, dazu aber auch noch für jede Straße zwischen zwei Städten. Natürlich müssen die Basenfolgen für Städte und für Straßen aufeinander abgestimmt sein: ADLEMAN faßte die Basenfolge für die i-te Stadt auf als ein Paar (x_i, y_i) aus zwei Basenfolgen der Länge zehn, und wenn es eine Straßenverbindung zwischen i-ter und j-ter Stadt gibt, kodierte er diese durch die Basenfolge $(\overline{y_i}, \overline{x_j})$, wobei die Überstreichung hier für das WATSON-CRICK-Komplement stehen soll.

Der erste Schritt der eigentlichen Berechnung bestand darin, alle diese Sequenzen (jeweils etwa 10^{14} Moleküle) in Wasser aufzulösen und Ligase dazuzugeben. (Ligasen sind Enzyme, die kurze DNS-Sequenzen zu langen zusammenfügen.) Dazu kommt noch etwas Puffer und ähnliches, und ungefähr eine Sekunde später sind im Reagenzglas viele doppelsträngige DNS-Sequenzen zu finden, die möglichen Wegen durch den Graph entsprechen.

Unter diesen Sequenzen sind mit praktisch 100%-iger Sicherheit auch solche, die der Lösung des Problems entsprechen; das Problem besteht genau wie bei den überlagerten Zuständen in Quantencomputern darin, diese herauszufiltern. Ein Vorteil gegenüber Quantencomputern ist allerdings, daß man beim molekularen Rechnen durch Messungen üblicherweise nichts zerstört.

Im zweiten Schritt ging es darum, Sequenzen zu finden, die zu Wegen mit dem Richtigen Anfangs- und Zielort gehören. Finden bedeutete in diesem Fall, daß diese Sequenzen stark vermehrt werden sollten, und das ist ein klarer Fall für die Polymerase-Kettenreaktion: Man nehme die WATSON-CRICK-Komplemente des Start- und des Zielorts als Primer und lasse die Reaktion laufen. Danach sind alle Sequenzen, bei denen Anfangs- und Zielort stimmen, etwa millionenfach verstärkt; die, bei denen nur einer der beiden Orte stimmt, etwa dreißigfach, und der Rest überhaupt nicht. Entnimmt man also eine kleine Probe zur Weiterverarbeitung, so enthält diese kaum noch Sequenzen, bei denen einer der beiden Orte falsch ist.

Im dritten Schritt ging es darum, alle Sequenzen auszusondern, die eine falsche Länge haben. Ein Weg, der jede Stadt genau einmal berührt, entspricht einer Folge von sieben Städten und damit einem DNS-Strang der Länge 140. Zur Isolation dieser Stränge führt eine Art chromatographisches Verfahren, die *Gel-Elektrophorese.*

Hierbei wird ausgenutzt, daß die betrachteten Moleküle negativ geladen sind (man achte auf das O^- in der Desoxyribose). Bringt man die Lösung also auf einen Gel auf, meist Agarose oder Polyacrylamid, und legt ein elektrisches Feld an, so hängt die Wanderungsgeschwindigkeit ab sowohl von der elektrischen Ladung als auch der Molekülgröse und (in geringerem Ausmaß) einigen anderen Gegebenheiten. Die Methode ist aber jedenfalls trennscharf genug, um Sequenzen, deren Länge sich um zwanzig Nucleotide unterscheidet, zuverlässig zu trennen; nachdem die Moleküle einige Zeit gewandert sind, entnimmt man die aus der 140er-Region und verwirft den Rest.

Unter den noch verbleibenden Sequenzen befinden sich immer noch zahlreiche Nichtlösungen, denn eine Folge von sieben Städten erhält man auch, wenn man einige Städte ausläßt und andere dafür mehrfach besucht. Es muß also entweder sichergestellt werden, daß keine Stadt mehrfach besucht wurde, was mit molekularbiologischen Methoden sehr schwer sein dürfte, oder aber, daß jede Stadt mindestens einmal besucht wurde.

Letzteres ist für den Anfangs- und den Zielort bereits bekannt; bleiben also noch fünf Städte. Für diese verwendete ADLEMAN eine Kombination aus molekularbiologischer und physikalischer Vorgehensweise: Um diejenigen Moleküle auszusondern, die die Sequenz für eine gegebene Stadt nicht enthalten, heftete er WATSON-CRICK-Komplemente dieser Stadt an kleine Eisenkügelchen mit einem Durchmesser von etwa 1 μm, gab dies zu der denaturierten (d.h. wieder durch Erhitzen in Einzelstränge zerlegten) noch verbliebenen Lösung und lies die Stränge sich wieder kombinieren. Dabei paarten sich die Sequenzen an Eisenkugeln mit Teilsequenzen jener Moleküle, die einen Weg durch die betrachtete Stadt beschreiben. Die so entstandenen DNS-Sequenzen waren also mit Eisenkugeln verbunden und konnten so mit einem Magneten am Rand des Reagenzglases festgehalten werden, während der Rest des Glases ausgeschüttet wurde.

Danach wurde frisches Wasser zugegeben und das ganze wieder erhitzt zur Denaturierung; jetzt aber wurde der Magnet an das erhitzte Reagenzglas gehalten, so daß der die Sequenzen mit Eisenkügelchen festhielt. Der Rest wurde umgegossen in ein anderes Reagenzglas, wo dann das gleiche Spiel für die nächste Stadt wiederholt werden konnte, bis alle fünf Städte abgearbeitet waren.

Falls danach noch Moleküle übrig waren, konnte es sich nur um Lösungen handeln. Da es aber wahrscheinlich nur noch relativ wenige waren, vermehrte sie ADLEMAN zunächst durch eine Polymerase-Kettenreaktion und überzeugte sich durch Gel-Elektrophorese davon, daß seine Lösung wirklich Sequenzen der Länge 140 enthielt. Zur vollständigen Lösung des Problems mußten diese dann nur noch analysiert werden.

Auch hierzu dient wieder die Polymerase-Kettenreaktion in Verbindung mit Gel-Elektrophorese: Für jeden Zwischenort führte ADLEMAN eine Polymerase-Kettenreaktion durch, wobei er als Primer die WATSON-CRICK-Komplemente von Anfangsstadt und Zwischenort nahm. Dadurch wurde die Teilsequenz bis zu diesem Zwischenort stark vermehrt, und durch Gel-Elektrophorese läßt sich feststellen, wie lange sie ist. Diese Länge, geteilt durch zwanzig, ist die Position der Stadt im Lösungsweg.

d) Wie geht es weiter?

ADLEMAN hat mit seinem Experiment zwar kein neues Problem gelöst, aber er hat gezeigt, daß molekularbiologische Verfahren zumindest grundsätzlich zur Lösung rechnerischer Probleme benutzt werden können – bis zu ihrem effizienten Einsatz ist es sicherlich noch ein langer Weg.

Das durchgeführte Experiment überprüft mit brutaler Gewalt (fast) *alle* möglichen Wege durch den Graphen, was schon bei 200 Städten eine DNS-Menge verlangen würde, die mehr wiegt als die Erde. Mit klassischen Computern und den besten derzeit bekannten Algorithmen lassen sich dagegen auch Probleme mit einigen Tausend Städten behandeln.

Auch der letzte Schritt des Experiments funktionierte nur deshalb, weil das Problem eine eindeutige Lösung hatte. Dies ist allerdings kein großes Problem, denn alternative Methoden zur Bestimmung der Basenfolge in einem DNS-Strang gibt es natürlich: Schließlich wurde inzwischen sogar das gesamte menschliche Genom entschlüsselt.

Um ein Gefühl für die mögliche Relevanz des molekularen Rechnens in der Zukunft zu bekommen, insbesondere auch in Bezug auf die Kryptologie, ist es vielleicht ganz nützlich, zum Vergleich ein völlig anderes Thema zu betrachten, die bisherige Geschichte der Faktorisierung ganzer Zahlen.

Das einfachste Verfahren zur Faktorisierung einer ganzen Zahl N ist das systematische Durchprobieren aller Zahlen $d \leq \sqrt{N}$; dieser Algorithmus ist vergleichbar mit ADLEMANs oben beschriebener Methode.

Wie wir bei der Diskussion von SHORs Algorithmus gesehen haben, liefert der Ansatz von FERMAT ein alternatives Verfahren; berechnet man, wie es FERMAT tat, für $a = 1, 2, 3, \ldots$ systematisch die Zahlen $N + a^2$ in der Hoffnung, darunter ein Quadrat zu finden, so lassen sich zumindest Zahlen mit zwei nahe beieinanderliegenden Faktoren deutlich schneller zerlegen.

Das neunzehnte Jahrhundert war von der Mechanisierung geprägt; in der ersten Hälfte des zwanzigsten Jahrhunderts erreichte diese auch die

Zahlentheorie, als D.H. LEHMER eine Siebversion der FERMAT-Methode mit Zahnrädern und Fahrradketten zur Faktorisierung verwendete.

Bei den elektrischen und elektronischen Rechner griff die neue Technik schneller auf die Zahlentheorie über: D.H. LEHMER hatte bereits auf den ersten Computern seiner Universität Hintergrundprogramme laufen, die sich mit der Faktorisierung von Zahlen beschäftigten, wenn es sonst nichts zu tun gab.

In den Siebzigerjahren, als Computer leistungsfähig genug waren, um in Zahlbereiche vorzudringen, die vorher jenseits praktisch durchführbarer Rechnungen lagen, war wieder die Mathematik gefragt, die in den letzten dreißig Jahren immer neue Faktorisierungsalgorithmen entwickelte. Diese mußten auf den jeweils aktuellen Computern implementiert werden, wobei in den Achtzigerjahren vor allem die CRAY-Computer eine sehr große Rolle spielten und aufgrund ihrer Pipeline-Architektur zur Entwicklung und Optimierung neuer Programmiertechniken zwangen.

In den Neunzigerjahren brachte das Internet die Möglichkeit zum verteilten Rechnen; 1994 wurde dadurch jene berühmte 129-stellige Zahl faktorisiert, die 1978 bei der Vorstellung des RSA-Systems als Beispiel diente und von der man damals überzeugt war, daß sie auch in hundert Jahren noch sicher sei. (Heute hält das Bundesamt für Sicherheit in der Informationstechnik Zahlen mit mindestens 1024 Bit für sicher; ab 2005 verlangt es die doppelte Länge.)

Für Fortschritte bei der Faktorisierung waren also jeweils drei Aspekte maßgeblich:

- Bessere mathematische Algorithmen
- Bessere Maschinen
- Besseres Verständnis des Umgangs mit diesen Maschinen.

Auch die weitere Entwicklung des molekularen Rechnens wird wohl von drei ähnlichen Aspekten abhängen.

Sechs Jahre nach ADLEMANs Experiment kann man versuchen, zumindest ein bißchen über jeden dieser drei Aspekte zu spekulieren.

Die weitaus größte Zahl von Publikationen im Umkreis der DNS-Computer beschäftigt sich mit theoretischen Algorithmen sowie mit abstrakten Maschinen und formalen Sprachen zur Modellierung des molekularen Rechnens. Viele dieser Arbeiten kommen von theoretischen Informatikern ohne Chemieausbildung und dürften wohl selbst innerhalb der Informatik schon in wenigen Jahren vergessen sein; bis zu einer Erprobung im Labor dürften es kurzfristig auch vom verbleibenden Rest kaum eines bringen. Langfristig allerdings könnten einige wenige dieser Ansätze durchaus zu brauchbaren Verfahren weiterentwickelt werden.

Von besseren *Maschinen* dürfte der Fortschritt des molekularen Rechnens angesichts der hochentwickelten verfügbaren chemischen Labortechnik in den nächsten Jahren wohl kaum abhängen; vielmehr geht es darum, zunächst zu erforschen, welche primitiven Grundoperationen wie durchgeführt werden können, um effizient molekular zu rechnen. Mit dieser Frage beschäftigen sich eine ganze Reihe molekularbiologischer Laboratorien, die beispielsweise mit an Oberflächen angehefteten Molekülen und ähnlichem arbeiten und teilweise auch schon erste Erfolge melden können: So gelang inzwischen die molekulare Addition zweier (kleiner) Binärzahlen. Im Augenblick ist noch nicht abzusehen, ob sich mehrere Grundparadigmen durchsetzen werden, oder ob die Entwicklung auf *den* DNS-Computer zusteuert.

Der dritte Aspekt, der bessere Umgang mit der Maschinerie, kann wohl erst dann zum Tragen kommen, wenn die Entwicklungen beim Algorithmenentwurf und bei den molekularbiologischen Ansätzen anfangen, gegeneinander zu konvergieren, wenn also die entsprechenden Experten anfangen, Notiz voneinander zu nehmen. Dies setzt einerseits voraus, daß die Labors stabile und zuverlässige Verfahren entwickeln. Vor allem aber wird ein wirklicher Fortschritt erst dann möglich sein, wenn es eine nennenswerte Anzahl von Wissenschaftlern gibt, die sowohl in der Molekularbiologie als auch in der Algorithmik zumindest eine gewisse Mindestqualifikation haben.

Der Zeitrahmen dafür hängt wesentlich davon ab, ob es gelingt, ausreichend Studenten für eine solche duale Qualifikation zu begeistern und sie ihnen auch zu ermöglichen, oder ob die Studenten angesichts des Booms sowohl in der Informationstechnik als auch der Molekularbiologie vollauf mit der Qualifikation in einem dieser Gebiete zufrieden sind.

An diesem Punkt geraten die Spekulationen ins Gebiet der Politik, wo

Spekulationen nur selten zu etwas vernünftigem führen; daher soll diese Vorlesung besser hier enden.

e) Literaturhinweise

Die Originalarbeit von Adleman ist

Leonard M. Adleman: Molecular computation of solutions to combinatorial problems, *Science* **266** (1994), S. 1021–1024

Eine erste Diskussion, bestehend aus mehreren Leserbriefen und einer Antwort Adlemans ist in *Science* **268** (1995), S. 481–484 zu finden. Eine gut lesbare Darstellung der wesentlichen Punkte des Experiments ist

Leonard M. Adleman: Rechnen mit DNA, *Spektrum der Wissenschaft,* November 1998, S. 70–77

Einen Überblick über die weitere Entwicklung geben beispielsweise die Tagungsbände *DNA Based Computers* I-V; die ersten drei wurden von der American Mathematical Society als Bände 27 (1996), 44 und 48 (beide 1999) der *Series in Discrete Mathematics and Theoretical Computer Science* veröffentlicht, die beiden weiteren habe ich noch nicht im Druck gesehen. Ein ähnlicher Tagungsband ist

Gheorghe Răun [Hrsg:] *Computing with bio-molecules,* Springer, 1998

Eine Übersicht über die molekularbiologischen Methoden, die dem molekularen Rechnen zugrunde liegen, findet man beispielsweise in den Büchern

Bernard L. Glick, Jack J. Pasternak: *Molekulare Biotechnologie,* Spektrum Akademischer Verlag, 1995 *und*

Rolf Knippers: *Molekulare Genetik,* Thieme, ^{7}1997,

von denen das erste mit etwas anschaulicheren Darstellungen arbeitet und das zweite stärker auf den biochemischen Hintergrund eingeht. Detailierte Laboranweisungen zu den Grundtechniken findet man etwa in

Hans Günther Gassen, Gangolf Schrimpf [Hrsg.]: *Gentechnische Methoden,* Spektrum Akademischer Verlag, ^{2}1999