eine in RFC XXXX genormte Liste von Namen, die jeweils ein asymmetrisches, ein symmetrisches und gegebenenfalls noch ein Unterschriftenverfahren enthalten. Jede der drei Komponenten kann auch einfach „Keines“ sagen, was zur Folge hat, daß keine entsprechende Verschlüsselung stattfindet.

Der Server vergleicht die erhaltene Liste mit den Kryptoverfahren, die er beherrscht, und wählt dann eines aus – oder aber beendet die Verbindung, falls es kein von beiden beherrschtes *bzw.* akzeptiertes Verfahren gibt.

Der entsprechende Austausch findet selbstverständlich im Klartext statt und ist daher ein möglicher Angriffspunkt für einen Gegner: Indem er die Liste des Clients abfängt und alle starken Verfahren daraus streicht, kann er die Verwendung schwacher Kryptographie erzwingen.

Im nächsten Schritt identifiziert sich der Server und schickt seinen öffentlichen Schlüssel. Da auch hier ein Angreifer sich als Server ausgeben könnte, sollte dieser Schritt mit einer Authentisierung verbunden sein, d.h. der Server legt dem Client ein unterschriebenes Zertifikat vor, das sowohl seine Identität als auch seinen öffentlichen Schlüssel und das dazugehörige Verfahren enthält.

Dadurch ist das Problem natürlich nur um eine Stufe verschoben, denn ein Angreifer könnte sich auch als Zertifizierungsbehörde ausgeben. Theoretisch ist dies dadurch gelöst, daß die öffentlichen Schlüssel der (relativ wenigen) anerkannten Zertifizierungsinstitutionen im Programmcode der Browser enthalten sind. Wer sich freilich seinen Browser einfach von irgendeiner Internetseite holt, hat keine Garantie, daß dort nicht auch zusätzliche Schlüssel eines Angreifer stehen.

Ein weiteres Problem besteht darin, daß Zertifizierungsbehörden relativ hohe Preise verlangen; ein einziges Zertifikat kann bereits über 300$ kosten. Für `amazon.com` und ähnliche Unternehmen sind das *peanuts;* kleinere Betriebe oder auch Institutionen wie etwa die Universität Mannheim aber schrecken vor diesen Kosten zurück und stellen für ihr Subnetz eigene Zertifikate aus. Grundsätzlich gibt es auch die Möglichkeit, daß sich solche selbsternannte Zertifizierungsstellen von einer offiziell anerkannten zertifizieren lassen mit einem Zertifikat, das ihnen (im Gegensatz zu den Inhabern „üblicher“ Zertifikate) auch das Recht

einräumt, selbst in einem gewissen Namensraum zu zertifizieren. Eine solche Lizens natürlich noch teurer und wird daher nicht oft vorgelegt. In einem solchen Fall, wenn die Identität des Servers nicht zweifelsfrei festgestellt werden kann, wird üblicherweise der Benutzer gefragt, ob er trotzdem weitermachen will und ob er gegebenenfalls die Unterschrift der dem Browser unbekannten Zertifizierungsstelle künftig anerkennen will.

Bei `ssh`-Verbindungen dürfte es wohl die Regel sein, daß der Server kein Zertifikat vorlegen kann; hier speichert der Client den Schlüssel *bzw.* einen Fingerabdruck davon, nachdem der Benutzer beim ersten Mal gefragt wurde, ob er sich sicher sei, mit dem richtigen Rechner verbunden zu sein. Künftig werden dann Verbindungen zu diesem Server nur noch aufgebaut, wenn der Server den richtigen Schlüssel schickt.

Wenn der Client den öffentlichen Schlüssel des Servers kennt, kann er nun zufällig einen Sitzungsschlüssel für das vom Server ausgewählte symmetrische Verfahren erzeugen und diesen mit dem asymmetrischen Verfahren verschlüsselt an den Server schicken. Die weitere Kommunikation erfolgt dann symmetrisch verschlüsselt, wobei gegebenenfalls noch zusätzlich eine Prüfsumme zur Sicherung der Nachrichtenintegrität übertragen wird. (Wie man solche Prüfsummen kryptographisch sicher erzeugt, werden wir weiter hinten sehen.)

b) PCKS #1v1.5

Wie so ziemlich alles im Internet müssen natürlich auch die Nachrichten, die Client und Server austauschen, in einem standardisierten Format sein Bei Verwendung von RSA als assymetrischen Verfahren legt der von RSA Data Security Inc. entwickelte Standard PKCS #1 fest, in welcher Form der Schlüssel für das symmetrische Verfahren übermittelt wird. In seiner alten Version „v1.5“, die auf Grund einer im nächsten Abschnitt beschriebenen Schwäche inzwischen nicht mehr empfohlen wird und durch eine Alternative ersetzt ist, geht man folgendermaßen vor:

Wird RSA mit einem N-Bit-Modul M verwendet, so sei zunächst $k = [N/8]$ der bei ganzzahliger Division mit ignoriertem Rest entstehende Quotient. Gesendet werden jeweils Blöcke aus $N + 1$ Bytes.

Da nicht alle durch $N+1$ Bytes darstellbare natürliche Zahlen kleiner als M sind, läßt sich das erste Byte nicht wirklich nutzen, denn die Verschlüsselung soll selbstverständlich injektiv sein. Daher wird dieses Byte stets auf 00 gesetzt.

Das nächste Byte gibt an, worum es sich bei dem zu übermittelnden Block handelt. Im Falle einer RSA-verschlüsselten Nachricht wird es auf 02 gesetzt, während beispielsweise 01 für eine elektronische Unterschrift steht.

Danach folgt ein Block von mindestens acht Zufallsbytes, die alle einen von Null verschiedenen Wert haben müssen; das Ende dieses Blocks wird durch ein angehängtes Nullbyte angezeigt. Die restlichen Bytes sind für die eigentliche Nachricht vorgesehen.

Auf den ersten Blick sieht es so aus, als würden die Zufallsbytes nur die Bandbreite verringern, tatsächlich aber erfüllen sie auch unter Sicherheitsaspekten einen wichtigen Zweck: Gelegentlich wird ein Angreifer in der Lage sein, den Inhalt einer Nachricht auf einige wenige Varianten einzuschränken: Das könnte im allertrivialsten Fall ein bloßes „Ja“ oder „Nein“ sein, aber auch in realistischeren Fällen wie bei stark normierten elektronischen Zahlungsverkehr gibt es eine ganze Reihe von Situationen, wo fast alles feststeht oder aber wo es nur darum geht, ob und wann eine bestimmte Nachricht geschickt wurde.

In solchen Fällen erweist sich der Vorteil der asymetrischen Kryptographie, daß nämlich jeder eine Nachricht an den Empfänger verschlüsseln kann, als Nachteil: Ein Lauscher muß einfach sämtliche in Frage kommenden Nachrichten mit dem öffentlichen Schlüssel des Empfängers kodieren und das Ergebnis mit der abgefangenen Nachricht vergleichen.

Als Schutz dagegen schlugen GOLDWASSER und MICALI bereits 1984 die sogenannte *probabilistische Verschlüsselung* vor, bei der nur ein Teil des Blocks für die eigentliche Nachricht genutzt wird, während der Rest mit Zufallszahlen aufgefüllt wird. Ein Angreifer, der eine Nachricht errät, muß nun nicht mehr einfach diese Nachricht verschlüsseln, sondern er muß gleichzeitig *alle* Möglichkeiten für die Zufallsbytes mitberücksichtigen. Bei acht von Null verschiedenen Zufallsbytes sind das $255^8 \approx 2^{63{,}95}$ Möglichkeiten, der Aufwand entspricht also etwa dem für

das Knacken einer symmetrischen Chiffre mit 64 Bit-Schlüssel. Da so etwas heute nicht mehr als sicher gelten kann, sollte man – falls die zu schützende Kommunikation anfällig gegen solche Angriffe ist – heute eher sechzehn als acht Zufallsbytes wählen.

Ein angenehmer Nebeneffekt der probabilistischen Kryptographie besteht noch darin, daß ein Angreifer jetzt auch nicht mehr einfach durch Vergleich der Chiffreblöcke feststellen kann, ob eine Nachricht Blöcke enthält, die bereits in einer ebenfalls abgefangenen früheren Nachricht auftauchten.

c) Der Angriff von Bleichenbacher

Bei Verwendung eines Kodierungsverfahrens wie dem gerade beschriebenen Standard entspricht nicht mehr jede Zahl $0 \le a \le M-1$ einer Nachricht. Damit kann es vorkommen, daß z.B. durch Übertragungsfehler ein Empfänger Nachrichten enthält, deren Entschlüsselung sich nicht sinnvoll entsprechend der Norm interpretieren läßt.

Ein menschlicher Empfänger wird solche Nachrichten, insbesondere wenn sie gehäuft auftreten, wohl einfach ignorieren oder vielleicht auch beim ersten Mal noch eine Nachricht an den Absender schicken, daß er dessen Nachricht nicht lesen kann; ein Server, der solche Nachrichten im Rahmen eines SSL-Verbindungsaufbaus erhält, wird jedes Mal genau das tun, was der Programmierer für diesen Fall vorgesehen hat. Früher war dies die kanonische Reaktion, die man in solchen Fällen erwartet: eine Fehlermeldung.

Eine solche Fehlermeldung kann ein Angreifer als eine Art *Orakel* benutzen: Wenn er eine Zahl $0 \le c \le M-1$ an den Server schickt, interpretiert dieser dies als eine verschlüsselte Nachricht $c = a^e \bmod M$ und entschlüsselt sie als $a = c^d \bmod M$, wobei (M, e) der öffentliche und d der private Schlüssel des Servers ist. Falls a nicht die erwartete Form hat, kann der Server die Nachricht nicht interpretieren und schickt eine Fehlermeldung zurück. Da Computer sehr geduldig sind, wird er dies nicht nur einmal, sondern gegebenenfalls auch mehrere Millionen Mal für denselben Absender tutn, so daß dieser beliebig viele Zahlen c testen kann.

Hier setzt der Angriff von BLEICHENBACHER an: Man kann zeigen, daß bei RSA jedes einzelne Bit so sicher ist wie der gesamte Block; mit anderen Worten: Falls jemand ein Verfahren hat, mit dem er ein festes Bit der Nachricht, z.B. das letzte oder das dritte, berechnen kann, so kann er daraus ein Verfahren machen, um die gesamte Nachricht zu entschlüsseln. Die wesentliche Idee des Beweis besteht darin, daß die RSA-Verschlüsselung $a \mapsto a^e \bmod M$ ein Gruppenhomomorphismus ist, was wir ja bereits bei den blinden Unterschriften und beim elektronischen Bargeld ausgenutzt hatten.

PKCS #1v1.5 liefert einem Angreifer, der den Server als Orakel nutzen kann, so etwas ähnliches wie die Entschlüsselung gewisser Bits: Er kann für gewisse Zahlen $0 \leq c \leq M-1$ erfahren, daß sie mit den beiden Bytes 00 und 02 beginnen. Er weiß dann also, daß $c^d \bmod M$ in einem gewissen Intervall $[B_1, B_2]$ liegt, wobei wir etwa

$$B_1 = 2^{[N/8]+1} + 2^{[N/8]} \quad \text{und} \quad B_2 = 3 \cdot 2^{[N/8]} - 1$$

setzen können. Falls bekannt ist, wie viele Zufallsbytes verwendet werden, kann man eventuell auch B_2 noch etwas schärfer abschätzen, andererseits bringt das nicht sonderlich viel. BLEICHENBACHER begnügt sich sogar bei der unteren Grenze einfach mit $B_1 = 2^{[N/8]+1}$.

Ziel der Attacke von BLEICHENBACHER ist es, zu einer vorgegebenen Zahl $0 \leq c \leq M-1$ die Zahl $c^d \bmod M$ zu bestimmen, um entweder eine abgefangene Chiffretextnachricht c wie den Sitzungsschlüssel für eine SSL-Verbindung zu entschlüsseln oder aber die Unterschrift des Servers für eine konstruierte Nachricht zu fälschen.

Falls c ein abgefangener Chiffretext ist, muß $c^d \bmod M$ in $[B_1, B_2]$ liegen. Andernfalls sucht der Angreifer nach einer Zahl s derart, daß für $c_0 = cs^e \bmod M$ die Entschlüsselung

$$c_0^d \bmod M = \left(cs^e\right)^d \bmod M = c^d s^{ed} \bmod M = c^d s \bmod M$$

vom Server akzeptiert wird.

Die Wahrscheinlichkeit dafür, daß dies für ein zufällig gewähltes s der Fall ist, läßt sich einigermaßen abschätzen: Wie können davon ausgehen, daß für zufälliges s auch die Zahlen $c_0^d \bmod M$ zufallsverteilt sind im

Intervall $[0, M-1]$. Die Wahrscheinlichkeit dafür, daß eine solche Zahl im Intervall $[B_1, B_2]$ liegt ist daher das Verhältnis der Intervallängen, also ungefähr $M/2^{[N/8]}$. Je nach Kongruenzklasse der Bitanzahl von M modulo acht liegt dieser Wert zwischen $2^{-16} = 1 : 65536$ und $2^{-8} = 1 : 256$; da die Bitlängen von RSA-Moduln meist Vielfache von acht sind, dürfte sie sich eher in der Nähe der unteren Grenze bewegen. Dazu kommt noch, daß auf die beiden Bytes 00 und 02 mindestens acht von Null verschiedene Bytes folgen müssen und dann irgendwann ein Nullbyte, was die Wahrscheinlichkeit der Akzeptanz noch etwas weiter verringert, wenn auch um keinen sonderlich großen Faktor: Falls wir etwa mit 2048 Bit-Modul arbeiten, besteht ein Block aus 256 Bytes; wir können also mit einer ziemlich hohen Wahrscheinlichkeit davon ausgehen, daß irgendeines davon zwischen den Positionen elf und 256 das Nullbyte ist.

Bei einem automatisierten Angriff kann man somit in relativ kurzer Zeit eine Zahl s finden, so daß $c_0 = cs^e \bmod M$ vom Server akzeptiert wird. Falls man dann $a_0 = c_0^d \bmod M$ bestimmen kann, läßt sich leicht auch

$$a = c^d \bmod M = s^{-1} a_0 \bmod M$$

berechnen. Wir können uns daher im folgenden auf das Problem beschränken, zu einem c, das einer korrekt verschlüsselten Nachricht a entspricht, deren Entschlüsselung $a = c^d \bmod M$ zu ermitteln.

Dazu bestimmt BLEICHENBACHER, wieder durch Probieren so lange, bis der Server keine Fehlermeldung mehr schickt, eine Folge von Zahlen

$$0 < s_1 < s_2 < \cdots < M$$

derart, daß $c_i = cs_i^e \bmod M$ vom Server akzeptiert wird. Dann ist

$$a_i = c_i^d \bmod M = as_i \bmod M \in [B_1, B_2]i\,,$$

es gibt also eine Zahl r_i derart, daß

$$as_i - r_i M \in [B_1, B_2] \qquad \text{oder} \qquad a \in \left[\frac{B_1 + r_i M}{s_i}, \frac{B_2 + r_i M}{s_i}\right]$$

Damit liegt a auch im Durchschnitt eines dieser Intervalle mit $[B_1, B_2]$, so daß wir a weiter eingegrenzt haben.

Im Hinblick auf die weiteren Schritte wollen wir annehmen, wir wüßten bereits, daß a in einem Intervall $[u, v]$ liegt. Dann können wir nun genauer sagen, daß a sogar im Durchschnitt von $[u, v]$ mit der Vereinigung der obigen Intervalle liegt, d.h. in der Vereinigung

$$\bigcup_{r\in\mathbb{Z}} \left(\left[\frac{B_1 + rM}{s_i}, \frac{B_2 + rM}{s_i} \right] \cap [u, v] \right) .$$

Tatsächlich sind natürlich fast alle diese Durchschnitte leer; ein nichtleerer Durchschnitt ist nur möglich, wenn

$$\frac{B_1 + rM}{s_i} \leq v \quad \text{und} \quad \frac{B_2 + rM}{s_i} \geq u ,$$

also

$$\frac{s_i u - B_2}{M} \leq r \leq \frac{s_i v - B_1}{M}$$

ist.

Damit ist BLEICHENBACHERs Vorgehensweise zumindest im Prinzip klar: Wir betrachten eine Menge L von Intervallen derart, daß a in einem Intervall aus der Liste sein muß; zu Beginn besteht L genau aus dem Intervall $[B_1, B_2]$. Außerdem setzen wir $s_0 = [M/B_2]$; man überzeugt sich leicht, daß sa für $s \leq s_0$ höchstens gleich M aber natürlich größer als B_2 ist, so daß as dann unmöglich akzeptiert werden kann.

Im i-ten Schritt für $i \geq 1$ wird durch Serveranfragen eine Zahl $s_i > s_{i-1}$ ermittelt derart, daß as_i in $[B_1, B_2]$ liegt; sodann wird L ersetzt durch die Menge aller Intervalle der Form

$$\left[\frac{B_1 + rM}{s_i}, \frac{B_2 + rM}{s_i} \right] \cap [u, v] \quad \text{mit} \quad [u, v] \in L \quad \text{und} \quad \frac{s_i u - B_2}{M} \leq r \leq \frac{s_i v - B_1}{M} .$$

Dieses Verfahren wird so lange forgesetzt, bis L nur noch ein Intervall der Länge eins enthält, das dann notwendigerweise gleich $[a, a]$ ist.

Tatsächlich optimiert BLEICHENBACHER noch etwas: Durch geschickte Wahl von s_i kann man nämlich r noch etwas genauer unter Kontrolle bekommen:

Rest noch nicht geschrieben

§8: Literatur

RSA ist immer noch das gebräuchlichste Verfahren der *public key* Kryptographie; es gibt daher 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 RSA 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/~weng/schabhueser_ECC_2002.` $\left\{\begin{array}{l}\texttt{PDF}\\ \texttt{ppt}\end{array}\right.$

Kapitel 6
Verfahren mit diskreten Logarithmen

Die Sicherheit des RSA-Verfahrens hängt zusammen mit der Schwierigkeit der Zerlegung großer Zahlen in ihre Primfaktoren. Eine zweite vor allem für elektronische Unterschriften populäre Gruppe von Verfahren baut stattdessen auf die Schwierigkeit der Umkehrung von Exponentialfunktionen mit Werten in der multiplikativen Gruppe eines endlichen Körpers oder auch in gewissen anderen Gruppen; diese Umkehrfunktion bezeichnet man in Analogie zur Umkehrfunktion einer reellen Exponentialfunktion als diskreten Logarithmus (oder Index). Wir beginnen mit dem ältesten Beispiel eines solchen Verfahrens:

§1: Schlüsselaustausch nach Diffie und Hellman

Wie wir im letzten Kapitel gesehen haben, waren DIFFIE und HELLMAN mit ihrer Arbeit *New directions in cryptography* die Initiatoren der asymmetrischen Kryptographie in der akademischen Welt; kurz nach dieser Arbeit entwickelten sie auch ein Verfahren dazu, das zwar nicht zur Verschlüsselung dienen konnte, aber dafür die bis heute wichtigste Aufgabe der Kryptographie mit öffentlichen Schlüssel lösen konnte: Die Vereinbarung eines Schlüssels über eine unsichere Leitung.

Im Gegensatz zum RSA-Verfahren brauchen sie dazu nicht einmal öffentliche Schlüssel: Die beiden Teilnehmer können miteinander sicher kommunizieren ohne zuvor irgendwelche öffentlichen oder privaten Schlüssel zu kennen. Damit ist dieses Verfahren vor allem interessant im privaten Bereich, wo zertifizierte öffentliche Schlüssel oder gelegentlich auch überhaupt die Speicherung von Schlüsseln zu aufwendig wäre.

a) Das Verfahren

Die beiden Teilnehmer 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 ein $y < p$; A schickt $u = a^x \bmod p$ an B und erhält dafür $v = a^y \bmod p$ von diesem.

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; er kann also problemlos alle möglichen Zahlen der Art

$$a^{\alpha x+\beta y} = u^\alpha \cdot v^\beta$$

berechnen, aber es fällt schwer, sich eine Art uns Weise vorzustellen, wie er $a^{xy} \bmod p$ finden kann, ohne den diskreten Logarithmus von u oder v berechnen. (Bewiesen ist hier, wie üblich, natürlich nichts.)

b) Die *man in the middle attack*

Da der Gegner die Wahl der Mittel hat, muß er aber natürlich nicht notwendigerweise die mathematische Seite des Verfahrens angreifen: Er kann angreifen, was immer er für die vielversprechendste Schwachstelle hält.

Nehmen wir etwa an, der Gegner habe eine gewisse Kontrolle über das Netz, in dem der Datenaustausch stattfindet – beispielsweise, weil er Systemverwalter eines für die betreffende Verbindung unbedingt notwendigen Knotenrechners ist. Dann kann er eine sogenannte *man in the middle attack* durchführen: Er fängt alle Datenpakete zwischen A und B ab und ersetzt sie durch selbstfabrizierte eigene Pakete.

Dies erlaubt ihm, sich gegenüber A als B auszugeben und umgekehrt: Alles, was A an B zu schicken glaubt, geht tatsächlich an den Gegner G, und auch alles was B von A zu erhalten glaubt, kommt tatsächlich von G. In Gegenrichtung ist es natürlich genauso.

Im Einzelnen läuft das Verfahren dann folgendermaßen ab:

Falls die Zahlen a und p nicht ohnehin Konstanten eines Verbunds sind, dem A und B angehören, läßt G die Kommunikation, die zu deren Vereinbarung führt, ungehindert zu: In diesem Stadium beschränkt er sich auf reines Abhören.

Als nächstes wählen A und B ihre Zufallszahlen $x < p$ und $y < p$; gleichzeitig wählt G eine Zufallszahl $z < p$ oder vielleicht auch zwei solche Zahlen z_A und z_B für jeden der beiden Teilnehmer.

Wenn A $u = a^x \bmod p$ an B schickt, fängt G diese Nachricht ab und ersetzt sie durch $w_A = a^{z_B} \bmod p$; entsprechend fängt er Bs Nachricht $v = a^y \bmod p$ ab und schickt stattdessen $w_B = a^{z_A}$ an A. Dies führt dazu, daß nun A und G einen gemeinsamen Schlüssel s_A haben und B und G einen gemeinsamen Schlüssel s_B. Sowohl A als auch B glauben, der ihnen bekannte Schlüssel s_A *bzw.* s_B sei aus $a^{xy} \bmod p$ abgeleitet und senden nun Nachrichten damit an ihren Partner. Diese Nachrichten fängt G ab, entschlüsselt sie mit dem Schlüssel, den er mit dem Absender gemeinsam hat, und verschlüsselt sie anschließend, gegebenenfalls nach einer seiner Interessen entsprechenden Modifikation, mit dem Schlüssel, den er mit dem Empfänger gemeinsam hat. Auf diese Weise hat er die gesamte Konversation unter Kontrolle, ohne daß A und B etwas merken.

Die Möglichkeit für diese Attacke kommt natürlich daher, daß sich A und B nicht sicher sein können, den jeweils anderen am anderen Ende der Leitung zu haben. Die kryptographisch einwandfreie Modifikation, die das Verfahren gegen diese Art von Angriff sicher macht, bestünde beispielsweise darin, daß A und B ihre Nachrichten u und v vor dem Versenden unterschreiben – aber dann verschwindet auch wieder der Vorteil, daß sie ohne Kenntnis irgendeines Schlüssels miteinander kommunizieren können: Zur Verifikation einer Unterschrift brauch man schließlich den öffentlichen Schlüssel des Unterschreibenden.

Falls sich A und B hinreichend gut kennen, um die Stimme des jeweils anderen am Telephon einigermaßen gut zu erkennen, können sie diese Art von Attacke auch dadurch erschweren, daß sie nach dem Austausch von u und v per Telephon über diese Zahlen (z.B. die 317. bis 320. Ziffer) und gegebenenfalls auch noch über Schwänke aus ihrer gemeinsamen Jugendzeit reden; dann müßte der Angreifer zusätzlich noch ein begabter, kundiger und reaktionsschneller Stimmenimitator sein, der auch die Telephonverbindung als *man in the middle* so angreifen kann, daß weder A noch B etwas merkt. Bei Videokonferenzen könnte man auch die Zahlen langsam über den Bildschirm des jeweils anderen laufen lassen. Die volle Sicherheit einer Schlüsselvereinbarung via RSA wird aber nicht erreicht, und da oft zumindest einer der Teilnehmer ein Unternehmen ist, das sich einen zertifizierten RSA-Modul leisten kann, werden Schlüssel für symmetrische Kryptoverfahren in der Praxis sehr viel häufiger via RSA vereinbart als via DIFFIE-HELLMAN.

c) Wie sind die vereinbarten Schlüssel verteilt?

Möglicherweise hat ein Gegner noch andere Angriffspunkte: Die Sicherheit von AES wie von jedem anderen symmetrischen Kryptoverfahren ist unter anderem wesentlich davon der Tatsache abhängig, daß die Schlüssel zufällig gewählt werden: Jede Inhomogenität ihrer Verteilung reduziert die Schlüsselentropie und erhöht die Chancen eines Angreifers, schneller als durch systematisches Ausprobieren aller Schlüssel die Nachricht zu entschlüsseln.

Noch nicht geschrieben und auch in der Vorlesung nicht behandelt

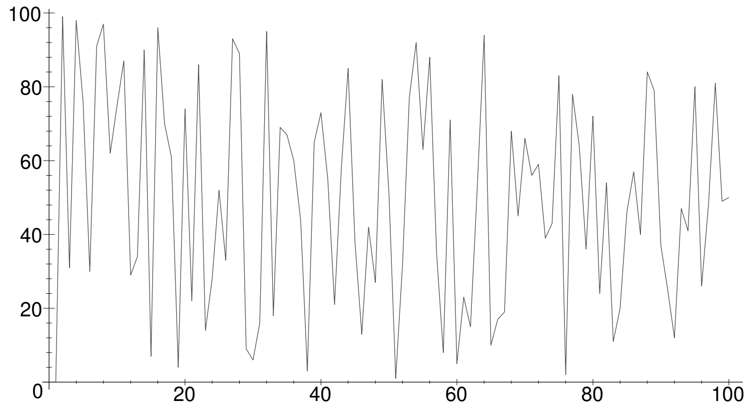
§2: Diskrete Logarithmen

a) Logarithmen in endlichen Körpern

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.

b) Wie groß sollten die Körper sein?

Da diskrete Logarithmen in der Praxis vor allem für elektronische Unterschriften verwendet werden, gibt die Regulierungsbehörde natürlich auch Empfehlungen für die Größe der hier zu verwendenden Primzahlen *bzw.* Zweierpotenzen. Zumindest bisher war hier immer dieselbe Bitzahl gefordert *bzw.* empfohlen wie für RSA-Moduln des entsprechenden Geltungszeitraums. Dieses Jahr wird es wahrscheinlich die erste Ausnahme geben: Für Gültigkeit bis Ende 2010 wird nach dem bislang vorliegenden (zweiten) Entwurf für RSA eine Mindestlänge von 1984 Bit gefordert, für Verfahren auf der Grundlage diskreter Logarithmen aber 2048 Bit. Der Grund dafür liegt allerdings nicht in unterschiedlichen Gefährdungspotentialen: Im ersten Entwurf waren auch für RSA noch 2048 Bit gefordert worden. Die Reduktion um 64 Bit auf 1984 geht wohl zurück auf eine Intervention des Zentralen Kreditausschuß, einem Dachverband der Banken, der auf den von ihm ausgegebenen Chipkarten zwar RSA verwendet, aber wegen der Grenzen des derzeit verwendeten Betriebssystems SECCOS dort keine größeren Schlüssellängen als 1984 Bit verwenden kann, und selbst die nur mit Schwierigkeiten. Da Chipkarten eine Gültigkeitsdauer von bis zu vier Jahren haben und die Herstellung einer großen Anzahl neuer Karten zwei bis drei Jahre Vor-

laufzeit erfordert, sahen sich die Banken nicht in der Lage, auf allen bis mindestens Ende 2010 gültigen Karten RSA-Modul von 2048 Bit zu realisieren. Diese Intervention führte anscheindend zur Änderung des ersten Entwurfs des Bundesamts für Sicherheit in der Informationstechnik, die wohl auch die Regulierungsbehörde für Telekommunikation und Post übernehmen wird – wenn sie nicht einen noch niedrigeren Wert wählt. Den Banken zumindest wären 1736 Bit deutlich lieber als 1984.

c) Der allgemeine diskrete Logarithmus

2048 Bit-Zahlen sind bereits ziemlich unhandlich, und selbst 1024 Bit-Zahlen sind nicht gerade sonderlich bequem. Daher suchen Kryptologen bereits seit langer Zeit nach Alternativen. Die derzeit interessantesten (und zunehmend, vor allem in USA, bereits in der Praxis eingesetzten) Verfahren beruhen auf einer Verallgemeinerung diskreter Logarithmen:

Ist G irgendeine Gruppe und $g \in G$, so können wir die Abbildung

$$\varphi_g\colon \mathbb{Z} \to G; \quad n \mapsto g^n$$

betrachten. Falls man in der Gruppe G überhaupt konkret rechnen kann, lassen sich die Werte $\varphi_g(n) = g^n$ nach dem üblichen Verfahren durch Quadrieren und Multiplizieren mit einem Aufwand in der Größenordnung $\log_2 n$ berechnen.

Die Umkehrfunktion $\varphi_g^{-1}\colon \operatorname{Bild} \varphi_g \to \mathbb{Z}/\operatorname{Kern} \varphi_g$ bezeichnen wir auch hier als einen diskreten Logarithmus; falls er hinreichend schwer zu berechnen ist, eignet er sich als Grundlage für Kryptosysteme.

Das Bild von φ_g besteht offensichtlich genau aus den Potenzen von g, ist also eine zyklische Gruppe. Damit können wir uns ohne Beschränkung der Allgemeinheit auf zyklische Gruppen beschränken. Da diese stets abelsch sind und die Verknüpfung in abelschen Gruppen traditionellerweise eher mit „+“ als mit „·“ bezeichnet wird, schreiben wir auch hier die Verknüpfung additiv; dann ist also

$$\varphi_g(n) = ng = \underbrace{g + \cdots + g}_{n \text{ mal}}$$

für $n > 0$ und für $n < 0$ ist $\varphi_g(n) = -\varphi_g(-n)$. Für $n = 0$ definieren wir natürlich $\varphi_g(0) = 0$ als das neutrale Element der Gruppe, das wir im additiven Fall als „0“ schreiben.

(Man beachte, daß dieses Neutralelement 0 im Falle der multiplikativen Gruppe eines Körpers natürlich die Eins ist. Die Addition „+“ der abstrakten Gruppe G entspricht dann der Körpermultiplikation, und das Neutralelement $0 \in G$ ist die Körpereins. Das erscheint zwar auf den ersten Blick etwas verwirrend, man sollte sich aber schnell daran gewöhnen können.)

d) Die Struktur zyklischer Gruppen

G sei eine (additiv geschriebene) endliche zyklische Gruppe, und $g \in G$ sei ein erzeugendes Element, d.h. jedes Element $a \in G$ läßt sich in der Form $a = xg$ schreiben mit einer ganzen Zahl $0 \le x \le n-1$, wobei n die Elementanzahl von G bezeichnet.

Wir zerlegen n in seine Primfaktoren

$$n = \prod_{i=1}^{r} p_i^{e_i} \quad \text{und setzen} \quad n_i = \frac{n}{p_i}, \quad g_i = n_i g\,.$$

Da g die Ordnung n hat, ist die Ordnung von g_i gleich $n/n_i = p_i^{e_i}$, das Element g_i erzeugt also eine zyklische Gruppe G_i der Ordnung $p_i^{e_i}$. Für jedes Element $a = xg \in G$ ist

$$n_i a = n_i(xg) = (n_i x)g = (xn_i)g = x(n_i g) = xg_i \in G_i\,.$$

Wir können daher eine Abbildung $\varphi\colon G \to G_1 \times \cdots \times G_r$ definieren durch

$$a \mapsto (n_1 a, \ldots, n_r a)\,.$$

Für ein Element $a \in G$ aus dem Kern von φ ist $n_i a = 0$ für alle i. Der größte gemeinsame Teiler der sämtlichen n_i ist, wie deren Primfaktorzerlegung zeigt, gleich eins, es gibt also ganze Zahlen $\alpha_1, \ldots, \alpha_r$ mit

$$\sum_{i=1}^{r} \alpha_i n_i = 1 \quad \text{und} \quad a = \left(\sum_{i=1}^{r} \alpha_i n_i\right) a = \sum_{i=1}^{r} \alpha_i (n_i a) = 0\,.$$

Also ist die Abbildung injektiv, und da G und das Produkt der G_i die gleiche Mächtigkeit n haben, ist sie auch surjektiv, also ein Isomorphismus.

Dieser Isomorphismus führt sofort zu einem Algorithmus zur Berechnung der Ordnung eines Elements von $a \in G$: Diese ist nach dem Satz von LAGRANGE ein Teiler der Gruppenordnung, hat also die Form

$$\operatorname{ord} a = \prod_{i=1}^{r} p_i^{s_i} \quad \text{mit} \quad 0 \le s_i \le e_i\,.$$

Mit $a_i = n_i a$ ist dann $p_i^{s_i} a_i = n_i p_i^{s_i} a = 0$, da der Multiplikator $n_i p_i^{s_i}$ ein Vielfaches der Ordnung ist. Da dies für keine kleinere p-Potenz gilt, ist $p_i^{s_i}$ die genaue Ordnung von a_i, und diese Ordnung kann berechnet werden, indem wir so lange mit p multiplizieren, bis wir zum ersten Mal das Neutralelement Null erhalten. Die Ordnung von a ist dann also das Produkt der so berechneten Ordnungen der a_i. Man beachte, daß wir für diese Rechnung in der Lage sein müssen, die Gruppenordnung zu faktorisieren!

Der obige Isomorphismus erlaubt es uns auch, die Berechnung diskreter Logarithmen in G auf das entsprechende Problem in den G_i zu reduzieren: Angenommen, wir kennen die diskreten Logarithmen x_i der Elemente $n_i a$ zu den Basen $n_i g$. Dann gilt für den diskrete Logarithmus x von a zur Basis g

$$xg = a \Longrightarrow x(n_i g) = n_i a \Longrightarrow xg_i = a_i \Longrightarrow x \equiv x_i \bmod p_i^{e_i}\,,$$

denn g_i hat ja die Ordnung $p_i^{e_i}$.

Die Kongruenz $x \equiv x_i \bmod p_i^{e_i}$ gilt für alle i, und damit können wir x nach dem chinesischen Restesatz berechnen sobald wir die x_i kennen. Der chinesische Restesatz liefert uns $x \bmod n$, aber das reicht, da g ja gerade die Ordnung n hat.

Die Berechnung diskreter Logarithmen in G kann also problemlos zurückgeführt werden auf die Berechnung diskreter Logarithmen in den maximalen Untergruppen von Primzahlpotenzordnung.

e) Das Verfahren von Pohlig und Hellman

Tatsächlich reicht es sogar, wenn wir das diskrete Logarithmenproblem statt in Gruppen der Ordnung $p_i^{e^i}$ in Gruppen der Ordnung p_i lösen können. Dazu gehen wir folgendermaßen vor:

Der Einfachheit halber beschränken wir uns auf eine zyklische Gruppe G von Primzahlpotenzordnung p^e und wählen dort ein erzeugendes Element g. Gesucht ist der diskrete Logarithmus eines weiteren Elements $a \in G$ zur Basis g.

Diese gesuchte Zahl x liegt zwischen null und $p^e - 1$; wenn wir sie in Ziffern zur Basis p schreiben, ist also

$$x = x_0 + x_1 p + x_2 p^2 + \cdots + x_{e-1} p^{e-1} \quad \text{mit} \quad 0 \le x_i \le p\,.$$

Wir wollen uns überlegen, daß wir die „Ziffern“ x_i als diskrete Logarithmen in einer Untergruppe der Ordnung p berechnen können.

Aus $a = xg$ folgt die Beziehung

$$p^j a = p^j xg = x_0 p^j g + x_1 p^{j+1} g + x_2 p^{j+2} g + \cdots + x_{e-1} p^{j+e-1} g\,.$$

Da aber $p^e g = 0$ ist, verschwinden hier alle Terme, in denen p einen größeren Exponenten als e hat; tatsächlich ist also

$$p^j a = p^j xg = x_0 p^j g + x_1 p^{j+1} g + x_2 p^{j+2} g + \cdots + x_{e-j-1} p^{e-1} g\,.$$

Insbesondere folgt für $j = e - 1$, daß $p^{e-1} a = x_0 p^{e-1} g$ ist, x_0 ist also die Lösung eines diskreten Logarithmenproblems in der von $p^{e-1} g$ erzeugten Untergruppe der Ordnung p.

Angenommen, wir haben die „Ziffern“ von x_0 bis x_r bereits bestimmt. Dann schreiben wir

$$a - x_0 g - x_1 pg - \cdots - x_r p^r g = x_{r+1} p^{r+1} g + \cdots + x_{e-1} p^{e-1} g$$

und multiplizieren diese Gleichung mit p^{e-2-r}. Dann verschwinden rechts alle Terme außer dem ersten, wir erhalten also die Gleichung

$$p^{e-2-r}(a - x_0 g - x_1 pg - \cdots - x_r p^r g) = x_{r+1}(p^{e-1} g)\,,$$

die uns x_{r+1} als diskreten Logarithmus der (bekannten) linken Seite liefert, und zwar wieder in der von $p^{e-1} g$ erzeugten Untergruppe der Ordnung p.

Auf diese Weise können wir nacheinander die sämtlichen x_i berechnen und damit auch x.

Zusammen mit dem oben diskutierten Ansatz über den chinesischen Restesatz ist dies das Verfahren von POHLIG und HELLMAN zur Berechnung diskreter Logarithmen in einer zyklischen Gruppe G der Ordnung n: Sie kann zurückgeführt werden auf die Berechnung diskreter Logarithmen in Untergruppen, deren Ordnungen die Primteiler von n sind.

Dies legt nahe, daß bei Kryptosystemen auf der Basis diskreter Logarithmen die Sicherheit nicht so sehr von der Ordnung n der Gruppe G abhängt, sondern von deren größtem Primteiler.

f) Folgerungen für Kryptosysteme über endlichen Körpern

Wenn wir für G die multiplikative Gruppe eines endlichen Körpers $\mathbb{F}_q$ nehmen, ist deren Ordnung gleich $q - 1$. Zur Anwendung des Verfahrens von DIFFIE und HELLMAN sowie auch für alle anderen noch zu diskutierenden Verfahren auf der Basis diskreter Logarithmen benötigen wir zunächst ein Element $g \in \mathbb{F}_q^\times$ mit möglichst hoher Ordnung, idealerweise eine primitive Wurzel.

Derzeit ist dazu kein besseres Verfahren bekannt als das Ausprobieren verschiedener Elemente g solange, bis man eines mit hinreichend hoher Ordnung gefunden hat, im Idealfall der Ordnung $q - 1$. Das nächste Problem besteht also darin, die Ordnung eines Elements $g \in \mathbb{F}_q^\times$ zu bestimmen.

Das beste dazu bekannte Verfahren ist leider das oben diskutierte: Wir müssen also die Gruppenordnung

$$q - 1 = \prod_{i=1}^{r} p_i^{e_i}$$

als ein Produkt von Primzahlpotenzen schreiben und dann die Ordnungen $p_i^{s_i}$ der Elemente

$$g_i = g^{n_i} \quad \text{mit} \quad n_i = \frac{q-1}{p_i^{e_i}}$$

bestimmen. Deren Produkt ist die Ordnung von g.

Da q von derselben Größenordnung ist wie ein RSA-Modul, erscheint die Faktorisierung von $q-1$ auf den ersten Blick hoffnungslos, allerdings ist $q-1$ im Gegensatz zu einem RSA-Modul kein Produkt zweier ungefähr gleich großer Primzahlen, sondern kann durchaus, wenn wir Glück haben, in ein Produkt von Potenzen relativ kleiner Primzahlen zerfallen. Dann ist die Faktorisierung von $q-1$ leicht, leider aber auch jedes Verfahren auf der Grundlage diskreter Logarithmen unsicher: Wie der Algorithmus von POHLIG und HELLMAN zeigt, sind diskrete Logarithmen dann ziemlich einfach zu berechnen.

Am sichersten sind wir also, wenn $q-1$ eine Primzahl ist, und dann haben wir auch keine Probleme mit der Faktorisierung. $q-1$ kann natürlich nur dann eine Primzahl sein, wenn q eine Zweierpotenz ist; alsdann ist $q-1=2^n-1$ eine MERSENNEsche Primzahl. Insbesondere ist also auch n eine Primzahl, so daß es nicht viel Auswahl gibt: Schließlich sind bislang erst 41 MERSENNEsche Primzahlen bekannt, und die meisten davon sind entweder zu groß oder zu klein, um hier in Frage zu kommen: Zwischen 1024 und 4048 Bit gibt es nur die vier Exponeten 1279, 2203, 2281 und 3217, die zu Primzahlen führen. Es gibt natürlich noch andere Exponenten n, für die w^n-1 mindestens einen großen Primfaktor hat, aber sehr groß ist die Auswahl nicht, und das macht es einem Gegner auch relativ einfach, für die interessanten Körper Spezialhardware zur Berechnung diskreter Logarithmen zu entwerfen: Für die meisten interessanten Exponenten existieren entsprechende Chips wenigstens als Entwurf. Von daher hat ein Gegner bei Körpern von Zweipotenzordnung geringfügige Vorteile, allerdings sind die Empfehlungen zur Bitlänge der Ordnung natürlich so, daß sie auch gegen Angriffe mit Spezialhardware sicher sein sollten.

Abgesehen von Körpern von Zweipotenzordnung sind vor allem Körper von Primzahlordnung interessant: Potenzen ungerader Primzahlen bieten keine Vorteile, dafür aber bei der Implementierung der Körpermultiplikation beträchtliche Nachteile gegenüber solchen von Primzahlordnung.

Hier hat die multiplikative Gruppe die gerade Ordnung $p-1$; das beste, was wir erreichen können, wäre also, daß $p-1$ das Doppelte einer Primzahl ist. Dies hätte dann auch den Vorteil, daß es keine Probleme mit

der Suche nach einer geeigneten Basis gibt: Falls $p-1=2q$ das Doppelte einer Primzahl ist, können Elemente von $\mathbb{F}_p^\times$ nur die Ordnungen $1, 2, q$ und $2q$ haben; man hat also eine primitive Wurzel gefundenen, sobald man ein Element a hat, dessen Quadrat und p-te Potenz beide von der Eins verschieden sind, wobei a^2 natürlich nur für ± 1 gleich Eins sein kann. Da Ordnung $2q$ wegen des Verfahrens von POHLIG und HELLMAN keine nennenswert größere Sicherheit bietet als q, kann man auch einfach jedes Element $a \neq \pm 1$ nehmen, denn dieses kann nur entweder q oder $2q$ als Ordnung haben.

Bleibt nur noch die Frage, ob es genügend viele Primzahlen p gibt, für die $p-1$ das Doppelte einer Primzahl ist. Theoretische Ergebnisse sind dazu nicht bekannt: Man weiß nicht einmal, ob es unendlich viele Primzahlen dieser Form gibt. Empirisch sind sie allerdings leicht zu finden, und in den Größenbereichen, in denem man dies bilsang untersucht hat, verhält sich ihre Dichte ungefähr wie $1 : (\log p)^2$. Dies ist die Zahl die man erhält, wenn man die naive Annahme macht, die Ereignisse „p prim“ und „$\frac{1}{2}(p-1)$ prim“ seien unabhängig voneinander und die Wahrscheinlichkeiten dafür einfach miteinander multipliziert.

Praktisch erhält man solche Primzahlen am effizientesten, indem man ein hinreichend großes Intervall zweimal à la ERATOSTHENES siebt: Sucht man eine entsprechende Primzahl oberhalb der Zahl $2N$, wendet man auf das Intervall $[N, N+\ell]$ zunächst das übliche Siebverfahren an, so dann siebt man ein zweites Mal, jetzt indem man alle Zahlen p streicht, für die $2p+1$ durch die betrachtete kleine Primzahl teilbar ist. Für die nichtgestrichenen Zahlen p wird dann ein Primzahltest sowohl auf p als auch auf $2p+1$ angewandt.

§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, diskrete Logarithmen zu berechnen. Wir gehen wieder aus von einer additiv geschriebenen abelschen Gruppe G und einem Element $g \in G$; gesucht ist zu einem vorgegebenen Element a aus der von g erzeugten zyklischen Gruppe die kleinste natürliche Zahl x, für die $g^x = a$ ist.

a) Probieren

Am einfachsten und langwierigsten ist Probieren: Um den diskreten Logarithmus von a in der von g erzeugten zyklischen Gruppe zu bestimmen, berechnet man einfach systematisch alle Vielfachen von g, bis man a erhält. Dies entspricht etwa dem Faktorisieren durch Abdividieren.

b) Das Verfahren von Pohlig und Hellman

Falls wir die Ordnung von g berechnen und faktorisieren können, reduziert die Methode von POHLIG und HELLMAN das Problem auf die Berechnung diskreter Logarithmen in Gruppen, deren Ordnungen Primteiler der ursprünglichen Gruppenorsdnung sind. Zumindest wenn einer dieser Primteiler sehr groß ist, haben wir damit allerdings nicht viel gewonnen; wir brauchen also auf jeden Fall noch andere Methoden.

c) Baby steps und giant steps

Ein solches Verfahren ist das *baby step – giant step* Verfahren von SHANKS, das auf Kosten von mehr Speicherplatz die Rechenzeit gegenüber reinem Probieren deutlich reduziert: Ist n die Ordnung von g, so ist der Aufwand nicht mehr proportional zu n, sondern nur noch zu $\sqrt{n}$.

Dazu wählt man eine natürliche Zahl m die ungefähr gleich $\sqrt{n} \cdot \log_2 n$ ist; falls man n nicht so genau kennt, kann zwar die Effiziens des Verfahrens unter einer schlechten Wahl von m geringfügig leiden, aber solange die Größenordnungen einigermaßen stimmen, ist das nicht so dramatisch.

Danach berechnet man die sämtlichen Vielfachen $i \cdot g$ von g für $i \leq m$; das sind m sogenannte *baby steps.*

Bei den dann folgenden *giant steps* berechnet man, um den diskreten Logarithmus von a zu erhalten, die Elemente $a - j \cdot mg$ für $j = 1, 2, \ldots$ und vergleicht sie mit den Vielfachen aus dem ersten Teil. Ein solcher

Vergleich kann etwa über eine binäre Suche oder eine *hash*-Tabelle implementiert werden und hat einen Aufwand proportional $\log_2 n$.

Sobald man einen Wert $a - j \cdot mg$ gefunden ist, der mit einem der in den *baby steps* berechneten Vielfachen $i \cdot g$ übereinstimmt, hat man die Gleichung

$$a - j \cdot mg = i \cdot g \qquad \text{oder} \qquad a = (mj + i) \cdot g \,,$$

der diskrete Logarithmus von a zur Basis g ist also $mj + i$. Die notwendige Anzahl von *giant steps* liegt im schlimmsten Fall bei $n/m \approx \sqrt{n}$; 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 $q \leq B$. 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

$$y \equiv e_1 \log_a q_1 + \cdots + e_r \log_a q_r \bmod m ,$$

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.

§3: Das Verfahren von ElGamal

Ab hier wird vieles umgestellt werden; das DSA aus dem nächsten Kapitel soll hierher. Die Nummerierung entspricht dem Skriptum von vor zwei Jahren und hat nichts mehr mit der aktuellen zu tun.

c) Verschlüsselung nach ELGAMAL

Das Verfahren von ELGAMAL ist eine einfache Variante des Verfahrens von DIFFIE-HELLMAN, die daraus ein Verschlüsselungsverfahren macht: 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 auch aus diesem Grund nicht attraktiv ist.

d) 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 $m^{e_A} \bmod p$. 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 $M^e \equiv m^{e_A} \bmod p$ 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.

§5: Elliptische Kurven

Hier zumindest am Anfang deutlich ausführlicher als in der Vorlesung, am Ende Lücken.

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, bald vielleicht 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 $x \geq 0$ 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 im Augenblick der einzige, der in der Praxis eine nennenswerte Rolle spielt) ist der der *elliptischen Kurven,* allerdings gibt es auch bereits Systeme auf der Grundlage der etwas komplizierteren sogenannten *hyperelliptischen* Kurven. Um beides definieren zu können, wollen wir uns zunächst allgemein mit ebenen Kurven beschäftigen.

a) Ebene algebraische Kurven

Zwei Geraden in der Ebene haben entweder einen Schnittpunkt, oder sie sind parallel. Entsprechend haben eine Gerade und ein Kreis entweder zwei Schnittpunkte, oder die Gerade ist Tangente, so daß es nur einen Schnittpunkt gibt, oder aber es gibt keinen Schnittpunkt. Bei zwei Ellipsen schließlich sind alle Schnittpunktanzahlen von null bis vier möglich, und je höher die Grade der Kurven werden, desto unübersichtlicher wird die Situation.

Zwei Zusatzvoraussetzungen und eine Definition helfen, diese Komplexität zu reduzieren.

Beginnen wir mit dem Problem, daß sich zwei Geraden in der Ebene nicht unbedingt schneiden müssen. Anschaulich sagt man oft, daß sich zwei parallele Geraden im Unendlichen schneiden, und das läßt sich auch mathematisch exakt formulieren, wenn wir übergehen zur *projektiven* Ebene.

Die Punkte der affinen Ebenen über einem Körper k können nach Wahl eines Koordinatensystems identifiziert werden mit den Elementen des Vektorraums k^2. Für die Definition der projektiven Ebenen $\mathbb{P}^2(k)$ über k gehen wir aus vom Vektorraum k^3 und bezeichnen dessen eindimensionale Untervektorräume als Punkte. Geometrisch betrachtet entsprechen die Punkte von $\mathbb{P}^2(k)$ also den Nullpunktgeraden in k^3.

Ein eindimensionaler Untervektorraum von k^3 wird erzeugt von einem Tripel $(x, y, z) \neq (0, 0, 0)$; wir bezeichnen x, y, z als die *homogenen Koordinaten* des Punktes. Sie sind natürlich nicht eindeutig bestimmt: Für jedes $\lambda \neq 0$ aus k definiert $(\lambda x, \lambda y, \lambda z)$ denselben Punkt. Der Einfachheit halber sprechen wir trotzdem vom Punkt $P = (x, y, z)$, müssen aber beachten, daß zwar die Koordinaten den Punkt bestimmen, nicht aber umgekehrt auch der Punkt seine Koordinaten.

Zur Veranschaulichung der projektiven Ebenen betrachten wir die (affine) Ebene $z = 1$ in k^3. Ein eindimensionaler Vektorraum $[(x, y, z)]$

schneidet diese Ebene genau dann, wenn z nicht verschwindet; der Schnittpunkt ist dann $(x/z, y/z, 1)$, und wir können den Punkt $P = (x, y, z)$ aus $\mathbb{P}^2(k)$ identifizieren mit diesem Schnittpunkt oder, noch einfacher, mit dem Punkt $(x/z, y/z)$ aus der affinen Ebenen k^2.

Für $z = 0$ ist der Untervektorraum $[(x, y, 0)]$ parallel zur Ebenen $z = 1$; hier gibt es also keinen Schnittpunkt. Genauso wie man sagt, daß sich zwei parallele Geraden im Unendlichen schneiden, kann sich aber auch vorstellen, daß sich eine Gerade mit einer dazu parallelen Ebenen im Unendlichen schneidet; wir können uns den Punkt mit homogenen Koordinaten $(x, y, 0)$ also vorstellen, als unendlich fernen Punkt in Richtung der durch $(x, y, 0)$ definierten Geraden. Man beachte, daß eine Gerade hier nur *eine* Richtung definiert, obwohl die reelle Anschauung nahelegt, daß es zwei geben sollte. Für einen beliebigen Körper allerdings, in dem man nicht zwischen positiven und negativen Zahlen unterscheiden kann, lassen sich diese beiden Richtungen nicht einmal definieren, und wie wir bald sehen werden, erhalten wir auch eine einfachere mathematische Theorie, wenn wir jeder solchen Geraden nur *einen* unendlich fernen Punkt zuordnen.

Dies entspricht auch der Tatsache, daß die eindimensionalen Untervektorräume von k^3 alle gleichberechtigt sind; erst durch (willkürliche) Auswahl der Ebene $z = 1$ bekommen wir eine Unterscheidung zwischen endlichen und unendlich fernen Punkten. Betrachten wir stattdessen etwa die Ebene $x = 1$ oder $y = 1$, werden Punkte der Form $(0, y, z)$ *bzw.* $(x, 0, z)$ unendlich fern. Unendlich ferne Punkte gibt es also erst, wenn wir zur besseren Veranschaulichung eine affine Ebene auszeichnen; in $\mathbb{P}^2(k)$ können wir den Begriff „unendlich fern“ nicht intrinsisch definieren.

Um in der projektiven Ebenen Geometrie treiben zu können, müssen wir in der Lage sein, dort Geraden und andere Kurven zu definieren. In der affinen Ebenen k^2 kann man eine algebraische Kurve definieren als Nullstellengebilde

$$V(f) = \left\{(x, y) \in k^2 \;\middle|\; f(x, y) = 0\right\}$$

eines Polynoms; im Projektiven haben wir die Schwierigkeit, daß die homogenen Koordinaten eines Punkts nicht eindeutig sind und daher der

Wert einer Funktion $f(x,y,z)$ nicht nur vom Punkt abhängt, sondern auch von den gerade betrachteten Koordinaten.

Wir können daher nicht einfach *irgendein* Polynom f betrachten, kommen aber zu einer sinnvollen Definition, wenn wir uns auf *homogene* Polynome beschränken:

Definition: Ein Polynom $F(X,Y,Z) = \sum_{i=1}^{m} a_i X^{c_i} Y^{d_i} Z^{e_i}$ heißt *homogen vom Grad d*, wenn $c_i + d_i + e_i = d$ ist für alle i, wenn also alle Terme von f denselben Grad d haben.

Für ein solches Polynom ist

$$\begin{aligned} f(\lambda x, \lambda y, \lambda z) &= \sum_{i=1}^{m} a_i (\lambda x)^{c_i} (\lambda y)^{d_i} (\lambda z)^{e_i} \\ &= \sum_{i=1}^{m} a_i \lambda^{c_i+d_i+e_i} x^{c_i} y^{d_i} z^{e_i} = \lambda^d \sum_{i=1}^{m} x^{c_i} y^{d_i} z^{e_i}, \end{aligned}$$

der *Wert* von F hängt also davon ab, welche homogenen Koordinaten eines Punkts wir benutzen, nicht aber das Verschwinden von F: Ein homogenes Polynom verschwindet entweder für alle Darstellungen eines Punkts $P \in \mathbb{P}^2(k)$ in homogenen Koordinaten oder für keine.

Damit können wir definieren

Definition: Eine Kurve vom Grad d in $\mathbb{P}^2(k)$ ist eine Teilmenge der Form

$$V(F) = \left\{ (x,y,z) \in \mathbb{P}^2(k) \;\middle|\; F(x,y,z) = 0 \right\},$$

wobei $F \in k[X,Y,Z]$ ein homogenes Polynom vom Grad d ist. Kurven vom Grad eins heißen Geraden.

Falls wir, zur besseren Veranschaulichung, diese Kurve affin betrachten wollen, können wir $z = 1$ setzen und erhalten ein (im allgemeinen nicht mehr homogenes) Polynom

$$f(X,Y) = F(X,Y,1)$$

in zwei Veränderlichen; die Einschränkung der Kurve auf die affine Ebene $z = 1$ ist dann

$$V(f) = \left\{ (x,y) \in k^2 \;\middle|\; f(x,y) = 0 \right\}.$$

Falls F nicht durch Z teilbar ist, hat auch f wieder den Grad d; andernfalls ist der Grad gleich $d - r$, wobei Z^r die höchste Z-Potenz ist, durch die F geteilt werden kann. Entsprechend lassen sich auch die Einschränkungen auf andere affine Ebenen definieren.

Ist umgekehrt eine affine Kurve

$$V(f) = \left\{ (x,y) \in k^2 \;\middle|\; f(x,y) = 0 \right\}$$

gegeben mit einem Polynom

$$f(X,Y) = \sum_{i=1}^{m} a_i X^{c_i} Y^{d_i}$$

vom Grad d in zwei Veränderlichen, so können wir dieses Polynom homogenisieren zu

$$F(X,Y,Z) = \sum_{i=1}^{m} a_i X^{c_i} Y^{d_i} Z^{d-c_i-d_i}$$

und die projektiv vervollständigte Kurve $V(F)$ betrachten. Sie unterscheidet sich von der affinen Kurve durch ihre unendlich fernen Punkte; das sind jene Punkte $(x,y,0)$ mit $F(x,y,0) = 0$.

Aus dem Einheitskreis $X^2+Y^2 = 1$ beispielsweise wird so die projektive Kurve $X^2 + Y^2 = Z^2$; über $k = \mathbb{R}$ gibt es hier keine unendlich ferne Punkte, da die Gleichung $x^2 + y^2 = 0$ dort nur die Lösung $x = y = 0$ hat, und $(0,0,0)$ definiert keinen Punkt von $\mathbb{P}^2(k)$, da der Nullvektor keinen eindimensionalen Untervektorraum von k^3 erzeugt.

Die Hyperbel $X^2 - Y^2 = 1$ wird projektiv zu $X^2 - Y^2 = Z^2$, und hier gibt es über jedem Körper unendlich ferne Punkte, nämlich $(1,1,0)$ und den in Charakteristik zwei damit identischen Punkt $(1,-1,0)$. Anschaulich sind diese beiden Punkte im Reellen die „Schnittpunkte“ der Hyperbel mit ihren beiden Asymptoten.

Die projektive Hyperbelgleichung $X^2 - Y^2 = Z^2$ kann auch als $Z^2 + Y^2 = X^2$ geschrieben werden; durch Vertauschung der X- und der Z-Koordinate wird sie dann zur oben betrachteten Gleichung des Einheitskreises. Im Projektiven gibt es also keinen Unterschied mehr zwischen einem Kreis und einer Hyperbel; ein Unterschied kommt nur dadurch zustande, daß man eine der Geraden in $\mathbb{P}^2(k)$ als unendlich fernn bezeichnet.

Das folgende Lemma zeigt, daß wir mit dem Übergang zur projektiven Ebene unser erstes Ziel zur Vereinfachung des Schnittverhaltens erreicht haben:

Lemma: Zwei verschiedene Geraden in $\mathbb{P}^2(k)$ schneiden sich stets in genau einem Punkt.

Beweis: Die beiden Geraden sind gegeben durch homogene PÜolynome vom Grad eins, also etwa durch

$$F(X,Y,Z) = a_1X+a_2Y+a_3Z \quad \text{und} \quad G(X;Y,Z) = b_1X+b_2Y+b_3Z\,;$$

ihre Schnittmenge ist die Lösungsmenge des homogenen linearen Gleichungssystems

$$F(x,y,z) = G(x,y,z) = 0\,.$$

Die beiden Geraden sind genau dann gleich, wenn die Polynome F und G Vielfache voneinander sind; andernfalls sind die beiden Gleichungen linear unabhängig, das Gleichungssystem hat also den Rang zwei. Seine Lösungsmenge ist somit ein eindimensionaler Untervektorraum von k^3, also ein Punkt aus $\mathbb{P}^2(k)$. ∎

Das zweite oben diskutierte Problem, die drei Möglichkeiten für die Anzahl der Schnittpunkte eines Kreises mit einer Geraden, ist dadurch allerdings noch nicht gelöst: Der projektive Kreis $X^2+Y^2 = Z^2$ schneidet die Gerade $X = 0$ in den beiden (in Charakteristik zwei identischen) Punkten $(0,1,1)$ und $(0,-1,1)$, die Gerade $X = Z$ schneidet er nur im Punkt $(1,0,1)$, und die Gerade $X = 2Z$ schneidet er im Reellen überhaupt nicht. Im Komplexen allerdings gibt es die beiden Schnittpunkte $(2,i,1)$ und $(2,-i,1)$, und entsprechende Lösungen gibt es offensichtlich auch über jedem algebraisch abgeschlossenen Körper. Das chaotische Schnittverhalten zwischen Kreisen und Geraden kann also erwas entschärft werden, indem man zum algebraischen Abschluß der

Grundkörpers übergeht. Darin sind wir in der Kryptologie zwar nicht sonderlich interessiert, denn alle algebraisch abgeschlossenen Körper sind unendlich, aber selbst über endlichen Körpern erweist es sich gelegentlich als nützlich, Zwischenschritte von Beweisen über dem algebraischen Abschluß durchzuführen und sich dann am Schluß gesondert zu überlegen, daß z.B. ein im Beweis konstruierter Punkt tatsächlich Koordinaten im betrachteten Grundkörper hat.

Das Problem der Tangenten allerdings wird durch den Übergang zum algebraischen Abschluß nicht gelöst; dazu brauchen wir eine andere Methode. Auch hier hilft wieder die Anschauung: Wenn wir eine Tangente als Grenzlage einer Sehne betrachten, sehen wir, daß beim Grenzübergang zwei Schnittpunkte zu einem Berührpunkt zusammenfallen; wir sollten diesen einen Punkt also doppelt zählen.

Der noch fehlende Begriff, der Schnitte wie die von Kreisen und Geraden übersichtlich macht, ist also eine Vielfachheit. Ihre Definition wird zurückgeführt auf die wohlbekannte Vielfachheit einer Nullstelle eines Polynoms:

Zur Definition der *Schnittmultiplizität* einer Geraden mit einer Kurve vom Grad d in einem Punkt P wählen wir zunächst eine affine Ebene, die den Punkt P enthält, für die er also nicht unendlich fern ist. So eine Ebene ist einfach zu finden, denn von den drei homogenen Koordinaten von P muß mindestens eine von null verschieden sein; die affine Ebene, in der diese gleich eins ist, leistet offensichtlich das Verlangte.

In dieser affinen Ebene ist die Kurve gegeben durch ein Polynom f in zwei Veränderlichen, die wir der Einfachheit halber mit X und Y bezeichnen wollen; die Gerade ist dort gegeben durch eine lineare Gleichung der Form $aX + bY = c$, in der a und b nicht beide verschwinden dürfen. Die Gleichung läßt sich daher nach mindestens einer der beiden Variablen auflösen; setzen wir diese Auflösung ein ins Polynom f, entsteht ein Polynom in nur noch einer Veränderlichen. Die Vielfachheit der P entsprechenden Nullstelle dieses Polynoms bezeichnen wir als Schnittmultiplizität.

Um etwa die Schnittmultiplizität des Kreises $X^2 + Y^2 = 2Z^2$ mit der Geraden $X + Y = 2Z$ im Punkte $(1,1,1)$ zu berechnen, gehen wir zunächst über zur affinen Ebene $z = 1$; dort wird die Kreisgleichung zu $X^2+Y^2 = 2$ und die Geradengleichung zu $X+Y = 2$. Auflösen nach Y ergibt $Y = 2 - X$, was in die Kreisgleichung eingesetzt auf

$$X^2 + (2 - X)^2 = 2X^2 - 4X + 4 = 2$$

oder

$$X^2 - 2X + 1 = (X - 1)^2 = 0$$

führt. Dieses Polynom hat bei $x = 1$ eine doppelte Nullstelle, also ist die Schnittmultiplizität des Kreises mit der Geraden im Punkt $(1, 1, 1)$ gleich zwei, wie es für eine Kreistangente auch sein soll.

Allgemein gilt

Satz: $C = V(F)$ sei eine Kurve vom Grad d in der projektiven Ebene $\mathbb{P}^2(k)$ über einem algebraisch abgeschlossenen Körper k. Dann hat C mit jeder Gerade, die nicht ganz auf C liegt, mit Vielfachheiten gezählt genau d Schnittpunkte.

Beweis: Klar, den löst man die Geradengleichung nach einer der Variablen auf und setzt dies ein in die Kurvengleichung, erhält man ein Polynom vom Grad d in einer Veränderlichen, und das hat, mit Vielfachheiten gezählt, genau d Nullstellen. ∎

Tatsächlich gilt noch erheblich mehr, nämlich der

Satz von Bézout: $C = V(F)$ und $D = V(G)$ seien Kurven vom Grad d *bzw.* e in der projektiven Ebene über dem algebraisch abgeschlossenen Körper k. Falls F und G teilerfremd sind, haben C und D mit Vielfachheit gezählt genau de Schnittpunkte.

Für die genaue Definition der Schnittmultiplizität bei beliebigen ebenen Kurven und den Beweis dieses Satzes sei auf die Vorlesung *Algebraische Kurven* verwiesen. Die Voraussetzung, daß F und G teilerfremd sein müssen, ist offensichtlich notwendig: Haben F und G einen gemeinsamen Teiler H, so ist $V(H)$ eine Kurve, die Teil sowohl von C als auch von D ist; die Schnittmenge von C und D ist also, zumindest wenn wir über einem algebraisch abgeschlossenen Körper arbeiten, unendlich.

b) Singularitäten

Wie wir im letzten Abschnitt gesehen haben, sind Ellipsen, Hyperbeln und Parabeln in der projektiven Ebenen nicht voneinander zu unterscheiden. Trotzdem definieren nicht alle homogenen Polynome zweiten Grades dieselbe Art von Kurven: Die Gleichung

$$F(X, Y; Z) = X^2 - Y^2 = 0$$

etwa läßt sich auch schreiben als

$$(X + Y)(X - Y) = 0\,,$$

für Punkte $(x, y, z) \in \mathbb{P}^2(k)$ mit $x^2 - y^2 = 0$ ist also $x = y$ oder $x = -y$. Die Kurve $V(F)$ besteht also aus den beiden Winkelhalbierenden der affinen (x, y)-Ebene, jeweils ergänzt durch ihren unendlichfernen Punkt $(1, 1, 0)$ *bzw.* $(1, -1, 0)$.

Der Nullpunkt $(0, 0, 1)$ der (x, y)-Ebene ist Schnittpunkt beider

c) Elliptische 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 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 oder komplexen Zahlen vorzustellen.

XXXXXXXXXXXXXXXXXXXXXXXXXXXXx

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.

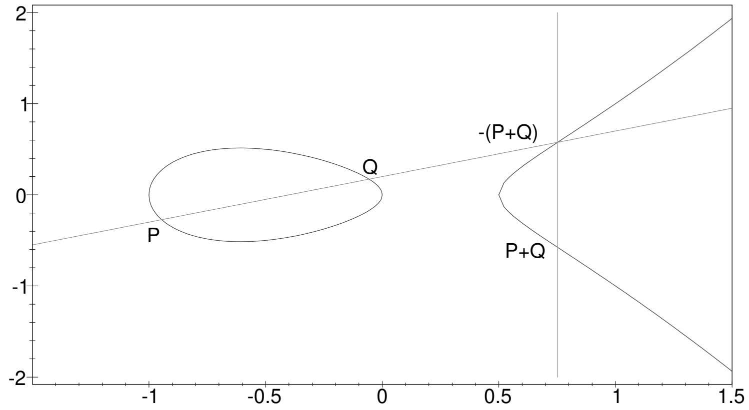
$$O + P = P + O \quad \text{für alle } P \in E\,,$$

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

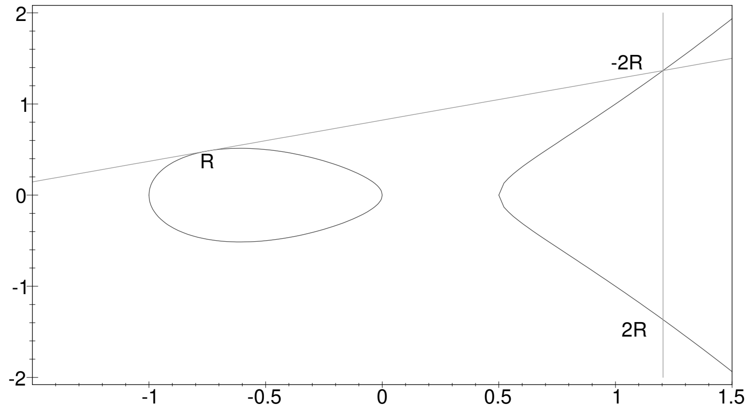
$$P + Q = P + Q + O = O\,,$$

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



Die Summe zweier verschiedener Punkte



Das Doppelte eines Punktes

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.

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-HELLMAN 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.

Inzwischen gibt es ein neues Buch von Menezes und anderen auf dem Stand von 2004

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: Nochmals elektronische Unterschriften

Elektronische Unterschriften, so wie wir sie bislang kennen, 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 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. Daher ist es im allgemeinen völlig problemlos, zu einem gegebenen Text beliebig viele weitere mit demselben Hashwert zu konstruieren. Mit der Unterschrift unter den Hashwert *irgendeines* Texts würde der Unterzeichner also jedem Interessenten die Möglichkeit geben, diese Unterschrift unter einen Text mit frei wählbarem Inhalt zu setzen – die Unterschrift wäre also wertlos.

Von einer kryptographisch brauchbaren Hashfunktion müssen wir daher fordern, daß es rechnerisch nicht mit vertretbarem Aufwand möglich sein darf, zu einem gegebenen 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 jeder Block der Nachricht zum Endergebnis beiträgt, erhalten wir automatisch eine deutlich Reduktion der Redundanz; wir brauchen daher nicht so viele oder nicht so aufwendige Runden pro Block wie bei der Verschlüsselung eines einzelnen Blocks.

Wie der nächste Paragraph zeigen wird, brauchen wir dafür aber 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:

$$e^{-\frac{n^2}{2N}} = p \Longleftrightarrow \frac{n^2}{2N} = -\ln p \Longleftrightarrow n = \sqrt{-2N\ln p}\,.$$

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 und ich mir nicht vorstellen kann, daß Hashwerte mit nur 160 Bit noch heutigen Sicherheitsanforderungen genügen.

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, fügte aber 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 $M^{(1)}, M^{(2)}, \ldots$ gehen nacheinander durch die folgenden Verarbeitungsschritte:

1. Setze

$$W_t = \begin{cases} M_t^{(i)} & \text{für } 0 \le t \le 15 \\ \mathrm{ROTL}(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) & \text{für } 16 \le t \le 79 \end{cases} .$$

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:

$$T \to \overset{5}{\mathrm{ROTL}}(a) + f_t(b, c, d) + K_t + W_t, \quad e \to d, \quad d \to c,$$
$$c \to \overset{30}{\mathrm{ROTL}}(b), \quad b \to a, \quad a \to T\ ,$$

wobei $f_t = \begin{cases} Ch & \text{für } \le t \le 19 \\ Maj & \text{für } 40 \le t \le 59 \\ Parity & \text{sonst} \end{cases}$ 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 $0 \leq m < q$, 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.

Andere nichtklassische Anwendungen der Kryptologie sind etwa das Werfen von Münzen (für eine zufällige Entscheidung „Kopf oder Zahl") via Telephon oder auch ein Pokerspiel per Internet. Die dafür eingesetzten Protokolle können durchaus auch ernste Anwendungen haben, beispielsweise beim verteilten Rechnen, wenn die Zuweisung von Aufgaben an die einzelnen Rechner nach einem Zufallsverfahren erfolgt. Falls die Kosten für die Inanspruchnahme der verschiedenen Rechner von verschiedenen Personen getragen werden, sollten diese definitiv daran interessiert sein, daß niemand dem Zufall auf ihre Kosten nachhilft.

§1: Werfen einer Münze per Telephon

Beim Werfen einer Münze geht es darum, daß zwei Partner A und B eine Entscheidung herbeiführen, die für beide als zufällig erkennbar ist. Bei einer Telephonverbindung ohne Videokanal sind also Münzen nutzlos – es sei denn, eine perfekte Synchronität der Übertragungen in beiden Richtungen wäre möglich. In diesem Fall könnte etwa jeder der beiden eine Münze werfen; falls beide Münzen das gleiche Ergebnis zeigen, gewinnt A, sonst B. Technisch ist diese perfekte Synchronität allerdings nur mit erheblichem Zusatzaufwand zu realisieren; fuer eine gewöhnliche Internetverbindung ist das unrealistisch.

Praktikabler ist das folgende Verfahren: A wählt zwei große Primzahlen p und q und schickt deren Produkt $N = pq$ an B. Dieser wählt eine Zufallszahl x zwischen $\sqrt{N}+1$ und $N-1-\sqrt{N}$ und schickt $y = x^2$ am A. Da A die Primzerlegung von N kennt, kann er Quadratwurzeln modulo N berechnen: Wie wir gleich sehen werden, gibt es Algorithmen, um Quadratwurzeln modulo einer Primzahl effizient zu berechnen, und sind w_1, w_2 Quadratwurzeln von y modulo p *bzw.* modulo q, so läßt sich daraus nach dem chinesischen Restesatz eine Quadratwurzel w modulo $N = pq$ berechnen.

Mit w_1 und w_2 sind allerdings auch $-w_1$ und $-w_2$ Quadratwurzeln modulo p *bzw.* q; je nachdem, für welche Werte sich A entscheidet, kommt er also auf einen von *vier* möglichen Werte für w. Darunter sind insbesondere die beiden Werte $\pm x$, dazu kommen zwei weitere Werte $\pm x'$. Einen dieser Werte schickt A an B. In 50% aller Fälle wird dies x oder $-x$ sein; damit kann B nichts anfangen, und er hat verloren.

In den restlichen 50% der Fälle erhält B einen von $\pm x$ verschiedenen Wert w. Dieser muß dann modulo einer der beiden Primzahlen kongruent x und modulo der anderen kongruent $-x$ sein, d.h. $x+w$ und $x-w$ sind jeweils durch *genau* eine der beiden Primzahlen p und q teilbar. Indem B also beispielsweise den ggT von $x+w$ und N berechnet, findet er die Faktorisierung von N und schickt diese an A zum Beweis, daß er gewonnen hat.

Falls p und q hinreichend groß sind, hat B keine realistische Möglichkeit, N auf andere Weise zu faktorisieren, insbesondere nicht in den wenigen Sekunden, mit denen man bei korrekter Durchführung des Protokolls auskommt. B hat somit keine Möglichkeit, den Ausgang zu seinen Gunsten zu beeinflussen. Er könnte zwar versuchen, zwei Zahlen $x \neq \pm x'$ mit gleichem Quadrat zu finden und eine davon an A schicken, aber wie wir bei der Diskussion des quadratischen Siebs gesehen haben, ist genau das die derzeit effizienteste Methode zur Faktorisierung von N und damit nicht praktikabel.

Auch A hat keine Möglichkeit, das Ergebnis zu seinen Gunsten zu beeinflussen, denn er kann zwar alle vier Quadratwurzeln von y modulo N berechnen, weiß aber nicht, welche davon die Zahl x ist, deren Quadrat ihm B übermittelte.

§2: Poker per Telephon

Poker ist ein Kartenspiel, das traditionellerweise in verrauchten Hinterzimmern von Restaurants gespielt wurde, wobei meist auch viel Alkohol im Spiel war. Der moderne Internetuser allerdings möchte sich nicht einen ganzen Pokerabend lang von seinem Computer trennen und muß daher einen anderen Weg finden. Das kryptographische Problem besteht darin, daß bei einem traditionellen Pokerspiel die Karten jeweils gemischt, abgehoben und verteilt werden müssen und am Ende niemand die Karten seiner Mitspieler kennen darf.

Der Ansatz ist ähnlich wie bei den blinden Unterschriften, die für elektronisches Bargeld benutzt werden, allerdings wird statt RSA ein einfacheres Verfahren verwendet, das Verfahren von POHLIG und HELLMANN. Es funktioniert so ähnlich wie RSA, allerdings wird anstelle des Produkts zweier Primzahlen nur eine Primzahl p als Modulus verwendet. Zur Verschlüsselung dient ein zu $p-1$ teilerfremder Exponent e, mit dem eine Nachricht m verschlüsselt wird als $m^e \bmod p$. Zur Entschlüsselung dient entsprechend ein zweiter Exponent d mit der Eigenschaft, daß $de \equiv 1 \bmod p-1$ ist, denn nach dem kleinen Satz von FERMAT ist dann $m^{de} \equiv m \bmod p$. d kann wie bei RSA nach dem erweiterten EUKLIDischen Algorithmus berechnet werden, angewandt auf e und $p-1$.

Die Berechnung von d kann jeder ausführen, der die zur Anwendung des Algorithmus notwendigen Zahlen p und e kennt; der Algorithmus von POHLIG und HELLMANN ist also kein asymmetrisches Verfahren, sondern ein symmetrisches Kryptoverfahren, dessen Schlüssel geheim bleiben muß. Man kann wahlweise das Paar (p, e) als Schlüssel betrachten oder aber die Primzahl p innerhalb eines Netzwerks ein für alle man fest wählen und öffentlich bekanntgeben und dann nur den Exponenten e als geheimen Schlüssel betrachten.

Auf den ersten Blick scheint das Verfahren von POHLIG und HELLMANN vor allem die Nachteile der symmetrischen und der asymmetrischen Kryptographie zu vereinen: Wie bei allen symmetrischen Verfahren hat man das Problem des Schlüsselaustauschs, und das bei einem Rechenaufwand, der dem des RSA-Verfahrens entspricht!

Tatsächlich muß man die Sicherheit des Verfahrens von POHLIG und HELLMANN nach völlig anderen Kriterien beurteilen als die des RSA-

Verfahrens: Die empfohlene Schlüssellänge von 2048 Bit bei RSA erklärt sich aus dem Stand und der erwarteten Entwicklung bei Faktorisierungsalgorithmen; diese aber spielen für die Sicherheit von POHLIG/HELLMANN aber keinerlei Rolle. Hier muß ein Angreifer versuchen, den bei RSA öffentlich bekannten Exponenten e zu ermitteln; falls er die möglichen Exponenten einfach durchprobiert, ist er ungefähr in derselben Situation wie bei einem Angriff auf DES oder AES, so daß man vielleicht argumentieren könnte, daß ungefähr dieselben Sicherheitsparameter wie für Algorithmen dieser Art gewählt werden sollten, d.h. nach heutigem Stand mindestens 128 Bit für Primzahl und Exponent.

Dies ist aber zu optimistisch: Wie wir schon bei der Diskussion der Sicherheit von DES gesehen haben, müssen sich bei einer guten Blockchiffre die Transformationen wie eine Zufallsauswahl aus der vollen Permutationsgruppe über der Menge aller möglicher Blöcke verhalten; insbesondere dürfen sie keine zu kleine Untergruppe dieser symmetrischen Gruppe erzeugen.

Diese Bedingung ist hier klar verletzt: Die Transformationen von POHLIG und HELLMANN bilden sogar bereits eine (zyklische) Untergruppe der vollen Permutationsgruppe. Dies gibt einem Angreifer eine ganze Reihe zusätzlicher Möglichkeiten; insbesondere muß er bei einer Attacke mit bekanntem Klartext nur ein diskretes Logarithmenproblem lösen, wozu es, wie wir aus §4 von Kapitel sechs wissen, deutlich schnellere Algorithmen gibt als das vollständige Durchsuchen des Schlüsselraums.

Wenn wir trotzdem oft mit deutlich kürzeren Schlüssellängen arbeiten als bei Verfahren mit diskreten Logarithmen, rechtfertigt sich das vor allem aus der Art der Anwendungen: Das Verfahren von POHLIG und HELLMANN wird in erster Linie eingesetzt für Protokolle, die in Echtzeit ablaufen; falls man dazu dann auch noch *ad hoc*-Schlüssel einsetzt, nützt eine Kryptanalyse dem Gegner nur dann, wenn er sie innerhalb weniger Sekunden oder höchstens Minuten durchführen kann. In solchen Situationen sind die Sicherheitsanforderungen natuerlich erheblich geringer als etwa bei elektronischen Unterschriften, die oft jahrelang sicher sein müssen.

§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

$$\left(\frac{q}{p}\right)\left(\frac{p}{q}\right) = (-1)^{\left(\frac{p-1}{2}\right)\left(\frac{q-1}{2}\right)} = \begin{cases} +1 & \text{falls } p \equiv q \bmod 4 \\ -1 & \text{sonst} \end{cases}.$$

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

$$\prod_{i=1}^{\ell} a_i \prod_{j=1}^{m} b_j \equiv \prod_{k=1}^{(p-1)/2} (ka) = \left(\frac{p-1}{2}\right)! \cdot a^{(p-1)/2}.$$

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

$$\left(\frac{p-1}{2}\right)! = \prod_{i=1}^{\ell} a_i \prod_{j=1}^{m}(p-b_j) \equiv (-1)^m \prod_{i=1}^{\ell} a_i \prod_{j=1}^{m} b_j$$
$$\equiv (-1)^m \left(\frac{p-1}{2}\right)!\ a^{(p-1)/2} \bmod p$$

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

$$n = \frac{p-1}{\mathrm{ggT}(r, p-1)}.$$

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

$$k = -\beta q \bmod 2^e \in \{0, 1, 2, \dots, 2^e - 1\} ,$$

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 berechnen 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 $\times$-System übertragen wurde, d.h. mit Polarisationsrichtung 45°. Entsprechend kann 135° im $\times$-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 0°. 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 dieser Vorlesung bieten kann. Verfügt man über solche 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

$$\left\langle \begin{pmatrix}1\\ i\end{pmatrix} \middle| \begin{pmatrix}1\\ i\end{pmatrix}\right\rangle = 1^2 + i\bar{i} = 2$$

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

$$\langle\cdot|\cdot\rangle : V\times V\to\mathbb{C};\quad \big(|v\rangle\,,|w\rangle\big)\mapsto\langle v|w\rangle$$

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 $A: V \to V$ 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

$$(x,y) \mapsto \begin{cases} (x, \neg y) & \text{falls } x = 1 \\ (x, \quad y) & \text{falls } x = 0 \end{cases} = (x, x \oplus y)\,,$$

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:

$$T(x,y,|0\rangle) = (x, y, x \wedge y) \quad \text{und} \quad T(\neg x, \neg y, |1\rangle) = (\neg x, \neg y, x \vee y)\,.$$

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 $\mathrm{ggT}(x - y, N)$ und $\mathrm{ggT}(x + y, N)$ 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 $0 \leq a < 2^{2L}$ 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 $0 \leq a < 2^{2L}$, 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.

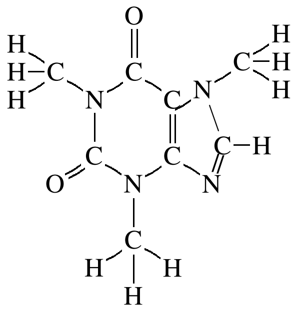
Die besten rationalen Approximationen einer reellen Zahl mit Nennern einer vorgegebenen Größenordnung liefert, wie wir in Kap. 6, §6f, gesehen haben, deren Kettenbruchentwicklung; also berechnen wir für jeden Meßwert μ alle Konvergenten des Kettenbruchs zu $\frac{\mu}{2^{2L}}$ mit Nenner kleiner N und 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 konzi-

pierte *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.

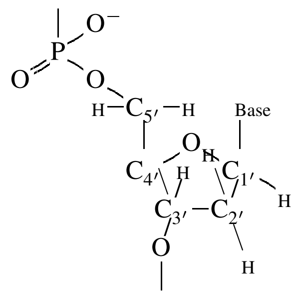
Wenn man dieses Potential für klassische Probleme der wissenschaftlichen oder gewerblichen Informationsverarbeitung nutzbar machen könnte, wäre dies eine Revolution, die auch die Kryptographie massiv verändern würde.

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.

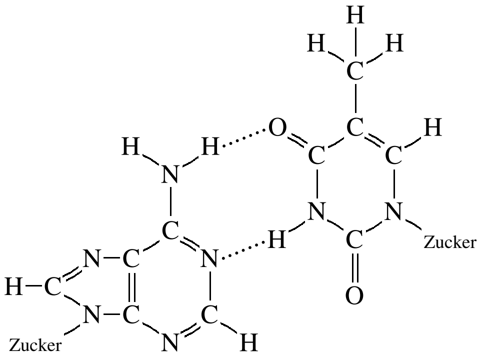
Um seine Vorgehensweise und das Potential der Methode zu verstehen, müssen wir zunächst kurz den Aufbau der DNS betrachten.

a) Die Desoxyribonucleinsäure

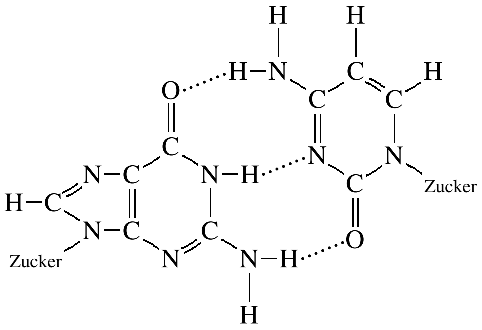
Die Desoxyribonucleinsäure hat ihren Namen vom hier abgebildeten Zuckermolekül Desoxyribose, aus dem ihr Gerüst aufgebaut ist.



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° – 95° 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.

Elf 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