Kapitel 2
Anwendungen in der Kryptographie

§ 1: New directions in cryptography

In der klassischen Kryptographie verläuft die Entschlüsselung entweder genauso oder zumindest sehr ähnlich wie die Verschlüsselung; insbesondere kann jeder, der eine Nachricht verschlüsseln kann, jede andere entsprechend verschlüsselte Nachricht auch entschlüsseln. Man bezeichnet diese Verfahren daher als *symmetrisch*.

Der Nachteil eines solchen Verfahrens besteht darin, daß in einem Netzwerk jeder Teilnehmer mit jedem anderen einen Schlüssel vereinbaren muß. In militärischen Netzen war dies üblicherweise so geregelt, daß das gesamte Netz denselben Schlüssel benutzte, der in einem Codebuch für jeden Tag im voraus festgelegt war; in kommerziellen Netzen wie beispielsweise einem Mobilfunknetz ist so etwas natürlich unmöglich.

1976 publizierten MARTIN HELLMAN, damals Assistenzprofessor in Stanford, und sein Forschungsassistent WHITFIELD DIFFIE eine Arbeit mit dem Titel *New directions in cryptography* (IEEE Trans. Inform. Theory **22**, 644–654), in der sie vorschlugen, den Vorgang der *Ver*schlüsselung und den der *Ent*schlüsselung völlig voneinander zu trennen: Es sei schließlich nicht notwendig, daß der Sender einer verschlüsselten Nachricht auch in der Lage sei, diese zu *ent*schlüsseln.

Der Vorteil eines solchen *asymmetrischen* Verfahrens wäre, daß jeder potentielle Empfänger nur einen einzigen Schlüssel bräuchte und dennoch sicher sein könnte, daß nur er selbst seine Post entschlüsseln kann. Der Schlüssel müßte nicht einmal geheimgehalten werden, da es ja

(meistens) nichts schadet, wenn jedermann Nachrichten *ver*schlüsseln kann. In einem Netzwerk mit n Teilnehmern bräuchte man also nur n Schlüssel, um es jedem Teilnehmer zu erlauben, mit jedem anderen sicher zu kommunizieren. Die Schlüssel könnten sogar in einem öffentlichen Verzeichnis stehen. Bei einem symmetrischen Kryptosystem wäre der gleiche Zweck nur erreichbar mit $\frac{1}{2}n(n-1)$ Schlüsseln, die auf einem sicheren Weg wie etwa bei einem persönliches Treffen oder durch vertrauenswürdige Boten ausgetauscht werden müßten.

BAILEY WHITFIELD DIFFIE wurde 1944 geboren. Erst im Alter von zehn Jahren lernte er lesen; im gleichen Jahr hielt eine Lehrerin an seiner New Yorker Grundschule einen Vortrag über Chiffren. Er ließ sich von seinem Vater alle verfügbare Literatur darüber besorgen, entschied sich dann 1961 aber doch für ein Mathematikstudium am MIT. Um einer Einberufung zu entgehen, arbeitete er nach seinem Bachelor bei Mitre; später, nachdem sein Interesse an der Kryptographie wieder erwacht war, kam er zu Martin Hellman nach Stanford, der ihn als Forschungsassistent einstellte. Seit 1991 arbeitet er als *chief security officer* bei Sun Microsystems. Seine dortige home page hat den URL `http://research.sun.com/people/diffie/` .

MARTIN HELLMAN wurde 1945 in New York geboren. Er studierte Elektrotechnik zunächst bis zum Bachelor an der dortigen Universität; für Master und Promotion studierte er in Stanford. Nach kurzen Zwischenaufenthalten am Watson Research Center der IBM und am MIT wurde er 1971 Professor an der Stanford University. Seit 1996 ist er emeritiert, gibt aber immer noch Kurse, mit denen er Schüler für mathematische Probleme interessieren will. Seine home page findet man unter `http://www-ee.stanford.edu/~hellman/` .

DIFFIE und HELLMAN machten nur sehr vage Andeutungen, wie so ein System mit öffentlichen Schlüsseln aussehen könnte. Es ist zunächst einmal klar, daß ein solches System keinerlei Sicherheit gegen einen Gegner mit unbeschränkter Rechenkraft (In der Kryptographie spricht man von einem BAYESschen Gegner) bieten kann, denn die Verschlüsselungsfunktion ist eine bijektive Abbildung zwischen endlichen Mengen, und jeder, der die Funktion kennt, kann zumindest im Prinzip auch ihre Umkehrfunktion berechnen.

Wer im Gegensatz zum BAYESschen Gegner nur über begrenzte Ressourcen verfügt, kann diese Berechnung allerdings möglicherweise nicht mit realistischem Aufwand durchführen, und nur darauf beruht die Sicherheit eines Kryptosystems mit öffentlichen Schlüsseln. Da Computer immer leistungsfähiger werden und auch immer neue und bessere Algorithmen gefunden werden, sind solche Systeme insbesondere nicht für die Ewigkeit gedacht, sondern nur für die einigermaßen überschaubare Zukunft. Derzeit geht man in der Kryptologie davon aus, daß kein Gegner 2^{128} oder gar mehr Entschlüsselungsversuche durchführen kann und bezeichnet ein System daher als sicher, wenn trotz intensiver Untersuchung kein Angriff bekannt ist, der mit geringerem Aufwand und einer nicht vernachlässigbaren Erfolgsaussicht Entschlüsselungen liefert. Man spricht dann von 128-Bit-Sicherheit, wobei diese Sicherheit natürlich nicht bewiesen ist.

Einige Stellen, darunter auch das Bundesamt für Sicherheit in der Informationstechnik, halten derzeit noch eine 100-Bit-Sicherheit für ausreichend. Dieser Auffassung schließen sich unsere Banken an: Bei den derzeit in Deutschland üblichen Bankkarten für Geldautomaten ist die Geheimzahl mit dem sogenannten Triple-DES geschützt, dessen Sicherheit bei knapp 112 Bit liegt.

DIFFIE und HELLMAN bezeichnen eine Funktion, deren Umkehrfunktion nicht mit vertretbarem Aufwand (im obigen Sinne) berechnet werden kann, als *Einwegfunktion* und wollen solche Funktionen zur Verschlüsselung verwenden. Das allein führt allerdings noch nicht zu einem praktikablen Kryptosystem, denn bei einer echten Einwegfunktion ist es auch für den legitimen Empfänger nicht möglich, seinen Posteingang zu entschlüsseln. DIFFIE und HELLMAN schlagen deshalb eine Einwegfunktion mit *Falltür* vor, wobei der legitime Empfänger zusätzlich zu seinem öffentlichen Schlüssel noch über einen geheimen Schlüssel verfügt, mit dem er (und nur er) diese Falltür öffnen kann.

Natürlich hängt alles davon ab, ob es solche Einwegfunktionen mit Falltür wirklich gibt. DIFFIE und HELLMAN gaben keine an, und es gab

unter den Experten einige Skepsis bezüglich der Möglichkeit, solche Funktionen zu finden.

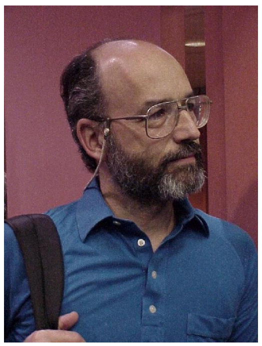
Tatsächlich aber gab es damals bereits Systeme, die auf solchen Funktionen beruhten, auch wenn sie nicht in der offenen Literatur dokumentiert waren: Die britische *Communications-Electronics Security Group* (CESG) hatte bereits Ende der sechziger Jahre damit begonnen, nach entsprechenden Verfahren zu suchen, um die Probleme des Militärs mit dem Schlüsselmanagement zu lösen, aufbauend auf (impraktikablen) Ansätzen von AT&T zur Sprachverschlüsselung während des zweiten Weltkriegs. Die CESG sprach nicht von Kryptographie mit öffentlichen Schlüsseln, sondern von *nichtgeheimer Verschlüsselung*, aber das Prinzip war das gleiche.

Erste Ideen dazu sind in einer auf Januar 1970 datierten Arbeit von JAMES H. ELLIS zu finden, ein praktikables System in einer auf den 20. November 1973 datierten Arbeit von CLIFF C. COCKS. Wie im Milieu üblich, gelangte nichts über diese Arbeiten an die Öffentlichkeit; erst 1997 veröffentlichten die *Government Communications Headquarters* (GCHQ), zu denen CESG gehört, einige Arbeiten aus der damaligen Zeit; eine Zeitlang waren sie auch auf dem Server `http://www.cesg.gov.uk/` zu finden, wo sie allerdings inzwischen anscheinend wieder verschwunden sind.

Im akademischen Bereich gab es ein Jahr nach Erscheinen der Arbeit von DIFFIE und HELLMAN das erste Kryptosystem mit öffentlichen Schlüsseln: Drei Wissenschaftler am Massachusetts Institute of Technology fanden nach rund vierzig erfolglosen Ansätzen 1977 schließlich jenes System, das heute nach ihren Anfangsbuchstaben mit RSA bezeichnet wird: RON RIVEST, ADI SHAMIR und LEN ADLEMAN.

RIVEST, SHAMIR und ADLEMAN gründeten eine Firma namens RSA Computer Security Inc., die 1983 das RSA-Verfahren patentieren ließ und auch nach Auslaufen dieses Patents im September 2000 noch erfolgreich im Kryptobereich tätig ist. 2002 erhielten RIVEST, SHAMIR und ADLEMAN für die Entdeckung des RSA-Systems den TURING-Preis der *Association for Computing Machinery* ACM, ein jährlich vergebener Preis, der als eine der höchsten Auszeichnungen der Informatik gilt.

RONALD LINN RIVEST wurde 1947 in Schenectady im US-Bundesstaat New York geboren. Er studierte zunächst Mathematik an der Yale University, wo er 1969 seinen Bachelor bekam; danach studierte er in Stanford Informatik. Nach seiner Promotion 1974 wurde er Assistenzprofessor am Massachusetts Institute of Technology, wo er heute einen Lehrstuhl hat. Er arbeitet immer noch auf dem Gebiet der Kryptographie und entwickelte eine ganze Reihe weiterer Verfahren, auch symmetrische Verschlüsselungsalgorithmen und Hashverfahren. Er ist Koautor eines Lehrbuchs über Algorithmen. Seine home page ist //http://theory.lcs.mit.edu/~rivest/ .



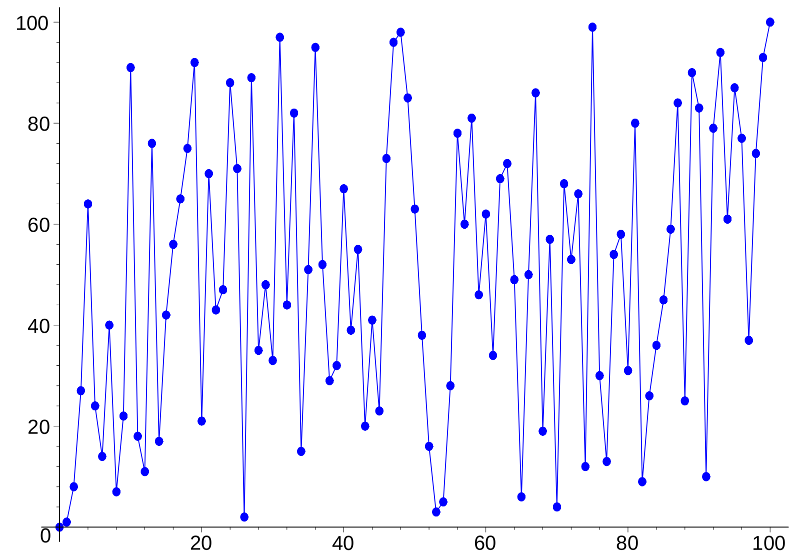
ADI SHAMIR wurde 1952 in Tel Aviv geboren. Er studierte zunächst Mathematik an der dortigen Universität; nach seinem Bachelor wechselte er ans Weizmann Institut, wo er 1975 seinen Master und 1977 die Promotion in Informatik erhielt. Nach einem Jahr als Postdoc an der Universität Warwick und drei Jahren am MIT kehrte er ans Weizmann Institut zurück, wo er bis heute Professor ist. Außer für RSA ist er bekannt sowohl für die Entwicklung weiterer Kryptoverfahren als auch für erfolgreiche Angriffe gegen Kryptoverfahren. Er schlug auch einen optischen Spezialrechner zur Faktorisierung großer Zahlen vor. Seine home page ist erreichbar unter http://www.wisdom.weizmann.ac.il/math/profile/scientists/shamir-profile.html

LEONARD ADLEMAN wurde 1945 in San Francisco geboren. Er studierte in Berkeley, wo er 1968 einen BS in Mathematik und 1976 einen PhD in Informatik erhielt. Thema seiner Dissertation waren zahlentheoretische Algorithmen und ihre Komplexität. Von 1976 bis 1980 war er an der mathematischen Fakultät des MIT; seit 1980 arbeitet er an der University of Southern California in Los Angelos. Seine Arbeiten beschäftigen sich mit Zahlentheorie, Kryptographie und Molekularbiologie. Er führte nicht nur 1994 die erste Berechnung mit einem „DNS-Computer" durch, sondern arbeitete auch auf dem Gebiet der Aidsforschung. Heute hat er einen Lehrstuhl für Informatik und Molekularbiologie. http://www.usc.edu/dept/molecular-science/fm-adleman.htm

RSA ist übrigens identisch mit dem von COCKS vorgeschlagenen System, so daß einige Historiker auch Zweifel an den Behauptungen der GCHQ haben. Die Beschreibung durch RIVEST, SHAMIR und ADLEMAN erschien 1978 unter dem Titel *A method for obtaining digital signatures and public-key cryptosystems* in Comm. ACM **21**, 120–126.

§2: Das RSA-Verfahren

Für eine natürliche Zahl e ist die reelle Funktion $x \mapsto x^e$ für positive x monoton ansteigend und bijektiv; ihre Umkehrfunktion $x \mapsto \sqrt[e]{x}$ läßt sich mit etwa demselben Aufwand berechnen wie die Funktion selbst. Betrachten wir $x \mapsto x^e$ allerdings als Funktion von $\mathbb{Z}/N \to \mathbb{Z}/N$, so erhalten wir einen sehr chaotisch aussehenden Graphen und können uns daher Hoffnungen machen, daß diese Funktion vielleicht als Grundlage einer kryptographischen Verschlüsselung brauchbar sein könnte.



Die Funktion $x \mapsto x^3$ in $\mathbb{F}_{101}$

Dazu muß sie natürlich zunächst einmal injektiv sein. Da die Ordnung eines Elements von $(\mathbb{Z}/N\mathbb{Z})^\times$ Teiler von $\varphi(N)$ ist, muß insbesondere

e teilerfremd zu $\varphi(N)$ sein. Dann lassen sich mit dem erweiterten EUKLIDischen Algorithmus Zahlen $d, k \in \mathbb{N}$ finden, so daß $de - k\varphi(N) = 1$ ist, d.h. für jedes zu N teilerfremde x ist

$$(x^e)^d = x^{ed} = x^{1+k\varphi(N)} \equiv x \bmod N\,.$$

Somit sind die Funktionen

$$\begin{cases} (\mathbb{Z}/N\mathbb{Z})^\times \to (\mathbb{Z}/N\mathbb{Z})^\times \\ x \mapsto x^e \end{cases} \quad \text{und} \quad \begin{cases} (\mathbb{Z}/N\mathbb{Z})^\times \to (\mathbb{Z}/N\mathbb{Z})^\times \\ x \mapsto x^d \end{cases}$$

zueinander invers.

Die Beschränkung auf prime Restklassen ist für kryptographische Anwendungen ungünstig: Am einfachsten wäre es, wenn wir jede Bitfolge, deren Länge kleiner ist als die der Binärdarstellung von N, als Zahl zwischen 0 und $N-1$ auffassen, verschlüsseln und übertragen könnten. Der Empfänger könnte dann die Zahl entschlüsseln, als Bitfolge hinschreiben und daraus die Nachricht rekonstruieren. Zum Glück ist das zumindest für *quadratfreie* Zahlen N, d.h. Zahlen, die durch kein Primzahlquadrat teilbar sind, auch möglich:

Satz: Für eine quadratfreie natürliche Zahl N sind die beiden Funktionen

$$\begin{cases} \mathbb{Z}/N\mathbb{Z} \to \mathbb{Z}/N\mathbb{Z} \\ x \mapsto x^e \end{cases} \quad \text{und} \quad \begin{cases} \mathbb{Z}/N\mathbb{Z} \to \mathbb{Z}/N\mathbb{Z} \\ x \mapsto x^d \end{cases}$$

bijektiv und invers zueinander.

Beweis: Als quadratfreie Zahl ist N ein Produkt verschiedener Primzahlen p_i, und $\varphi(N)$ ist das Produkt der zugehörigen $\varphi(p_i) = p_i - 1$. Somit ist auch $ed \equiv 1 \bmod \varphi(p_i)$ für alle i, und nach dem chinesischen Restesatz genügt es, wenn wir den Satz für die einzelnen p_i beweisen. Ist $N = p$ prim, so ist die Null das einzige Element von $\mathbb{Z}/p$, das nicht in $(\mathbb{Z}/p)^*$ liegt, und sie wird von beiden Funktionen auf sich selbst abgebildet. ■

Jeder, der e und N kennt, kann auch d berechnen, allerdings muß er dazu als erstes $\varphi(N)$ bestimmen. Nach der Formel aus Kapitel 1, §6

ist das möglich, wenn er die Primfaktorzerlegung von N kennt. Einfachere alternative Verfahren sind nicht bekannt, und wie wir im Kapitel über Faktorisierungsalgorithmen sehen werden, ist diese Zerlegung mit heutigen Mitteln für ein Produkt zweier gut gewählter Primzahlen nicht mehr mit realistischem Aufwand möglich, wenn dieses Produkt mehr als etwa 200 Dezimalstellen hat. Natürlich wird diese Schranke im Laufe der Jahrzehnte ansteigen, und wahrscheinlich können die Geheimdienste schon heute etwas mehr als der Rest der Welt. Es ist aber unwahrscheinlich, daß bei einem schon seit Jahrhunderten untersuchten Problem wie der Faktorisierung ganzer Zahlen ausgerechnet einem Geheimdienst ein Durchbruch gelingen sollte, von dem der Rest der Welt nichts bemerkt. Die Effekte einer leistungsfähigeren Hardware lassen sich durch großzügige Sicherheitszuschläge kompensieren.

Für eine Primzahl $N = p$ kann natürlich jeder ganz einfach $\varphi(p) = p - 1$ berechnen; ist $N = pq$ dagegen das Produkt zweier Primzahlen, so ist die Bestimmung von

$$\varphi(N) = (p-1)(q-1) = N - (p+q) + 1$$

äquivalent zur Kenntnis der Faktorisierung: Kennt man nämlich das Produkt $N = pq$ sowie die Summe $N + 1 - \varphi(N) = p + q$ der beiden Primzahlen, so kann man sie einfach berechnen als Lösungen der quadratischen Gleichung

$$(x-p)(x-q) = x^2 - \big(N + 1 - \varphi(N)\big)x + N = 0\,.$$

Auch bei Produkten von mehr als drei Primzahlen ist keine Methode zur Berechnung der EULERschen φ-Funktion bekannt, die effizienter wäre als der Umweg über die Faktorisierung, allerdings wird die Faktorisierung bei konstanter Größenordnung von N tendenziell einfacher, wenn die Anzahl der Faktoren steigt, da wir es dann zumindest teilweise mit kleineren Faktoren zu tun haben.

Zur praktischen Durchführung des RSA-Verfahrens wählt sich daher jeder Teilnehmer zwei verschiedene Primzahlen p, q, die unbedingt geheim gehalten werden müssen, und eine natürliche Zahl e, die keinen gemeinsamen Teiler mit $(p-1)(q-1)$ hat. Die Zahlen $N = pq$ und e sind sein öffentlicher Schlüssel, der beispielsweise in einem Verzeichnis publiziert werden kann.

Des weiteren berechnet er zu $\varphi(N) = (p-1)(q-1)$ nach dem EUKLIDischen Algorithmus eine natürliche Zahl d, so daß $de + k\varphi(N) = 1$ ist für ein gewisses $k \in \mathbb{Z}$. Diese Zahl d ist sein geheimer Schlüssel; da

$$\left(a^e\right)^d \equiv a \bmod N$$

für alle a, läßt sich damit die Entschlüsselung rückgängig machen.

Jeder, der den öffentlichen Schlüssel (N, e) kennt, kann damit Nachrichten verschlüsseln: Er bricht die Nachricht auf in Blöcke, die durch ganze Zahlen zwischen 0 und $N-1$ dargestellt werden können, berechnet für jeden so dargestellten Block a den Chiffretext $b = a^e \bmod N$ und schickt diesen an den Inhaber des geheimen Schlüssels. Dieser berechnet $b^d \bmod N = a^{ed} \bmod N = a$, und da er dazu seinen geheimen Schlüssel braucht, kann dies niemand außer ihm auf diese Weise berechnen.

Es wäre allerdings verfrüht, daraus schon auf die Sicherheit des RSA-Verfahrens zu schließen: Der Gegner, vor dem eine Nachricht geschützt werden soll, ist schließlich frei in der Wahl seiner Mittel und kann sich auch andere Formen der Attacke überlegen. Soweit entsprechende Strategien bekannt sind, muß man sich auch dagegen schützen. In der bislang dargestellten sogenannten „Lehrbuchversion“ bietet RSA, wie wir im Verlauf der Vorlesung noch mehrfach sehen werden, eine ganze Reihe von Angriffsmöglichkeiten, die gelegentlich erheblich schneller ans Ziel führen als die Faktorisierung des Moduls N.

Beispielsweise wird in der Praxis oft der öffentliche Exponent $e = 3$ verwendet (was natürlich voraussetzt, daß die verwendeten Primzahlen kongruent zwei modulo drei sind), so daß zumindest die Verschlüsselung recht schnell geht. Außerdem läßt sich dann der private Exponent d sehr einfach bestimmen: Nach der allgemeinen Theorie gibt es Zahlen d, k, so daß $de - k\varphi(N) = 1$ ist. Durch Addition eines Vielfachen der Gleichung $\varphi(N)e - e\varphi(N) = 0$ läßt sich dabei erreichen, daß $1 \le k \le e-1$ ist. Für $e = 3$ kommen also nur $k = 1$ und $k = 2$ in Frage. Man muß daher nur $d_k = \bigl(1 + k\varphi(N)\bigr)/3$ für diese beiden Werte berechnen, und erhält d als das ganzzahlige der beiden Ergebnisse.

Bei so vielen Vorteilen muß es auch Nachteile geben, darunter einen ganz offensichtlichen: Ist nämlich die Nachricht x kleiner als die dritte Wurzel aus N, so ist $x^3 < N$, d.h. der Chiffretext ist einfach x^3.

Die Kubikwurzel aus dieser ganzen Zahl kann natürlich leicht gezogen werden.

Auch für größere Exponenten e sind kurze Nachrichten problematisch. Ein Grund liegt darin, daß die Verschlüsselungsfunktion mit der Multiplikation verträglich ist: Auch im Ring $\mathbb{Z}/NZ$ ist $(yz)^e y^e \cdot z^e$.

Nehmen wir an, unsere Nachricht x habe höchstens 2ℓ Bit, sei also kleiner als $2^{2\ell}$. Dann gibt es eine nicht vernachlässigbare Chance, daß sich $x = yz$ als Produkt zweier Zahlen darstellen läßt, die beide kleiner als 2^ℓ (oder als eine etwas größere Schranke M) sind. Wir können für alle Zahlen y von Null bis M deren Verschlüsselung $y^e \bmod N$ berechnen und die Ergebnisse in einer Tabelle notieren. Um nun vom Chiffretext $c = x^e \bmod N$ auf x zurückzuschließen, berechnen wir für jedes dieser Ergebnisse in $\mathbb{Z}/N$ den Quotienten c/y^e. (Falls sich dieser Quotient nicht bilden läßt, ist y^e nicht teilerfremd zu $N = pq$, und wir erhalten sogar eine Faktorisierung von N; das ist aber für gut gewählte, große N extrem unwahrscheinlich.) Falls einer dieser Quotienten als Eintrag $z^e \bmod N$ in unserer Tabelle auftaucht, haben wir eine Relation der Form $c \equiv y^e \cdot z^e = (yz)^e \bmod N$ gefunden, und damit kennen wir $x = yz$.

Um diese Attacke zu verhindern, muß bei 128-Bit-Sicherheit $\ell \ge 128$ sein, die übermittelten Nachrichten müssen also mindestens 256 Bit lang sein. Auch dann sind wir allerdings noch nicht unbedingt auf der sicheren Seite, denn wenn nur wenige Nachrichten in Frage kommen, kann ein Gegner die einfach alle mit dem öffentlichen Schlüssel chiffrieren und schauen, wann das Ergebnis mit dem Chiffretext übereinstimmt. Beim praktischen Einsatz von RSA werden daher nie einfach die zu übermittelnden Nachrichten übertragen, sondern Blöcke eines vorher festgelegten Formats. Diese Formate sehen vor, daß alle unbenutzten Positionen mit Zufallsbits gefüllt werden und daß jeder Block mindestens 128 Zufallsbits enthält. Auf dem Niveau der 128-Bit-Sicherheit ist dann jede Entschlüsselung durch systematisches Probieren ausgeschlossen, denn Nachrichtenlängen größer 256 Bit sind bei den heute üblichen Parameterwerten ohnehin selbstverständlich.

Falls eine Nachricht an mehrere Empfänger geschickt wird, müssen

– vor allem bei kleinen Verschlüsselungsexponenten wie $e = 3$ – die Zufallsbits für jeden Empfänger neu erzeugt werden, denn wenn jedes Mal derselbe Block x verschlüsselt wird und dabei – wie dies häufig in der Praxis der Fall ist – stets mit drei potenziert wird, kennt ein Gegner anschließend $x^3 \bmod N_i$ für die Moduln N_i der sämtlichen Empfänger, kann also nach dem chinesischen Restesatz x^3 modulo dem Produkt der N_i berechnen, und da dieses Produkt bei mindestens drei Empfängern größer ist als x^3, kennt er x^3 und damit auch x.

Bei der Wahl der Schlüsseldaten geht man stets aus vom öffentlichen Exponenten e und berechnet dann dazu nach dem EUKLIDischen Algorithmus den privaten Exponenten d. Dadurch ist praktisch sichergestellt, daß dieser in der Größenordnung von N liegen wird, und das muß auch so sein: Im Kapitel über Kettenbrüche werden wir sehen, daß N leicht faktorisiert werden kann, wenn d zu klein ist.

Bei der Berechnung von $x^d \bmod N$ und $x^e \bmod N$ darf man natürlich nicht erst x^d *bzw.* x^e berechnen und dann modulo N reduzieren: Schon für rund dreißigstellige Exponenten würde das zu Zwischenergebnissen führen, mit denen selbst die leistungsfähigsten heutigen Supercomputer nicht mehr fertig würden. Um die Länge der Zwischenergebnisse in Grenzen zu halten, muß nach jeder Multiplikation sofort modulo N reduziert werden.

Für große Exponenten e ist es auch nicht mehr möglich, die Potenz durch sukzessive Multiplikation mit x zu berechnen, die benötigten $d-1$ Multiplikationen lägen ebenfalls weit jenseits der Rechenleistung selbst von Supercomputern.

Zum Glück gibt es eine erheblich effizientere Alternative: Um beispielsweise x^{32} zu berechnen brauchen wir keine 31 Multiplikationen, sondern erhalten das Ergebnis über die Formel

$$x^{32} = \left(\left(\left(\left(x^2\right)^2\right)^2\right)^2\right)^2$$

mit nur fünf Multiplikationen (genauer: Quadrierungen).

Entsprechend können wir für jede gerade Zahl $n = 2m$ die Potenz x^n als Quadrat von x^m berechnen. Für einen ungeraden Exponenten e ist $e-1$ gerade, wenn wir also x^e als Produkt von x und x^{e-1} berechnen, können wir zumindest im nächsten Schritt wieder die Formel für gerade Exponenten verwenden. Somit reichen pro Binärziffer des Exponenten ein bis zwei Multiplikationen; der Aufwand wächst also nur proportional zur Stellenzahl von e. Für den ebenfalls recht populären Verschlüsselungsexponenten $e = 2^{16} + 1 = 65537$ beispielsweise braucht man nur 17 Multiplikationen, nicht 65536.

§3: Weitere Anwendungen des RSA-Verfahrens

Im Gegensatz zu symmetrischen Kryptoverfahren endet die Nützlichkeit des RSA-Verfahrens nicht mit der bloßen Möglichkeit einer Verschlüsselung; es erlaubt noch eine ganze Reihe weiterer Anwendungen:

a) Identitätsnachweis

Hier geht es darum, in Zugangskontrollsystemen, vor Geldautomaten oder bei einer Bestellung im Internet die Identität einer Person zu beweisen: Mit RSA ist das beispielsweise dadurch möglich, daß nur der Inhaber des geheimen Schlüssels d zu einer gegebenen Zahl a eine Zahl b berechnen kann, für die $b^e \equiv a \bmod N$ ist. Letzteres wiederum kann jeder überprüfen, der den öffentlichen Schlüssel (N, e) kennt.

Falls also der jeweilige Gegenüber eine Zufallszahl a erzeugt und als Antwort das zugehörige b verlangt, kann er anhand eines öffentlichen Schlüsselverzeichnisses die Richtigkeit von b überprüfen und sich so von der Identität seines Partners überzeugen. Im Gegensatz zu Kreditkarteninformation oder Paßwörtern ist dieses Verfahren auch immun gegen Abhören: Falls jedesmal ein neues zufälliges a erzeugt wird, nützt ein einmal abgehörtes b nichts.

Grundsätzlich bräuchte man hier kein Kryptosystem mit öffentlichen Schlüsseln; in der Tat funktionierten die ersten Freund-/Feinderkennungssysteme für Flugzeuge zur Zeit des zweiten Weltkriegs nach diesem Prinzip, aber damals natürlich mit einem klassischen symmetrischen Kryptosystem, wobei alle Teilnehmer mit demselben Schlüssel arbeiteten. Der Vorteil eines asymmetrischen Systems besteht darin, daß sich keiner der Teilnehmer für einen anderen ausgeben kann, was beispielsweise wichtig ist, wenn man sich gegenüber weniger vertrauenswürdigen Personen identifizieren muß.

Trotzdem ist das Verfahren in dieser Form nicht als Ersatz zur Übertragung von rechtlich bindender Information geeignet, da der Gegenüber anhand des öffentlichen Schlüssels jederzeit zu einer willkürlich gewählten Zahl b die Zahl $a = b^e \bmod N$ erzeugen kann um dann zu behaupten, er habe b als Antwort darauf empfangen. Daher kann der Inhaber des geheimen Schlüssels zwar seine Identität beweisen, aber sein Gegenüber kann später nicht beispielsweise vor Gericht beweisen, daß er dies (zum Beispiel bei einer Geldabhebung oder Bestellung) getan hat. Falls dies eventuell nötig werden könnte, ist das hier vorgestellte Verfahren also ungeeignet; es funktioniert nur zwischen Personen, die einander vertrauen können.

Eine mögliche Modifikation bestünde darin, daß man beispielsweise noch zusätzlich verlangt, daß die Zahl a eine spezielle Form hat, etwa daß die vordere Hälfte der Ziffernfolge identisch mit der hinteren Hälfte ist. Ohne Kenntnis von d hat man praktisch keine Chancen eine Zahl b zu finden, für die $b^e \bmod N$ eine solche Gestalt hat: Bei Zahlen mit $2r$ Ziffern liegt die Wahrscheinlichkeit dafür bei 10^{-r}.

b) Elektronische Unterschriften

Praktische Bedeutung hat vor allem eine andere Variante: die elektronische Unterschrift. Hier geht es darum, daß der Empfänger erstens davon überzeugt wird, daß eine Nachricht tatsächlich vom behaupteten Absender stammt, und daß er dies zweitens auch einem Dritten gegenüber beweisen kann. (In Deutschland sind solche elektronischen Unterschriften, sofern gewisse formale Voraussetzungen erfüllt sind, rechtsverbindlich.)

Um einen Nachrichtenblock a mit $0 \le a < N$ zu unterschreiben, berechnet der Inhaber des öffentlichen Schlüsseln (N, e) mit seinem geheimen Schlüssel d die Zahl

$$b = a^d \bmod N$$

und sendet das Paar (a, b) an den Empfänger. Dieser überprüft, ob

$$b^e \equiv a \bmod N\,;$$

falls ja, akzeptiert er dies als unterschriebene Nachricht a. Da er ohne Kenntnis des geheimen Schlüssels d nicht in der Lage ist, den Block (a, b) zu erzeugen, kann er auch gegenüber einem Dritten beweisen, daß der Absender selbst die Nachricht a unterschrieben hat.

Für kurze Nachrichten ist dieses Verfahren in der vorgestellten Form praktikabel; in vielen Fällen kann man sogar auf die Übermittlung von a verzichten, da $b^e \bmod N$ für ein falsch berechnetes b mit an Sicherheit grenzender Wahrscheinlichkeit keine sinnvolle Nachricht ergibt.

Falls die übermittelte Nachricht geheimgehalten werden soll, müssen a und b natürlich noch vor der Übertragung mit dem öffentlichen Schlüssel des Empfängers oder nach irgendeinem anderen Kryptoverfahren verschlüsselt werden.

Bei langen Nachrichten ist die Verdoppelung der Nachrichtenlänge nicht mehr akzeptabel, und selbst, wenn man auf die Übertragung von a verzichten kann, ist das Unterschreiben jedes einzelnen Blocks sehr aufwendig. Deshalb unterschreibt man meist nicht die Nachricht selbst, sondern einen daraus extrahierten Hashwert. Dieser Wert muß natürlich erstens von der gesamten Nachricht abhängen, und zweitens muß es für den Empfänger (praktisch) unmöglich sein, zwei Nachrichten zu erzeugen, die zum gleichen Hashwert führen. Letzteres bedeutet wegen des sogenannten *Geburtstagsparadoxons*, daß für n-Bit Sicherheit Hashwerte der Länge $2n$ erforderlich sind. Die immer noch recht verbreiteten Hashalgorithmen, die 160 Bit liefern, haben somit nur eine Sicherheit von etwa 80 Bit, was heute nicht mehr als wirklich sicher gelten kann. Neuere Hashalgorithmen liefern Werte mit 224 oder 256 Bit, was einer 112- oder 128-Bit Sicherheit entspricht. Die Algorithmen funktionieren ähnlich wie symmetrische Kryptoverfahren; sie versuchen durch Konfusion und Diffusion ein Ergebnis zu berechnen, dessen sämtliche Bits in einer nicht offensichtlichen Weise von jedem einzelnen Nachrichtenbit abhängen.

Eine wichtige Anwendung elektronischer Unterschriften ist übrigens auch die Veröffentlichung von RSA-Schlüsseln: Falls es einem Angreifer gelingt, einem Teilnehmer A einen falschen öffentlichen Schlüssel von Teilnehmer B unterzuschieben, kann (nur) der Angreifer die Nachrichten von A an B lesen, und er kann sich gegenüber A mittels elektronischer Unterschrift als B ausgeben. Daher sind öffentliche Schlüssel meist unterschrieben von einer Zertifizierungsstelle. Auch deren Unterschrift muß natürlich gegen Manipulationen gesichert sein, beispielsweise indem sie von der nächsthöheren Zertifizierungsstelle unterschrieben ist. An der Spitze der Zertifizierungshierarchie stehen Stellen, deren elektronische Unterschrift jeder Teilnehmer kennen sollte, weil es sich entweder um staatliche Stellen handelt, deren elektronische Unterschriften auf leicht zugänglichen Webseiten verifiziert werden können, oder aber – in der Praxis häufiger – weil die Unterschriften dieser Stellen in Mail- und Browserprogramme eingebaut sind. Letzteres bietet selbstverständlich keine Sicherheit gegen manipulierte Browserprogramme aus dubiosen Quellen, die möglicherweise auch die Mafia als Zertifizierungsstelle anerkennen.

c) Blinde Unterschriften und elektronisches Bargeld

Einer der erfolgversprechendsten Ansätze zum Aushebeln eines Kryptosystems besteht darin, sich auf die Dummheit seiner Mitmenschen zu verlassen.

So sollte es durch gutes Zureden nicht schwer sein, jemanden zu Demonstrationszwecken zum Unterschreiben einer sinnlosen Nachricht zu bewegen: Eine Folge von Nullen und Einsen ohne sinnvolle Interpretation hat schließlich keine rechtliche Wirkung.

Nun muß eine sinnlose Nachricht aber nicht unbedingt eine Zufallszahl sein: Sie kann sorgfältig präpariert sein. Sei dazu etwa m eine Nachricht, die ein Zahlungsversprechen enthält, (N, e) der öffentliche Schlüssel des Opfers und r eine Zufallszahl zwischen 2 und $N - 2$. Dann wird

$$x = m \cdot r^e \bmod N$$

wie eine Zufallsfolge aussehen, für die man eine Unterschrift

$$u = x^d \bmod N = (mr^e)^d \bmod N = m^d r \bmod N$$

bekommt. Multiplikation mit r^{-1} macht daraus eine Unterschrift unter die Zahlungsverpflichtung m.

Das angegebene Verfahren kann nicht nur von Trickbetrügern benutzt werden; blinde Unterschriften sind auch die Grundlage von *digitalem Bargeld.*

Zahlungen im Internet erfolgen meist über Kreditkarten; die Kreditkartengesellschaften haben also einen recht guten Überblick über die Ausgaben ihrer Kunden und machen teilweise auch recht gute Geschäfte mit Kundenprofilen.

Digitales Bargeld will die Anonymität von Geldscheinen mit elektronischer Übertragbarkeit kombinieren und so ein anonymes Zahlungssystem z.B. für das Internet bieten.

Es wir ausgegeben von einer Bank, die für jede angebotene Stückelung einen öffentlichen Schlüssel (N, e) bekanntgibt. Eine Banknote ist eine mit dem zugehörigen geheimen Schlüssel unterschriebene Seriennummer.

Die Seriennummer kann natürlich nicht einfach *jede* Zahl sein; sonst wäre jede Zahl kleiner N eine Banknote. Andererseits dürfen die Seriennummern aber auch nicht von der Bank vergeben werden, denn sonst wüßte diese, welcher Kunde Scheine mit welchem Seriennummern hat. Als Ausweg wählt man Seriennummern einer sehr speziellen Form: Ist $N > 10^{150}$, kann man etwa als Seriennummer eine 150-stellige Zahl wählen, deren Ziffern spiegelsymmetrisch zur Mitte sind, d.h. ab der 76. Ziffer werden die vorherigen Ziffern rückwärts wiederholt. Die Wahrscheinlichkeit, daß eine zufällige Zahl x nach Anwendung des öffentlichen Exponenten auf so eine Zahl führt, ist 10^{-75} und damit vernachlässigbar.

Seriennummern werden von den Kunden zufällig erzeugt. Für jede solche Seriennummer m erzeugt der Kunde eine Zufallszahl r, schickt $mr^e \bmod N$ an die Bank und erhält (nach Belastung seines Kontos) eine Unterschrift u für diese Nachricht zurück. Wie oben berechnet er daraus durch Multiplikation mit r^{-1} die Unterschrift $v = m^d \bmod N$ für die Seriennummer N, und mit diesem Block kann er bezahlen.

Der Zahlungsempfänger berechnet $v^e \bmod N$; falls dies die Form einer gültigen Seriennummer hat, kann er sicher sein, einen von der Bank unterschriebenen Geldschein vor sich zu haben. Er kann allerdings noch nicht sicher sein, daß dieser Geldschein nicht schon einmal ausgegeben wurde.

Deshalb muß er die Seriennummer an die Bank melden, die mit ihrer Datenbank bereits ausbezahlter Seriennummern vergleicht. Falls sie darin noch nicht vorkommt, wird sie eingetragen und der Händler bekommt sein Geld; andernfalls verweigert sie die Zahlung.

Bei 10^{75} möglichen Nummern liegt die Wahrscheinlichkeit dafür, daß zwei Kunden, die eine (wirklich) zufällige Zahl wählen, dieselbe Nummer erzeugen, bei etwa $10^{-37,5}$. Die Wahrscheinlichkeit, mit jeweils einem Spielschein fünf Wochen lang hintereinander sechs Richtige im Lotto zu haben, liegt dagegen bei $\binom{49}{6}^{-5} \approx 5 \cdot 10^{-35}$, also etwa um den Faktor sechzig höher. Zwei gleiche Seriennummern sind also praktisch auszuschließen, wenn auch theoretisch möglich.

Falls wirklich einmal zufälligerweise zwei gleiche Seriennummern erzeugt worden sein sollten, kann das System nur funktionieren, wenn der zweite Geldschein mit derselben Seriennummer nicht anerkannt wird, so daß der zweite Kunde sein Geld verliert. Dies muß als eine zusätzliche Gebühr gesehen werden, die mit an Sicherheit grenzender Wahrscheinlichkeit nie fällig wird, aber trotzdem nicht ausgeschlossen werden kann.

Da digitales Bargeld nur in kleinen Stückelungen sinnvoll ist (Geldscheinen im Millionenwert wären auf Grund ihrer Seltenheit nicht wirklich anonym und würden wegen der damit verbundenen Möglichkeiten zur Geldwäsche auch in keinem seriösen Wirtschaftssystem akzeptiert), wäre der theoretisch mögliche Verlust ohnehin nicht sehr groß.

Digitales Bargeld der gerade beschriebenen Form wurde 1982 von DAVID CHAUM vorgestellt; 1990 gründete er eine Firma namens DigiCash, die es kommerziell vermarkten sollte. Zu deren Kunden gehörte beispielsweise auch die Deutsche Bank, die allerdings nur 27 Kunden fand, die Zahlungen damit akzeptierten. DigiCash wurde 1998 zahlungsunfähig; derzeit gibt es meines Wissens keine ähnlichen Zahlungssysteme.

d) Bankkarten mit Chip

In Deutschland und den meisten anderen Ländern hat eine Bankkarte einen Magnetstreifen, auf dem die wichtigsten Informationen wie Kontenname und -nummer, Bankleitzahl, Gültigkeitsdauer *usw.* gespeichert sind; dazu kommt verschlüsselte Information, die unter anderem die Geheimzahl enthält, die aber auch von den obengenannten Daten abhängt. Zur Verschlüsselung verwendet man hier ein konventionelles, d.h. symmetrisches Kryptoverfahren; derzeit noch meist Triple-DES.

Der Schlüssel, mit dem dieses arbeitet, muß natürlich streng geheimgehalten werden: Wer ihn kennt, kann problemlos die Geheimzahlen fremder Karten ermitteln und eigene Karten zu beliebigen Konten erzeugen.

Um eine Karte zu überprüfen, muß daher eine Verbindung zu einem Zentralrechner aufgebaut werden, an den sowohl der Inhalt des Magnetstreifens als auch die vom Kunden eingetippte Zahl übertragen werden; dieser wendet Triple-DES mit dem Systemschlüssel an und meldet dann, wie die Prüfung ausgefallen ist.

In Frankreich haben die entsprechenden Karten zusätzlich zum Magnetstreifen noch einen Chip, in dem ebenfalls die Kontendaten gespeichert sind sowie, in einem auslesesicheren Register, Informationen über die Geheimzahl. Dort wird die ins Lesegerät eingetippte Geheimzahl nicht an den Zentralrechner übertragen, sondern an den Chip, der sie überprüft und akzeptiert oder auch nicht.

Da frei programmierbare Chipkarten relativ billig sind, muß dafür Sorge getragen werden, daß ein solches System nicht durch einen *Yes-Chip* unterlaufen werden kann, der ebenfalls die Konteninformationen enthält, ansonsten aber ein Programm, das ihn *jede* Geheimzahl akzeptieren läßt. Das Terminal muß also, bevor es überhaupt eine Geheimzahl anfordert, zunächst einmal den Chip authentisieren, d.h. sich davon überzeugen, daß es sich um einen vom Bankenkonsortium ausgegebenen Chip handelt.

Aus diesem Grund sind die Kontendaten auf dem Chip mit dem privaten RSA-Schlüssel des Konsortiums unterschrieben. Die Terminals kennen den öffentlichen Schlüssel dazu und können so die Unterschrift überprüfen.

Diese Einzelheiten und speziell deren technische Implementierung wurden vom Bankenkonsortium zunächst streng geheimgehalten. Trotzdem machte sich 1997 ein elsässischer Ingenieur namens SERGE HUMPICH daran, den Chip genauer zu untersuchen. Er verschaffte sich dazu ein (im freien Verkauf erhältliches) Terminal und untersuchte sowohl die Kommunikation zwischen Chip und Terminal als auch die Vorgänge innerhalb des Terminals mit Hilfe eines Logikanalysators. Damit gelang es ihm nach und nach, die Funktionsweise des Terminals zu entschlüsseln und in ein äquivalentes PC-Programm zu übersetzen. Durch dessen Analyse konnte er die Authentisierungsprozedur und die Prüflogik entschlüsseln und insbesondere auch feststellen, daß hier mit RSA gearbeitet wurde.

Blieb noch das Problem, den Modul zu faktorisieren. Dazu besorgte er sich ein japanisches Programm aus dem Internet, das zwar eigentlich für kleinere Zahlen gedacht war, aber eine Anpassung der Wortlänge ist natürlich auch für jemanden, der den Algorithmus hinter dem Programm nicht versteht, kein Problem. Nach sechs Wochen Laufzeit hatte sein PC damit den Modul faktorisiert:

$$\begin{aligned} & 213598703592091008239502270499962879705109534182\backslash \\ & 641740644252416500858395774644508840500943086599 9 \\ = {} & 1113954325148827987925490175477024844070922844843 \\ \times {} & 1917481702524504439375786268230862180696934189293 \end{aligned}$$

Als er seine Ergebnisse über einen Anwalt dem Bankenkonsortium mitteilte, zeigte sich, was dieses sich unter Sicherheitsstandards vorstellt: Es erreichte, daß HUMPICH wegen des Eindringens in ein DV-System zu zehn Monaten Haft auf Bewährung sowie einem Franc Schadenersatz plus Zinsen verurteilt wurde; dazu kamen 12 000 F Geldstrafe. Einzelheiten findet man in seinem Buch

SERGE HUMPICH: Le cerveau bleu, Xo, 2001

Seit November 1999 haben neu ausgegebene Bankkarten noch ein zusätzliches Feld mit einer Unterschrift, die im Gegensatz zum obigen 320-Bit-Modul einen 768-Bit-Modul verwendet. Natürlich können damit erzeugte Unterschriften nur von neueren Terminals überprüft werden, so daß viele Transaktionen weiterhin nur über den 320-Bit-Modul mit inzwischen wohlbekannter Faktorisierung „geschützt“ sind.

§4: Wie groß sollten die Primzahlen sein?

Das Beispiel der französischen Bankkarten zeigt, daß RSA höchstens dann sicher ist, wenn die Primzahlen p und q hinreichend groß gewählt werden. Als erstes müssen wir uns daher die Frage stellen, wie groß eine „hinreichend große“ Zahl heute sein muß.

Ein treu sorgender Staat läßt seine Bürgern bei einer derart wichtigen Frage natürlich nicht allein: Zwar gibt es noch keine oberste Bundesbehörde für Primzahlen, aber das Bundesamt für Sicherheit in der Informationstechnik (BSI) und die Bundesnetzagentur für Elektrizität, Gas, Telekommunikation, Post und Eisenbahnen publizieren jedes Jahr ein Dokument mit dem Titel *Geeignete Kryptoalgorithmen zur Erfüllung der Anforderungen nach* §*17 Abs. 1 bis 3 SigG vom 22. Mai 2001 in Verbindung mit Anlage 1 Abschnitt I Nr. 2 SigV vom 22. November 2001.*

SigV steht für die aufgrund des Signaturgesetzes SigG erlassene Signaturverordnung; beide gemeinsam legen fest, daß elektronische Unterschriften in Deutschland grundsätzlich zulässig und rechtsgültig sind, sofern gewisse Bedingungen erfüllt sind. Zu diesen Bedingungen gehört unter anderem, daß das Verfahren und die Schlüssellänge gemeinsam einen „geeigneten Kryptoalgorihmus“ im Sinne der jeweils gültigen Veröffentlichung der Bundesnetzagentur ist.

Da Rechner immer schneller und leistungsfähiger werden und auch auf der mathematisch-algorithmischen Seite fast jedes Jahr kleinere oder größere Fortschritte zu verzeichnen sind, gelten die jeweiligen Empfehlungen nur für etwa sechs Jahre. Für Dokumente, die länger gültig sein sollen, sind elektronische Unterschriften also nicht vorgesehen.

Offiziell geht bei den Empfehlungen allgemein um geeignete Algorithmen für elektronische Unterschriften sowie deren Schlüssellängen, aber wie die Entwicklung der letzten Jahre zeigte, drehen sich die Diskussionen, die zu den jeweiligen Empfehlungen führen, tatsächlich fast ausschließlich um die jeweils notwendige Schlüssellänge für RSA.

Natürlich hat in einer Demokratie bei so einer wichtigen Frage auch die Bevölkerung ein Mitspracherecht; deshalb beginnt das BSI jeweils zunächst einen Entwurf, zu dem es um Kommentare bittet; erst einige Monate später wird die endgültige Empfehlung verkündet und im Bundesanzeiger veröffentlicht.

Die interessierte Öffentlichkeit, von der die Kommentare zu den Entwürfen kommen, besteht einerseits aus Anbietern von Hardware und Software für Kryptographie, und als erfahrene Experten für Datensicherheit wissen diese, daß ein Verfahren nur dann wirklich geeignet sein kann, wenn es die eigene Firma im Angebot hat. (Am geeignetsten sind natürlich die Verfahren, die keines der Konkurrenzunternehmen anbietet.)

Andererseits melden sich die Anwender von Kryptoverfahren zu Wort; vor allem sind das Vertreter der Dachverbände des Kreditgewerbes. Diese müssen für eine starke Kryptographie eintreten, denn falls die Kryptographie einer von ihnen ausgegebenen Chipkarte geknackt wird, könnte das für ihre Mitglieder sehr teuer werden. Teuer wird es aber auch, wenn Chipkarten vor Ablauf ihrer Gültigkeit ausgetauscht werden müssen, weil sie nicht mehr den aktuellen Anforderungen entsprechen. Da Chipkarten ein bis zwei Jahre vor Ausgabe in Auftrag gegeben werden müssen und dann im allgemeinen drei Jahre lang gültig sind, versucht dieser Teil der Öffentlichkeit vor allem, die von den Kryptologen für notwendig erachteten Änderungen um ein bis zwei Jahre hinauszuzögern.

Das endgültige Ergebnis ist dann ein Kompromiss zwischen den verschiedenen Positionen.

So ist beispielsweise zu erklären, daß es vieler Anläufe bedurfte, um die Schlüssellänge für RSA auf einen Wert über 1024 Bit zu bringen, denn es gab viele Chips mit Hardware-Implementierungen von RSA für Schlüssellängen von bis zu 1024 Bit, während größere Schlüssellängen zunächst vor allem in *public domain* Software wie PGP zu finden waren.

Bis Ende 2000 galten 768 Bit als ausreichende Größe für das Produkt N der beiden Primzahlen, jener Wert also, den die *neueren* französischen Bankkarten verwenden und den ebenfalls nur die neueren Terminals lesen können. Schon in den Richtlinien für 1998 wurden 768 Bit jedoch ausdrücklich nur übergangsweise zugelassen; längerfristig, d.h. bei Gültigkeit über 2000 hinaus, waren mindestens 1024 Bit vorgeschrieben.

Die Richtlinien für 2000 erlaubten die 768 Bit ebenfalls noch bis zum Ende des Jahres; für Dokumente mit einer längeren Gültigkeit verlangten sie bis Mitte 2005 eine Mindestgröße von 1024 Bit, danach bis Ende 2005 sogar 2048 Bit.

Anbieterproteste führten dazu, daß nach den Richtlinien von 2001 eine Schlüssellänge von 1024 dann doch noch bis Ende 2006 sicher war; die Schlüssellänge 2048 war nur noch „empfohlen“, also nicht mehr verbindlich.

Im April 2002 erschien der erste Entwurf für die 2002er Richtlinien; darin war für 2006 und 2007 nur eine Mindestlänge von 2048 Bit wirklich sicher. Einsprüche führten im September 2002 zu einem revidierten Entwurf, wonach 2006 doch noch 1024 Bit reichen, 2007 aber mindestens 1536 notwendig werden. Die Mindestlänge von 2048 Bit wurde wieder zur „Empfehlung“ zurückgestuft.

Am 2. Januar 2003 erschienen endlich die offiziellen Richtlinien des Jahres 2002; veröffentlicht wurden sie am 11. März 2003 im Bundesanzeiger Nr. 48, S. 4202–4203. Danach reichen 1024 Bit auch noch bis Ende 2007, erst 2008 werden 1280 Bit erforderlich. Die 2048 Bit blieben dringend empfohlen.

Nach diesem großen Kraftakt erschienen 2003 keine neuen Richtlinien mehr; erst für 2004 gab es am 2. Januar 2004 neue Empfehlungen (Bundesanzeiger Nr. 30 vom 13. Februar 2004, S. 2537–2538). Für den Zeitraum bis Ende 2008 wurden die alten Empfehlungen beibehalten, bis Ende 2009 aber 1536 Bit gefordert. Die nächsten Richtlinien für 2005 sahen in ihrem ersten Vorentwurf 2048 Bit bis Ende 2010 vor; nach Einsprüchen der Banken, daß das Betriebssystem SECCOS der heute üblichen Chipkarten nur mit maximal 1984 Bit-Schlüsseln umgehen kann, wurde die Länge im zweiten Entwurf auf 1984 gesenkt; in den endgültigen Richtlinien vom 2. Januar 2005 waren es schließlich nur noch 1728.

Die neuesten Richtlinien stammen vom 17. November 2008 und wurden am 27. Januar 2009 im Bundesanzeiger Nr. 13, S. 346 veröffentlicht. Sie empfehlen grundsätzlich schon heute 2048 Bit, aber wirklich verbindlich sind

bis Ende	2008	2009	2010	2015	
Minimallänge	1280	1536	1728	1976	*Bit.*

(1976 unterscheidet sich nicht wesentlich von 2048; der minimal kleinere Wert wurde in Hinblick auf die oben erwähnten Probleme mit SECCOS gewählt.)

Die beiden Primfaktoren p, q sollen zufällig und unabhängig voneinander erzeugt werden und aus einem Bereich stammen, in dem

$$\varepsilon_1 < |\log_2 p - \log_2 q| < \varepsilon_2$$

gilt. Als *Anhaltspunkte* werden dabei die Werte

$$\varepsilon_1 = 0{,}1 \quad \text{und} \quad \varepsilon_2 = 30$$

vorgeschlagen; ist p die kleinere der beiden Primzahlen, soll also

$$2^{-10}p < q < 2^{30}p \approx 10^9 p$$

gelten, d.h. die beiden Primzahlen sollten zwar ungefähr dieselbe Größenordnung haben, aber nicht zu nahe beieinander liegen. Der Grund dafür ist ein von FERMAT entdecktes Faktorisierungsverfahren auf Grundlage der dritten binomischen Formel: Falls für eine Zahl N und eine natürliche Zahl y die Zahl $N + y^2$ eine Quadratzahl x^2 ist, ist $N = x^2 - y^2 = (x + y)(x - y)$, womit zwei Faktoren gefunden sind. Probiert man alle kleinen natürlichen Zahlen y systematisch durch, führt dieses Verfahren offensichtlich umso schneller zum Erfolg, je näher die beiden Faktoren von N beieinander liegen. Wir werden uns in Kapitel über Faktorisierung noch genauer damit befassen.

§5: Verfahren mit diskreten Logarithmen

Kurz nach der Veröffentlichung des RSA-Algorithmus fanden auch DIFFIE und HELLMAN ein Verfahren, das im Gegensatz zu RSA sogar ganz ohne vorvereinbarte Schlüssel auskommt: Zwei Personen vereinbaren über eine unsichere Leitung einen Schlüssel, den anschließend nur sie kennen.

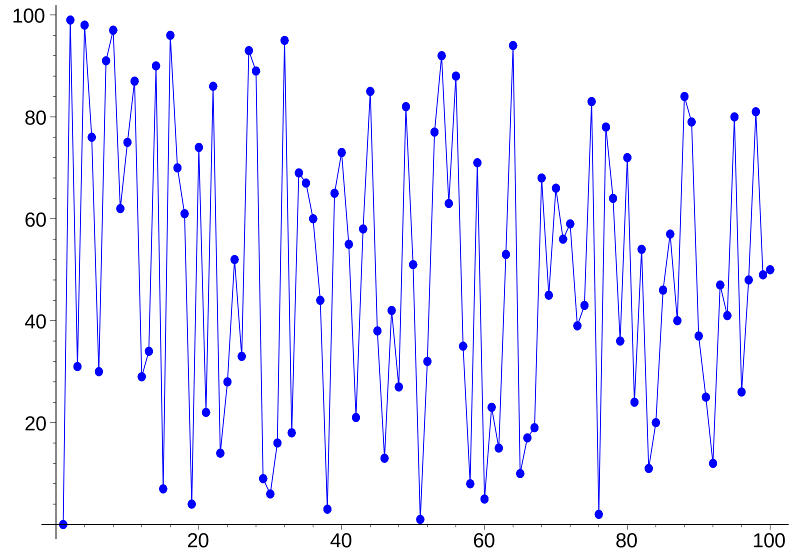
Ausgangspunkt ist wieder das Potenzieren im Körper $\mathbb{F}_p$; hier betrachten wir aber die Exponentialfunktion $x \mapsto a^x$ zu einer geeigneten Basis a. Ihre Umkehrfunktion bezeichnet man als *Index* oder *diskreten Logarithmus* zur Basis a:

$$y = a^x \Longrightarrow x = \log_a y\,.$$

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 es in der Abbildung zu sehen ist. Auch ist im Reellen der Logarithmus zur Basis $a > 1$ für jede positive 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 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 Potenzfunktion und Logarithmen kann kryptologisch ausgenutzt werden.

Als Körper verwendet man entweder Körper von Zweipotenzordnung, die wir weiter unten betrachten werden, oder Körper von Primzahl-



Die Funktion $\log_{51} x$ in $\mathbb{F}_{101}$

ordnung. Da es für viele interessante Körper von Zweipotenzordnung bereits Chips gibt, die dort diskrete Logarithmen berechnen, dürften Körper von Primzahlordnung bei ungefähr gleicher Elementanzahl wohl etwas sicherer sein: Es gibt einfach viel mehr Primzahlen als Zweierpotenzen, und jeder Fall erfordert einen neue Hardwareentwurf. Falls man die Primzahlen hinreichend häufig wechselt, dürfte sich dieser Aufwand für kaum einen Gegner lohnen.

Da Körper von Primzahlordnung auch einfacher sind als solche von Primzahlpotenzordnung, wollen wir uns zunächst auf diese beschränken; die spätere Übertragung des Algorithmus auf Körper von Zweipotenzordnung sollte dem Leser keine Schwierigkeiten machen.

Beim DIFFIE-HELLMAN-Verfahren, dem ältesten auf der Grundlage diskreter Logarithmen, geht es wie gesagt darum, daß zwei Teilnehmer, die weder über gemeinsame Schlüsselinformation noch über eine sichere Leitung verfügen, einen Schlüssel vereinbaren wollen.

Dazu einigen sie sich zunächst (über die unsichere Leitung) auf eine

Primzahl p und eine natürliche Zahl a derart, daß die Potenzfunktion $x \mapsto a^x$ möglichst viele Werte annimmt.

Als nächstes wählt Teilnehmer A eine Zufallszahl $x < p$ und B entsprechend $y < p$; A schickt $u = a^x \bmod p$ an B und erhält dafür $v = a^y \bmod p$.

Sodann berechnet A die Zahl

$$v^x \bmod p = \left(a^y\right)^x \bmod p = a^{xy} \bmod p$$

und B entsprechend

$$u^y \bmod p = \left(a^x\right)^y \bmod p = a^{xy} \bmod p\,;$$

beide haben also auf verschiedene Weise dieselbe Zahl berechnet, die sie nun als Schlüssel in einem klassischen Kryptosystem verwenden können, wobei sie sich wohl meist auf einen Teil der Bits beschränken müssen, da solche Schlüssel typischerweise eine Länge von 128 bis 256 Bit haben, während die Primzahl p erheblich größer sein muß.

Ein Gegner, der den Datenaustausch abgehört hat, kennt die Zahlen p, a, u und v; um $a^{xy} \bmod p$ zu finden, muß er den diskreten Logarithmus von u oder v berechnen.

Mit den besten heute bekannten Algorithmen ist die möglich, wenn p eine Primzahl von bis zu etwa 200 Dezimalstellen ist; dies entspricht etwa 665 Bit. Auch in diesem Fall dauert die Berechnung allerdings selbst bei massiver Parallelisierung über das Internet mehrere Monate, gefolgt von einer Schlußrechnung auf einem Supercomputer.

Natürlich gibt es keine Garantie, daß kein Gegner mit einem besseren als den bislang bekannten Verfahren diskrete Logarithmen oder Faktorisierungen auch in weitaus größeren Körpern berechnen kann. Dazu bräuchte er allerdings einen Durchbruch entweder auf der mathematischen oder auf der technischen Seite, für den weit und breit keine Grundlage zu sehen ist.

Trotzdem gibt es einen verhältnismäßig einfachen Angriff auf den Schlüsselaustausch nach DIFFIE und HELLMAN, die sogenannte *man in the middle attack*. Dabei unterbricht der Angreifer die Verbindung zwischen A und B und gibt sich gegenüber A als B aus und umgekehrt. So kann er mit beiden Teilnehmern je einen Schlüssel vereinbaren, und die damit verschlüsselte Kommunikation kann (nur) von ihm gelesen und gegebenenfalls manipuliert werden. In der vorgestellten Form funktioniert das Verfahren also nur, wenn man sicher sein weiß, mit wem man kommuniziert.

§6: 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 Bundesnetzagentur empfohlenen „Geeigneten Algorithmen" zählt.

Aus Sicht der amerikanischen Behörden hat DSA gegenüber RSA und Verfahren wie DIFFIE-HELLMAN vor allem einen großen Vorteil: Es läßt sich *nur* für elektronische Unterschriften benutzen, nicht zur Verschlüsselung.

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 im ursprünglichen Standard eine Länge von mindestens 160 Bit haben sollte. Laut Bundesnetzagentur sollte diese Länge auch noch bis Ende 2009 ausreichen, bis Ende 2012 sind allerdings nach dem Entwurf für 2007 mindestens 224 Bit vorgeschrieben, was wahrscheinlich mehr mit den verwendeten Hashfunktionen als mit der Sicherheit der Unterschrift zu tun hat.

Zu dieser Primzahl q sucht man eine Primzahl $p \equiv 1 \bmod q$, für deren Länge die Bundesnetzagentur bis Ende 2007 mindestens 1024 Bit vorschreibt, bis Ende 2008 mindestens 1280, bis Ende 2009 mindestens 1536 und bis Ende 2012 mindestens 2048. „Empfohlen" sind auch hier 2048 Bit.

Daß diese Zahlen (bis auf die unwesentliche Differenz zwischen 2048 und 1976) mit den RSA-Modullängen für die entsprechenden Jahre übereinstimmen, ist kein Zufall: Auch wenn kein direkter Zusammenhang zwischen Faktorisierung und der Berechnung diskreter Logarithmen bekannt ist, hat bislang doch jede neue Idee für einen Faktorisierungsalgorithmus auch zu einem Algorithmus zur Berechnung diskreter Logarithmen geführt, und die auch Laufzeiten dieser Algorithmen sind bei gleicher Zahlenlänge ungefähr gleich.

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 ein neues g_0 betrachtet werden.

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 *bzw.* 224 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.

§7: Anwendungen bei SSL/TLS

SSL steht für *secure socket layer*, TLS für *transport layer security;* Zweck ist jeweils der Aufbau einer sicheren Internetverbindung.

Wie im Internet üblich, können dazu die verschiedensten Verfahren benutzt werden; die auf Grundlage von RSA zählen derzeit zu den populärsten.

Natürlich ist RSA zu aufwendig, um damit eine längere Kommunikation wie beispielsweise eine *secure shell* Sitzung zu verschlüsseln; tatsächlich dient RSA daher nur zur Übertragung eines Schlüssels für ein konventionelles Kryptoverfahren wie AES, IDEA oder Triple-DES, auf das sich die Beteiligten unter SSL/TLS ebenfalls einigen müssen.

Am einfachsten wäre es, wenn der Client einen Schlüssel für ein solches Verfahren wählt und dann diesen mit dem RSA-Schlüssel des Servers verschlüsselt an diesen schickt – vorausgesetzt, er kennt diesen RSA-Schlüssel. Letzteres ist im allgemeinen nicht der Fall; daher muß zunächst der Server dem Client seinen Schlüssel mitteilen.

Da der Client nicht sicher sein kann, mit dem richtigen Server verbunden zu sein, schickt er diesen Schlüssel meist zusammen mit einem Zertifikat, das sowohl seine Identität als auch seinen RSA-Schlüssel enthält und von einer Zertifizierungsstelle unterschrieben ist.

Die öffentlichen Schlüssel der gängigen Zertifizierungsstellen sind in die Browserprogramme eingebaut; bei weniger bekannten Zertifizierungsstellen wie etwa dem Rechenzentrum der Universität Mannheim

fragt der Browser den Benutzer, ob er das Zertifikat anerkennen will oder nicht. Bei *secure shell* schließlich, wo die Gegenseite typischerweise keinerlei Zertifikat vorweisen kann, fragt das Programm beim ersten Verbindungsaufbau zu einem Server, ob dessen Schlüssel anerkannt werden soll und speichert dann einen sogenannten *fingerprint* davon; dieser wird bei späteren Verbindungen zur Identitätsfeststellung benutzt.

§8: Ausblick

Dieses kurze Kapitel konnte selbstverständlich keine umfassende Übersicht über die Kryptographie oder auch nur die asymmetrische Kryptographie geben: Auch das RSA-Verfahren kann mit anderen Methoden angegriffen werden als der direkten Faktorisierung des Moduls. Eine dieser Methoden werden wir im Kapitel über Kettenbrüche kennenlernen, und auch sonst werden im weiteren Verlauf der Vorlesungen noch gelegentlich Themen aus der Kryptologie angeschnitten werden.

Mit Ausnahme von Verfahren wie dem sogenannten *one time pad* gibt es für keines der heute benutzten Kryptoverfahren einen Sicherheitsbeweis, nicht einmal in dem Sinn, daß man den Aufwand eines Gegners zum Knacken des Verfahrens in irgendeiner realistischen Weise nach unten abschätzen könnte. Seriöse Kryptographie außerhalb des Höchstsicherheitsbereichs muß sich daher damit begnügen, daß die Verantwortlichen für den Einsatz eines Verfahrens und der Wahl seiner Parameter (wie den Primzahlen bei RSA) darauf achten, auf dem neuesten Stand der Forschung zu bleiben und ihre Wahl so treffen, daß nicht nur die bekannten Angriffsmethoden versagen, sondern daß auch noch ein recht beträchtlicher Sicherheitszuschlag für künftige Entwicklungen und für nicht publizierte Entwicklungen bleibt.

Auf ewige Sicherheit kann man mit Verfahren wie RSA ohnehin nicht hoffen: Als RSA 1977 von MARTIN GARDNER im *Scientific American* vorgestellt wurde, bekam er von RIVEST, SHAMIR und ADLEMAN die 129-stellige Zahl

114381625757888867669235779976146612010218296721242362562561842935\
70693524573389783059712356395870505898907514759929002687954 3541

(seither bekannt als RSA-129) und eine damit verschlüsselte Nachricht, für deren Entschlüsselung die drei einen Preis von hundert Dollar ausgesetzt hatten. Sie schätzten, daß eine solche Entschlüsselung etwa vierzig Quadrillionen ($4 \cdot 10^{25}$) Jahre dauern würde. (Heute sagt RIVEST, daß dies auf einem Rechenfehler beruhte.) Tatsächlich wurde der Modul 1994 faktorisiert in einer gemeinsamen Anstrengung von 600 Freiwilligen, deren Computer immer dann, wenn sie nichts besseres zu tun hatten, daran arbeiteten. Nach acht Monaten war die Faktorisierung gefunden: Die obige Zahl ist gleich

$$\begin{aligned} &3490529510847650949147849619903898133417764638493387843990820577 \\ \times &32769132993266709549961988190834461413177642967992942539798288533\,. \end{aligned}$$

Mit dem Schema $A = 01$ bis $Z = 26$ und Zwischenraum gleich 00 war die Nachricht *The Magic Words are Squeamish Ossifrage* dann schnell entschlüsselt.

Auch bei heute den heute als sicher geltenden symmetrischen Kryptoverfahren rechnet niemand ernsthaft damit, daß sie noch in hundert Jahren sicher sind: Diese Verfahren werden üblicherweise so gewählt, daß man auf eine Sicherheit für etwa dreißig Jahren hoffen kann – sicher kann aber auch das niemand vorhersagen.

Falls sich sogenannte *Quantencomputer* realisieren lassen, werden alle heute bekannten Verfahren der Kryptographie mit öffentlichen Schlüsseln, egal ob mit diskreten Logarithmen, RSA oder elliptischen Kurven, unsicher sein. Bislang können Quantencomputer kaum mit acht Bit rechnen, und nicht alle Experten sind davon überzeugt, daß es je welche geben wird, die mit mehreren Tausend Bit rechnen können.

Wer mehr über Kryptographie wissen will, findet einen ersten Überblick beispielsweise bei

BUCHMANN: Einführung in die Kryptographie, *Springer*, ^{3}2004

oder natürlich auch in der hier immer wieder angebotenen Kryptologie-Vorlesung.

Mehr über die Geschichte der Kryptographie mit öffentlichen Schlüsseln ist (mathematikfrei) zu finden in

STEVEN LEVY: **crypto:** how the rebels beat the government – saving privacy in the digital age, *Penguin Books*, 2002