Kapitel 5
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 $y = a^y \bmod p$ 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 zum Beispiel verwenden können, um daraus einen Schlüssel für ein symmetrisches Kryptosystem zu bestimmen. Verfahren dazu gibt es mehr als genug: Sie könnten etwa die letzten oder mittleren oder sonst irgendwelche Bits dieser Zahl verwenden, aber auch einen irgendwie definierten Hashwert.

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 zu 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 eine vielversprechende 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 verschiedene solche Zahlen z_A und z_B für jeden der beiden Teilnehmer.

Wenn A die Zahl $u = a^x \bmod p$ an B schickt, fängt G diese Nachricht ab und ersetzt sie durch $w_B = a^{z_B} \bmod p$; entsprechend fängt er Bs Nachricht $y = a^y \bmod p$ ab und schickt stattdessen $w_A = a^{z_A}$ an A. Dies führt dazu, daß am Ende 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 damit verschlüsselte Nachrichten 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 seinen 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 x und y vor dem Versenden unterschreiben – aber dann verschwindet auch wieder der Vorteil, daß sie ohne Kenntnis irgendeines Schlüssels miteinander kommunizieren können: Zur Verifikation einer Unterschrift braucht man schließlich den öffentlichen Schlüssel des Unterschreibenden.

Falls sich A und B hinreichend gut kennen, um die Stimme des jeweils anderen am Telephon einigermaßen sicher 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-Schlüssel leisten kann, werden Schlüssel für symmetrische Kryptoverfahren in der Praxis sehr viel häufiger via RSA vereinbart als via DIFFIE-HELLMAN.

§ 2: Verschlüsselung und elektronische Unterschriften

Zwischen RSA und den Verfahren mit diskreten Logarithmen gibt es einen ganz wesentlichen Unterschied: Wer die Faktorisierung des RSA-Moduls N kennt, kann die sonst schwer zugängliche Umkehrfunktion von $x \mapsto x^e \bmod N$ leicht berechnen, so daß Potenzieren mit e direkt als Verschlüsselung benutzt werden kann. Bei der modularen „Exponentialfunktion“ $x \mapsto a^x \bmod p$ sind keine speziellen Wahlen von a und p bekannt, die vermöge einer geheimen Information zu einer einfachen Umkehrfunktion führen – diskrete Logarithmen sind für alle gleich schwer zu berechnen.

Die geheime Information bei einem asymmetrischen Verfahren auf der Basis diskreter Logarithmen kann daher nur in der Kenntnis *einzelner* diskreter Logarithmen bestehen: Wer für einen speziellen Wert x die

Potenz $y = a^x \bmod p$ berechnet hat, weiß anschließend, daß x der diskrete Logarithmus von y mod p zur Basis a ist.

Bei diesen sehr viel spezielleren „Geheimnissen“ ist klar, daß Kryptoverfahren auf der Basis von diskreten Logarithmen anders aussehen müssen als RSA.

a) Verschlüsselung nach ElGamal

Im Prinzip könnte man die Schlüsselvereinbarung nach DIFFIE und HELLMAN direkt zu einem Verschlüsselungsverfahren erweitern: Nachdem das gemeinsame Geheimnis $\gamma = a^{xy} \bmod p$ vereinbart ist, können Nachrichtenblöcke m_i mit $0 \le m_i < p-1$ in beide Richtungen verschlüsselt werden als $c_i = \gamma m_i \bmod p$. Da beide Partner den Wert von γ kennen, können sie leicht nach dem erweiterten EUKLIDischen Algorithmus ein δ berechnen, so daß $\gamma\delta \equiv 1 \bmod p$, und die verschlüsselte Information kann einfach entschlüsselt werden als $m_i = \delta c_i \bmod p$.

Solange nur ein einzelner Block m übertragen werden soll, ist dagegen nichts einzuwenden. Sobald aber mehrere Blöcke zu übertragen sind, wird dieses Verfahren verwundbar gegen Angriffe mit bekanntem Klartext: Falls ein Gegner für einen einzigen Chiffreblock c_i den Klartextblock m_i kennt (oder errät), kann er $\delta = m_i/c_i \bmod p$ berechnen und damit den gesamten Klartext entschlüsseln.

In dieser Form wäre das Verfahren daher höchstens dann sicher, wenn für jeden einzelnen Block ein eigenes γ vereinbart wird, wenn also für jeden Block das gesamte DIFFIE-HELLMAN-Protokoll durchlaufen wird. Das wäre allerdings sehr aufwendig.

Das Verfahren von ELGAMAL umgeht dies, indem es exakt dieselbe Mathematik mit einem leicht modifizierten Protokoll zu einem asymmetrischen Kryptoverfahren macht:

Die Parameter a und p sind entweder allgemein bekannte Systemparameter, oder jeder Teilnehmer A wählt sie selbst als Teil seines öffentlichen Schlüssels. Zusätzlich wählt er sich eine geheime Zufallszahl x und veröffentlicht $u = a^x \bmod p$.

Wer immer eine Nachricht $m_1, \ldots, m_r$ an A schicken möchte, erzeugt für jeden Block m_i eine Zufallszahl y_i berechnet daraus $v_i = a^{y_i} \bmod p$ und $c_i = u^{y_i} m_i$. Dann schickt er die Folge der Paare (v_i, c_i) an A. Der Chiffretext ist damit doppelt so lang wie der Klartext, was das Verfahren insbesondere für lange Texte nicht sonderlich attraktiv macht.

A muß zur Entschlüsselung den Multiplikator u^{y_i} kennen; dann kann er m_i als $c_i u^{-y_i}$ berechnen. Da

$$u^{y_i} \equiv a^{x y_i} \equiv \left(a^{y_i}\right)^x \equiv v_i^x \bmod p$$

ist, hat er damit keine Probleme.

Der offensichtliche Angriff eines Gegners besteht darin, aus u und a den diskreten Logarithmus x zu ermitteln, was nach derzeitigem Stand der Dinge schwierig erscheint. Ob andere Angriffe zum Erfolg führen könnten, ist (wie üblich) unbekannt – hoffentlich auch unseren Gegnern.

Genau wie bei RSA gibt es aber natürlich eine ganze Reihe von Möglichkeiten, das Verfahren durch schlechte Parameterwahl unsicher zu machen; einige davon sind in der am Ende von Kap. 4, §3*b)* zitierten Arbeit zu finden.

TAHER ELGAMAL wurde 1956 in Ägypten geboren. Er studierte zunächst Elektrotechnik an der Universität Kairo; nachdem er dort seinen BSc bekommen hatte, setzte er seine Studien fort an den Information Systems Laboratories der Stanford University. In seiner Masterarbeit ging es hauptsächlich um Systemtheorie, jedoch hörte er parallel auch freiwillig viele Mathematikvorlesungen und kam auf diesem Weg zur Kryptographie, die zum Thema seiner Doktorarbeit wurde. Nach dem Studium arbeitete er für eine ganze Reihe von Unternehmen, beispielsweise war er von 1995–1998 als Chefwissenschaftler von Netscape maßgeblich an der Entwicklung von SSL beteiligt. Zeitweise arbeitete er auch in selbst gegründeten Firmen. Seit Oktober 2006 ist er Chief Technology Officer der Tumbleweed Communications Corporation, außerdem ist er einer der Direktoren sowohl dieser als auch einiger weiterer Firmen.

b) Das Verfahren von Massey-Omura

Bei diesem Verfahren geht es, wie bei der Schlüsselvereinbarung nach DIFFIE-HELLMAN, um Nachrichtenaustausch zwischen zwei Partnern A und B, die über keinerlei gemeinsame Schlüsselinformation verfügen; es gibt auch keine öffentlichen Schlüssel.

Die beiden einigen sich auf eine Primzahl p (die auch Konstante eines ganzen Netzwerks sein kann), 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 B eine Nachricht m verschlüsselt an A schicken, so schickt er $c_1 = m^{e_B} \bmod p$. Damit kann natürlich weder A noch ein etwaiger Lauscher etwas anfangen: Da niemand außer B die beiden Exponenten e_B und d_B kennt, ist das einfach *irgendeine* Potenz zu *irgendeiner* Basis. Selbst ein BAYESscher Gegner, der alle Kombinationen (M, e) mit $M^e \equiv m^{e_B} \bmod p$ durchprobieren kann, wird dort für große p eine Fülle von potentiellen Klartexten finden, die alle ungefähr gleich wahrscheinlich sind.

A schickt die Nachricht daher gleich wieder zurück, potenziert sie aber vorher mit seinem Exponenten e_A. Was B erhält, ist also $c_2 = m^{e_B e_A}$, eine Nachricht die niemand entschlüsseln kann.

B potenziert diese Nachricht mit seinem Exponenten d_B; die liefert

$$\left(m^{e_B e_A}\right)^{d_B} = m^{e_B e_B d_B} = m^{e_B d_B e_A} = \left(m^{e_B d_B}\right)^{e_A} \equiv m^{e_A} \bmod p\,.$$

Diese Nachricht schickt er an A, der nun mit seinem Exponenten d_A leicht den Klartext ermitteln kann.

Auch die Sicherheit dieses Verfahrens hängt an diskreten Logarithmen: Ein etwaiger Lauscher kennt die Zahlen

$$m^{e_B} \bmod p, \quad m^{e_B e_A} = \left(m^{e_B}\right)^{e_A} \quad \text{und} \quad m^{e_A} = \left(m^{e_B e_A}\right)^{d_B};$$

falls er in der Lage ist, diskrete Logarithmen modulo p zu berechnen, kann er also e_A bestimmen als den diskreten Logarithmus von $m^{e_B e_A}$ zur Basis m^{e_B} und d_B als diskreten Logarithmus von $\left(m^{e_B e_A}\right)^{d_B}$ zur Basis $m^{e_B e_A}$. Damit kann auch er m berechnen, indem er beispielsweise m^{e_A} modulo p mit d_B potenziert. Die Primzahl p muß also auch bei diesem Verfahren so groß sein, daß die Berechnung diskreter Logarithmen modulo p zumindest praktisch undurchführbar ist.

JAMES L. MASSEY wurde 1934 in Wauseon, Ohio geboren. Er studierte Elektrotechnik an der University of Notre Dame und am MIT, wo er sich vor allem auf Informations- und Kodierungstheorie konzentrierte. Nach dem Studium war er 14 Jahre lang Professor in Notre Dame, dann kurz am MIT und an der UCLA, bis er 180 einem Ruf an die ETH Zürich folgte, wo er bis zu seiner Emeritierung zum 1. April 1999 einen Lehrstuhl für Signal- und Informationsverarbeitung hatte.

JIM K. OMURA studierte in Stanford Elektrotechnik und war dann 15 Jahre lang Professor an der UCLA. Danach gründete er eine eigene Firma namens Cylink (inzwischen von Safenet übernommen) und arbeitete als Berater für verschiedene Firmen und Stiftungen.

c) DSA

Der Zusatzaufwand gegenüber RSA macht sowohl ELGAMAL aus auch MASSEY-OMURA für praktische Anwendungen eher uninteressant; hinzu kommt, daß weder bei ELGAMAL noch bei MASSEY-OMURA auch nur einer der beiden Partner sicher sein kann, daß er wirklich mit dem anderen kommuniziert. Dasselbe gilt natürlich zumindest für den Empfänger auch bei RSA. Dort allerdings liefert das Verfahren selbst eine Möglichkeit für elektronische Unterschriften, die dieses Problem löst. Auch das Verfahren von ELGAMAL kann so modifiziert werden, daß es elektronische Unterschriften realisiert, mit diskreten Logarithmen kann man aber auch noch weiter gehen: Wegen des großen Aufwands asymmetrischer Kryptoverfahren und der oftmals gewaltigen Länge zu unterzeichnender Texte wird meist nicht die gesamte Nachricht unterzeichnet, sondern nur ein Hashwert derselben.

Ein solcher Hashwert hat heute oft noch nur 160 Bit; sicherheitsbewußte Anwender wählen Verfahren, deren Resultat 224 oder besser noch 256 Bit liefern.

Wenn wir so einen Hashwert mit RSA unterzeichnen, hat die Unterschrift eine Länge von (je nach Sicherheitsstandard) 1024 bis 2048 Bit. Verglichen mit der Länge des zu unterzeichnenden Hashwerts ist das offensichtlich übertrieben. Andererseits wäre eine Unterschrift, die auf diskreten Logarithmen in einem Körper mir nur etwa 2^{256} Elementen beruht ohne großen aufwand fälschbar.

Der Ausweg aus diesem Dilemma besteht darin, daß man zwar in einer großen Gruppe rechnet, die Unterschriften aber in einer deutlich klei-

neren Untergruppe liegen. Praktisch realisiert ist dies bei Unterschriften nach 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.

Als Ordnung der Untergruppe wählt man eine Primzahl q, für die nach den derzeitigen Empfehlungen der Bundesnetzagentur bis Ende 2009 eine Länge von 160 Bit ausreicht; 224 Bit reichen auch bis Ende 2012. Die Gruppenordnung sollte natürlich an die Größe des zu unterzeichnenden Hashwerts angepaßt sein.

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 2009 mindestens 1536, bis Ende 2012 mindestens 2048. „Empfohlen“ sind auch hier schon jetzt 2048 Bit.

Primzahlen $p \equiv 1 \bmod q$ sind nicht schwerer zu finden als beliebige Primzahlen: Falls man bei der Primzahlsuche wirklich auf Nummer sicher geht und Zufallszahlen auf Primalität testet, nimmt man hier einfach Zufallszahlen k und testet $kq + 1$ auf Primalität. Falls man mit ERATOSTENES arbeitet, kann man das Sieben leicht so modifizieren, daß nur Zahlen der Form $kq + 1$ gesiebt werden. An den Erfolgschancen ändert dies in beiden Fällen nichts: Nach einem Satz von DIRICHLET über Primzahlen in arithmetischen Folgen ist die Dichte der Primzahlen der Form $kq + i$ für jedes i mit $0 < i < q$ dieselbe; in der Größenordnung n ist also weiterhin im Mittel jede $\ln n$-te solche Zahl eine Primzahl. (Tatsächlich sind es sogar geringfügig mehr, denn außer q selbst gibt es natürlich keine Primzahl der Form $p = kq$. Bei den Größenordnungen von q mit denen wir arbeiten, geht aber der Unterschied zwischen q und $q - 1$ definitiv im „Rauschen“ der im Kleinen sehr unregelmäßigen Primzahlverteilung unter.)

Als nächstes muß ein Element g gefunden werden, dessen Potenzen im Körper $\mathbb{F}_p$ eine Gruppe der Ordnung q bilden. Auch 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 $u = g^x \bmod p$.

Unterschreiben lassen sich mit diesem Verfahren Nachrichtenblöcke m mit $0 \le m < q$; im allgemeinen wird es sich dabei um Hashwerte der eigentlich zu unterschreibenden Nachricht handeln. 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 modulo q 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. Eine solche Nachricht kann nur erzeugt 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,

$$g^k \equiv g^{tm} g^{xtr} \equiv g^{tm} u^{tr} \bmod p\,.$$

Modulo q ist die linke Seite gleich g^{tm} kann wie auch u^{tr} aus öffentlicher Information und der Unterschrift berechnet werden, von der linken Seite wissen wir, daß sie modulo q gleich dem ersten Teil r der Unterschrift. Die Unterschrift wird daher anerkannt, wenn

$$r \equiv (g^{tm} u^{tr} \bmod p) \bmod q$$

ist. (Die beiden Potenzen und ihr Produkt müssen natürlich zunächst modulo p berechnet werden: Zwei modulo p kongruente Zahlen sind im allgemeinen nicht kongruent modulo q.)

Ein Angreifer müßte sich x aus u verschaffen, müßte also ein diskretes Logarithmenproblem modulo der großen Primzahl p lösen.

§3: Diskrete Logarithmen in anderen Gruppen

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:

a) Die abstrakte Situation

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 $\varphi_g^{-1}\colon \text{Bild}\,\varphi_g \to \mathbb{Z}/\,\text{Kern}\,\varphi_g$ 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.)

b) Der chinesische Restesatz im abstrakten Fall

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

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

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 mod 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.

c) 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

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

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

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

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

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

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

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

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.

d) Folgerungen für die Sicherheit von Kryptosystemen

Die Diskussion in den beiden vorigen Abschnitten zeigt, daß die Schwierigkeit der Berechnung einen diskreten Logarithmus in einer zklischen Gruppe der Ordnung n relativ einfach zurückgeführt werden kann auf die Berechnung diskreter Logarithmen in Untergruppen, deren Ordnungen die Primteiler von n sind. Als Fausregel können wir daher festhalten, daß die Sicherheit diskreter Logarithmen in einer zyklischen Gruppe der Ordnung n im wesentlichen nur gleich der Sicherheit in einer Untergruppe der Ordnung q ist, wobei q der größten Primteiler von n ist.

Idealerweise sollte daher die Gruppenordnung n selbst eine Primzahl sein, was allerdings zumindest für die multiplikative Gruppe modulo p nur im kryptographisch völlig uninteressanten Fall $p = 3$ der Fall ist.

Hier empfiehlt sich, eine Primzahl p der Form $2q + 1$ zu wählen, wobei auch q eine Primzahl ist. Die multiplikative Gruppe modulo p hat dann die Ordnung $2q$, und die Sicherheit entspricht der in einer Gruppe der Ordnung q.

In diesem Fall ist es auch sehr einfach, erzeugende Elemente zu finden: Da $\mathbb{F}_p$ ein Körper ist, haben nur die beiden Elemente ± 1 die Ordnung zwei, alle anderen haben entweder die Ordnung q oder die Ordnung $2q$. Da es zwischen den beiden Fällen keinen nennenswerten Unterschied in der Sicherheit gibt, können wir also jedes Element außer ± 1 als Basis nehmen. (Natürlich ist es auch kein Problem, zwischen Ordnung q und Ordnung $2q$ zu unterscheiden: Falls $a^q = 1$ für ein $a \not\equiv \pm 1$, hat a die Ordnung gleich q, ansonsten $2q$.)

e) Multiplikative Gruppen beliebiger endlicher Körper

Tatsächlich gibt es nicht nur für jede Primzahl p einen Körper mit p Elementen, sondern für jede Primzahl*potenz* $q = p^n$. Für den Fall $q = 2^8 = 256$ werden wir dies im nächsten Kapitel genauer betrachten.

Die multiplikative Gruppe des Körpers mit p Elementen ist einfach die Gruppe $\{1, \ldots, p-1\}$ mit der Multiplikation modulo p als Verknüpfung; statt in dieser Gruppe können wir genauso gut in der multiplikativen Gruppe eines Körpers von Primzahlpotenzordnung diskrete Logarithmen betrachten.

Man kann zeigen, daß diese Gruppe stets zyklisch ist; siehe dazu beispielsweise das Zahlentheorieskriptum des vergangenen Semesters, Kap I, §7; falls ihre Ordnung $q - 1 = p^n - 1$ prim ist oder einen großen Primteiler hat, lassen sich auch so sichere Kryptosysteme aufbauen.

In der Praxis kryptographischen Parxis spielen Potenzen ungerader Primzahlen allerdings kaum eine Rolle: Das Rechnen in den entsprechenden Körpern ist aufwendiger als das in einem vergleichbar großen Körper von Primzahlordnung, ohne daß dies zu einem Sicherheitsgewinn führen würde.

Anders steht es mit Körpern von Zweipotenzordnung: Da Computer ohnehin im Zweiersystem rechnen, können sie damit recht effizient umgehen. Es gibt sogar eine Reihe von Exponenten, für die $2^n - 1$ prim ist; für $n \leq 2\,500$ sind dies $2, 3, 5, 7, 13, 17, 19, 31, 61, 89, 107, 127, 521, 607, 1279, 2203$ und 2281.

Grundsätzlich dürfte wohl die Sicherheit diskreter Logarithmen in einem Körper mit 2^n Elementen vergleichbar sein mit der eines Körpers, dessen Elementanzahl eine Primzahl derselben Größenordnung ist, allerdings gibt es sehr viel weniger Zweierpotenzen als Primzahlen, und nicht für alle n hat $2^n - 1$ einen großen Primteiler. Dadurch besteht die Gefahr, daß sich kriminelle Energie (sowie auch Nachrichtendienste) auf die wenigen interessanten Werte von n stürzen und dort mit Spezialhardware arbeiten, was die Sicherheitssituation zu ihren Gunsten verschiebt. Somit dürften im allgemeinen Primzahlen vozuziehen sein; hinzu kommt, daß zumindest DSA ohnehin nur für diesen Fall definiert ist.

f) Elliptische Kurven

In der Praxis ist für Kryptoverfahren auf der Basis diskreter Logarithmen abgesehen von den multiplikativen Gruppe der Körper von Primzahlpotenzordnung vor allem noch eine andere Art von Gruppe wichtig:

die Gruppe der rationalen Punkte einer elliptischen Kurve über einem endlichen Körper. In bislang eher noch experimentellen Systemen arbeitet man auch mit höherdimensionalen Verallgemeinerungen davon, hauptsächlich den JACOBIschen hyperelliptischer Kurven. Da elliptische und hyperelliptische Kurven Gegenstand einer eigenen Vorlesung im nächsten Semester sind, soll hier nur kurz erläutert werden, um welche Art von Gruppe es sich bei einer elliptischen Kurven handelt.

Elliptische Kurven sind keine Ellipsen; sie haben ihren Namen davon, daß bei der Berechnung der Bogenlänge einer Ellipse algebraische Integrale auftreten, deren Integranden solche Kurven definieren.

Eine elliptische Kurve über einem Körper k ist eine ebene Kurve vom Grad drei; sie ist also gegeben durch ein Polynom f vom Grad drei mit Koeffizienten aus k in zwei Variablen x und y. Wir verlangen zusätzlich, daß sich das Polynom f auch über Erweiterungskörpern von k nicht als Produkt eines linearen und eines quadratischen Polynoms schreiben läßt, daß es mindestens eine Nullstelle $(x, y) \in k^2$ hat und daß die partiellen Ableitungen f_x und f_y für keine Lösung (x, y) der Gleichung $f(x, y) = 0$ über irgendeinem Erweiterungskörper von k simultan verschwinden.

Geometrisch bedeuten diese Forderungen, daß die durch $f(x, y) = 0$ definierte Kurve nicht als Vereinigung einer Geraden und einer anderen Kurve geschrieben weden kann, daß sie mindestens einen Punkt mit Koordinaten aus k hat und daß es in jedem Punkt eine wohldefinierte Tangente gibt.

Schneiden wir eine solche Kurve mit einer Geraden, so erlaubt uns die Geradengleichung die Elimination einer der beiden Variablen x und y; was übrigbleibt ist eine höchstens kubische Gleichung in einer Variablen. Damit ist klar, daß eine Gerade eine elliptische Kurve in höchstens drei Punkten schneidet.

Betrachtet man die Kurve nicht in der Ebenen k^2 sondern in der projektiven Ebenen über k, so kann man leicht zeigen, daß es mit Vielfachheiten gezählt genau drei Schnittpunkte gibt.

Damit könnte man versuchen, eine Verknüpfung zu definieren, indem je zwei Punkten der dritte Schnittpunkt ihrer Verbindungsgeraden mit der

Kurve zugeordnet wird. Diese Verknüpfung erfüllt aber leider abgesehen vom Kommutativgesetz keines der Gruppenaxiome.

Eine leichte Modifikation führt aber zu einer Gruppenstruktur: Drei Punkte P, Q, R liegen genau dann auf einer Geraden, wenn $P + Q + R$ gleich dem neutralen Element O ist. Da in einer Gruppe $O + O + O = O$ sein muß, impliziert dies insbesondere, daß die Tangente durch den Punkt O die Kurve dort mit Vielfachheit drei schneidet, d.h. O ist ein Wendepunkt. Er wird meist ins „Unendliche“ gelegt, ist also im (x, y)-Koordinatensystem kein Punkt von k^2. Auch die Tangente dort wird als „unendlichferne Gerade“ im Sinne der projektiven Geometrie gewählt; dann ist O der einzige Kurvenpunkt, der nicht in k^2 liegt, so daß man für die konkrete Durchführung der Gruppenoperation mit Rechnungen in k^2 auskommt mit wenigen und einfachen Zusatzregeln für den Fall, daß O als Summand oder Ergebnis auftritt.

Der wesentliche Grund für die Verwendung elliptisher Kurven liegt darin, daß die effizienteste der bekannten Strategien zur Berechnung diskreter Logarithmen nur für die multiplikative Gruppe eines endlichen Körpers verwendet werden kann; in allgemeinen Gruppen funktionieren nur erheblich langsamere Verfahren.

Tatsächlich gibt es natürlich auch spezielle Methoden für elliptische Kurven; in machen Fällen kann man die Berechnung diskreter Logarithmen in einer elliptischen Kurve über einem endlichen Körper zurückführen auf die Berechnung diskreter Logarithmen in der multiplikativen Gruppe eines nicht sehr viel größeren Körpers. Wer mit den theoretischen Grundlagen der Kryptographie mit elliptischen Kurven vertraut ist, kann zumindest die bekannten Angriffsmethoden so erschweren, daß der entsprechende Erweiterungskörper schon für relativ kleine Grundkörper so groß ist, daß dort nach heutigem Kenntnisstand auch klassische diskrete Logarithmenprobleme sicher sind. Die Empfehlungen der Bundesnetzagentur sehen hier daher für die Kryptographie mit elliptischen Kurven deutlich kleinere Primzahlen vor als im klassischen Fall. Wie dort gibt es zwei Primzahlen p und q: Die elliptische Kurve ist definiert über einem Körper mit p Elementen, und die Unterschrift liegt in einer Untergruppe der Ordnung q der elliptischen Kurve. Bis Ende 2009 muß q eine Länge von mindestens 180 Bit haben, also etwas mehr als beim klassischen

DSA, dafür reichen für p aber 192 Bit. Bis Ende 2012 muß q, genau wie beim klassischen DSA, mindestens 224 Bit haben, und an p gibt es keine Bedingung außer daß $p \neq q$ sein muß. Zur Erschwerung der oben erwähnten Angriffsmethode wird außerdem stets gefordert, daß es kein $r \leq 10^4$ geben darf, so daß q ein Teiler von $p^r - 1$ ist; die Gruppe, in der die Unterschriften liegen, soll sich also nicht in die multiplikative Gruppe einer „kleinen“ Erweiterung des Grundkörpers einbetten lassen.

§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. Obwohl es keinen klaren theoretischen Zusammenhang zwischen den beiden Problemen gibt, zeigt die Erfahrung der letzten Jahren eine erstaunliche Parallelität zwischen den entsprechenden Algorithmen.

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 das 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 (was man bei kryptographischen Anwendungen immer anstreben sollte), 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

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

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.

d) Indexkalkül

Die derzeit besten Faktorisierungsalgorithmen beruhen auf dem quadratischen und dem Zahlkörpersieb; für beide wurden bald nach ihrer Einführung ähnliche Siebalgorithmen gefunden, die zur Berechnung diskreter Logarithmen führen. Wir beschränken uns hier, wie auch schon bei der Faktorisierung, auf das quadratische Sieb, dessen Variante für diskrete Logarithmen als *Indexkalül* bezeichnet wird, und auch hier beschränken wir uns auf eine einfache Variante speziell für Primkörper $\mathbb{F}_p$.

Wie beim quadratischen Sieb wird eine Schranke B festgelegt und damit eine Faktorbasis $\mathcal{B}$ definiert; diese besteht hier aus *allen* Primzahlen $q \le B$. Der Algorithmus besteht aus zwei Teilen:

Im ersten Teil berechnet man die diskreten Logarithmen aller Primzahlen q aus der Faktorbasis zur gegebenen Basis a modulo p. Dies mag auf den ersten Blick unsinnig erscheinen, denn schließlich suchen wir den diskreten Logarithmus *einer* Zahl und beginnen dazu mit der Berechnung der diskreten Logarithmen vieler Zahlen. Die Logarithmen der Primzahlen lassen sich aber simultan wie folgt berechnen: Man berechne viele Potenzen $a^y \bmod p$ und suche diejenigen, die eine Primfaktorzerlegung mit lauter Faktoren aus $\mathcal{B}$ haben. Ist

$$a^y \bmod p = q_1^{e_1} \cdots q_r^{e_r} ,$$

so ist

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

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.

Leider gibt es kein Siebverfahren, mit dem sich feststellen läßt, welche Werte $a^y \bmod p$ durch eine gegebene Primzahl q teilbar sind; hier muß man also explizit faktorisieren –zumindest so lange, bis alle Faktoren aus $\mathcal{B}$ gefunden sind. Egal ob man hier mit Probedivisionen arbeitet oder etwa mit POLLARDs ρ-Methode oder mit elliptischen Kurven: Das Verfahren ist für große p deutlich schneller als beispielsweise *baby step – giant step*, und der Aufwand steigt langsamer als exponentiell in der Zifferzahl von p.

§5: 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

Als ersten Überblick über elliptische Kurven kann man etwa das Buch

ANNETTE WERNER: Elliptische Kurven in der Kryptographie, *Springer,* 2002,

konsultieren, das eine elementare Einführung in die Theorie elliptischer Kurven unter dem Gesichtspunkt der Kryptographie geht. Beweise sind nicht immer vollständig, und anspruchsvollere Algorithmen werden nur sehr kurz oder gar nicht behandelt.

Deutlich anspruchsvoller und vollständiger sind

DARREL HANKERSON, ALFRED MENEZES, SCOTT VANSTONE: Guide to Elliptic Curve Cryptography, *Springer,* 2004

LAWRENCE C. WASHINGTON: Elliptic Curves – Number Theory and Cryptography, *Chapman & Hall/CRC,* 2003

IAN BLAKE, GADIEL SEROUSSI, NIGEL SMART: Elliptic Curves in Cryptography, *London Mathematical Society Lecture Notes Series* **265**, *Cambridge University Press,* 1999

IAN BLAKE, GADIEL SEROUSSI, NIGEL SMART [HRSG.:] Advances in Elliptic Curve Cryptography, *London Mathematical Society Lecture Notes Series* **317**, *Cambridge University Press,* 2005

und vor allem

HENRI COHEN, GERHARD FREY, ROBERTO AVANZI, CHRISTOPHE DOCHE, TANJA LANGE, KIM NGUYENM FREDERIK VERCAUTEREN: Handbook of Elliptic and Hyperelliptic Curve Cryptography, *Chapman & Hall/CRC,* 2006

Speziell mit Implementierungsfragen beschäftigt sich

MICHAEL ROSING: Implementing Elliptic Curve Cryptography, *Manning,* 1999