Erschien in *Elemente der Mathematik*, Birkhäuser, Vol. 50, No. 3, pp. 89-106, 1995.

Kryptologie: eine neuartige Anwendung der Mathematik

Ueli M. Maurer

Institut für Theoretische Informatik
ETH Zürich
CH-8092 Zürich
Email address: maurer@inf.ethz.ch

Abstract. Die Kryptologie kann als die Wissenschaft der mathematischen und algorithmischen Aspekte der Informationssicherheit definiert werden. Sie befasst sich mit Protokollen für vielfältige Anwendungen wie z.B. Datenchiffrierung, digitale Unterschriften, Benutzeridentifikation in Computersystemen und digitales Geld, und verwendet verschiedene mathematische Theorien wie Algebra, Zahlentheorie, Kombinatorik, Wahrscheinlichkeitstheorie, usw. Einige interessante Themen der Kryptologie und überraschende Anwendungen der Mathematik werden diskutiert. Sie eignen sich zum Teil gut für den Mathematikunterricht auf Gymnasialstufe.

1. Einleitung

1.1. Bemerkungen zur Mathematik

Die Wissenschaft Mathematik erfüllt zwei wichtige Funktionen in unserer Gesellschaft. Erstens bildet die Mathematik eine Basis der Ingenieurdisziplinen und Naturwissenschaften, und die modernen Errungenschaften der Technik wären ohne Mathematik nicht denkbar. Zweitens muss die Mathematik als Teil unserer Kultur betrachtet werden, die sich ebenso wie die Kunst an den Ansprüchen der Ästhetik orientiert.

Leider ist vielen Personen, die sich mit Mathematik in Forschung oder Anwendung beschäftigen, nur einer dieser zwei Aspekte wirklich zugänglich. "Reine" Mathematiker erfreuen sich an der Schönheit einer Theorie, die aber dem Praktiker oft verborgen bleibt, weil er sie nicht verstehen kann. Umgekehrt wird die Mathematik in sehr vielen Disziplinen nutzbringend angewendet, ohne dass die dabei auftretenden Probleme für den wahren Theoretiker genügend ästhetisch sind, um von grossem Interesse zu sein.

Die Kryptologie ist wahrscheinlich eines der wenigen Anwendungsgebiete der Mathematik, bei dem beide oben erwähnten Aspekte gleichermassen vertreten sind. Sie greift

auf verschiedenartige und unerwartete Weise auf mathematische Theorien zu, von denen zuvor nie eine Anwendung erwartete wurde. Zum Beispiel galt die Zahlentheorie lange als reinste Disziplin innerhalb der Mathematik, rein unter anderem in dem Sinn, dass sie am weitesten entfernt von Anwendungen sei. Zudem eignet sich die Kryptologie aus zwei Gründen sehr gut als Illustrationswerkzeug für verschiedene Gebiete der Mathematik. Erstens erscheinen viele kryptologische Anwendungen verblüffend oder sogar paradox und können einen Leser oder Zuhörer nur schon deshalb faszinieren. Wie soll es z.B. möglich sein, einem Computersystem zu beweisen, dass man ein Passwort kennt, ohne jegliche Information über das Passwort wegzugeben und ohne dass das Passwort im Computer gespeichert ist ? Zweitens sind einige der in der Kryptologie benutzten mathematischen Gesetze für den interessierten Laien durchaus verständlich.

Dieser Artikel soll einen Beitrag leisten zum Brückenschlag zwischen Anwendung und Ästhetik der Mathematik. Das Niveau ist absichtlich so gewählt, dass es für einen interessierten Laien grösstenteils verständlichen sein sollte, obwohl natürlich auch kompliziertere Theorien ihre Anwendung in der Kryptologie finden. An verschiedenen Stellen werden Hinweise auf mathematische Theorien gegeben, die in der Praxis angewendet werden, oder die sogar aus der Praxis entstanden sind.

1.2. Information und Informationssicherheit

Wir befinden und im Informationszeitalter. Man ist sich heute einig, dass Information die wichtigste Ressource der modernen Wirtschaft ist. Information zu definieren ist allerdings nicht einfach. Generell kann man Information als Wissensgewinn definieren, wobei also der Neuheitsaspekt eine wichtige Rolle spielt, aber der Informationsgehalt einer bestimmten Nachricht natürlich stark vom empfangenden System, vom Kontext und der Zeit abhängt. Shannon (übrigens ein Ingenieur), publizierte 1948 die Informationstheorie [31] (siehe z.B. [6]), die basierend auf der Wahrscheinlichkeitstheorie eine präzise Definition von Information gibt und diese rechtfertigt, indem sie für viele konkrete Probleme der Praxis wie die Datenkompression oder die fehlerfreie Übertragung von Information über fehlerbehaftete Kommunikationskanäle die optimale Lösung liefert.

Information kann viele Formen annehmen: Personendaten, Unternehmensdaten, Software, militärische und strategische Information, usw. Der Wert von Information ist bestimmt durch deren örtliche und zeitliche Verfügbarkeit, weshalb Datenkommunikation und -speicherung zentrale Anforderung an Informationssysteme sind. Eine weitere zentrale Anforderung ist die Informationssicherheit, was unter anderem die Vertraulichkeit, Authentizität und Integrität von sensitiver Information einschliesst.

Sicherheit ist in modernen Informations- und Computersystemen durch verschiedene bekanntgewordene Fälle von Hackerangriffen zu einer brisanten Problematik geworden. Die Wichtigkeit und Aktualität der Sicherheitsproblematik hat mehrere Gründe und dürfte in den nächsten Jahren noch stark zunehmen. Einerseits haben die fortschreitende Vernetzung von Computersystemen und die steigende Abhängigkeit der Unternehmen von Informatikmitteln zu einer starken Erhöhung des Gefahren- und Schadenpotentials geführt. Andererseits werden durch die enorm zunehmenden Datenmengen die Anforderungen an den Persönlichkeitsschutz immer wichtiger (neues Datenschutzgesetz der Schweiz). Sicher-

heit könnte zu einem Schlüsselfaktor in der zukünftigen Entwicklung von global vernetzten Informations- und Computersystemen werden.

1.3. Kryptologie

Es gibt vielfältige Gründe für das Fehlverhalten eines Informationssystems: Einflüsse der Umgebung (Stromausfall, Brand, usw.), Versagen der Hardware, fehlerhafte Software, Bedienungsfehler sowie absichtliche Eingriffe von Kriminellen oder Hackern. Die Kryptologie ist eine moderne Wissenschaft, die sich mit der letzten Kategorie von absichtlichen und intelligenten Bedrohungen befasst. Sie besteht aus zwei Teilgebieten: die *Kryptographie* befasst sich mit dem Entwurf von Systemen und die *Kryptoanalyse* mit dem Brechen von Systemen.

Der Begriff Kryptologie wird von Laien oft mit militärischen Anwendungen in Verbindung gebracht. Tatsächlich spielte die Kryptologie, oder präziser die Kryptoanalyse, eine wichtige wenn nicht kriegsmitentscheidende Rolle in beiden Weltkriegen. Heute überwiegen aber die vielfältigen zivilen Anwendungen bei weitem. An der Geschichte der Kryptologie interessierte Leser seien auf das umfassende Buch von Kahn [14] verwiesen. Zur eigentlichen Wissenschaft wurde die Kryptologie erst 1976, als Whitfield Diffie und Martin Hellman von der Stanford University eine bahnbrechende Erfindung machten. Ihr Protokoll (siehe Abschnitt 3.3) erlaubt es zwei kommunizierenden Partnern, die keinen geheimen Schlüssel besitzen und lediglich über eine vom Gegner abgehörte Leitung verbunden sind, Information geheim zu übertragen, ohne dass der Gegner sie verstehen kann. Diese Art der Verschlüsselung, welche paradox erscheint, nennt man Public-Key Kryptographie, weil sozusagen der Schlüssel zur Verschlüsselung öffentlich ist. Die ETH Zürich verlieh übrigens Diffie 1992 den Ehrendoktortitel.

Die Kryptologie befasst sich aber nicht nur mit der Datenverschlüsselung. Ebenso wichtige wenn zum Teil nicht wichtigere Themen sind Datenauthentifikation, Benutzeridentifikation in Computersystemen, digitale Unterschriften, Sicherheit in grossen Computernetzen, digitales Geld, usw. Einige Bücher zum Thema Kryptolgie sind, in absteigendem Empfehlungsgrad: [29], [32], [16], [33], [34], und [17]. Bauer [2] behandelt die klassische Kryptologie, die allerdings heute in der Praxis keine wichtige Rolle mehr spielt.

In Abschnitt 2 wird ein kurzer Überblick über die konventionelle Kryptologie gegeben und die Wichtigkeit der Public-Key Kryptographie motiviert, welche in Abschnitt 3 behandelt wird. Abschnitt 4 behandelt sogenannte "Zero-knowledge"-Beweisprotokolle und in Abschnitt 5 werden einige weitere Kuriositäten aus der Kryptographie kurz vorgestellt.

2. Konventionelle Kryptologie

Die zwei fundamentalen Sicherheitsanforderungen an eine Informationsübertragung oder -speicherung sind *Geheimhaltung* und *Authentizität* der Information. Geheimhaltung bedeutet, dass die Information keiner nicht legitimierten Person in verständlicher Form zugänglich wird, und Authentizität bedeutet, dass die vorhandene Information echt ist,

d.h. von der Person stammt, unter deren Namen sie abgepeichert ist. Dies schliesst auch Integrität ein, das heisst die Gewissheit, dass Information nicht auf unbefugte Art verändert wurde.

Diese zwei Ziele können mittels konventioneller Verschlüsselung mit einem geheimen, nur dem Sender und Empfänger bekannten Schlüssel erreicht werden. Ein konventionelles Kryptosystem für einen Klartextraum $\mathcal{X}$, einen Chiffratraum $\mathcal{Y}$ und einen Schlüsselraum $\mathcal{Z}$ besteht also aus einer Verschlüsselungstransformation $\mathcal{X} \times \mathcal{Z} \to \mathcal{Y}$ und der dazu (für fixen Schlüssel) inversen Entschlüsselungstransformation $\mathcal{Y} \times \mathcal{Z} \to \mathcal{X}$. Wer den Schlüssel kennt kann sowohl ver- als auch entschlüsseln, aber ohne Kenntnis des Schlüssels ist dies nicht möglich. Der Schlüssel kann z.B. mit einem Kurier an beide Benutzer verteilt werden und kann anschliessend für eine gewisse Zeit (z.B. ein Jahr) verwendet werden.

In diesem Modell der Kryptographie geht man davon aus, dass ein Gegner alle Teile des Systems (inklusive Transformationen) kennt, dass er aber keine *a priori* Information über den geheimen Schlüssel besitzt. Die gesamte Sicherheit beruht also auf dem geheimen Schlüssel. Man kann zwei Arten von Sicherheit unterscheiden. Ein System ist *berechenmässig* sicher, wenn es berechenmässig zu aufwendig ist, es zu brechen (wobei wir hier bewusst einer Definition für "Brechen" aus dem Weg gehen). Falls der Schlüssel eines Systems genügend gross gewählt wird, so würde das Durchprobieren aller Schlüssel (z.B. 2^{128} im Fall eines 128-Bit Schlüssels) selbst mit den schnellsten Computern Milliarden von Jahren dauern. Trotzdem gibt es bis heute kein praktisches System, für welches die berechenmässige Sicherheit bewiesen werden konnte. Mit andern Worten, es ist für jedes System denkbar, dass es durch irgendeinen Trick effizient gebrochen werden kann.

Obwohl es scheint, dass jedes System theoretisch mittels Durchprobieren aller Schlüssel gebrochen werden kann, ist es möglich, *informationstheoretisch* sichere Systeme zu entwerfen, die selbst ein Gegner mit unendlichen Computerressourcen nicht brechen kann. Ein solches System ist der sogenannte "One-Time Pad", bei dem die binäre Klartextfolge durch bitweise Addition modulo 2 einer echt zufälligen Schlüsselbitfolge verschlüsselt wird. Klartext und Chiffrat sind in diesem System statistisch unabhängig, aber natürlich ist es völlig unpraktisch, weil der Schlüssel gleich gross sein muss wie der zu übertragende Klartext. Wie der Name besagt, kann ein Schlüssel nur einmal verwendet werden. Ein neuartiges System [21] erreicht als erstes beweisbare informationstheoretische Sicherheit mit nur einem kurzen Schlüssel, was zuvor als unmöglich galt. Der Entwurf von informationstheoretisch sicheren Chiffriersystemen ist ein hochaktuelles Forschungsgebiet, welches auch die durch einen Artikel im Scientific American [4] bekannt gewordene Quantenkryptographie [3] umfasst. Aus Platzgründen wird informations-theoretische Sicherheit hier nicht weiter behandelt.

Ein weitverbreitetes konventionelles Kryptosystem ist der Data Encryption Standard (DES). Dieses sogenannte Blockcipher-System verschlüsselt den Klartext blockweise (64-Bit Blöcke) mit Hilfe eines 56-Bit Schlüssels. DES kann also als Menge von 2^{56} Permutationen der Menge $\{0,1\}^{64}$ aufgefasst werden. Neben den Blockciphern sind die sogenannten Streamcipher eine in der Praxis wichtige Chiffriermethode. Sie simulieren den One-Time Pad indem die Schlüsselbitfolge pseudo-zufällig, abhängig vom geheimen Schlüssel, statt echt zufällig erzeugt wird, wodurch natürlich der Sicherheitsbeweis für den One-Time Pad hinfällig wird. Eine in der Praxis oft verwendete Struktur zur Erzeugung von Folgen mit

hoher Periode und guten statistische Eigenschaften sind linear rückgekoppelte Schieberegister [33], die theoretisch interessant sind und deren Zustandsübergang als Multiplikation in einem endlichen Körper interpretiert werden kann.

Ein scheinbar unlösbares Problem bei der Verwendung konventioneller Kryptosysteme in grossen Netzwerken (in denen sich die kommunizierenden Benutzer resp. Applikationen unter Umständen nicht einmal kennen), stellt aber die Verteilung der geheimen Schlüssel dar: In einem Netzwerk mit N Benutzern gibt es $N(N-1)/2$ verschiedene Benutzer-Paare, die alle einen verschiedenen geheimen Schlüssel haben müssen.

Ein weiteres Problem, das mit konventionellen Kryptosystemen nicht gelöst werden kann, ist die Erzeugung digitaler Unterschriften, also das Anbringen eines fälschungssicheren aber trotzdem universell verifizierbaren digitalen Musters an eine digitale Nachricht (z.B. einen Vertrag, dessen Unterschrift von einem Richter geprüft werden kann). Solche Anwendungen sind z.B. in der aufkommenden EDI-Technologie von grosser Bedeutung. Diese und andere Probleme können mit Hilfe der Public-Key Kryptographie gelöst werden.

3. Public-Key Kryptographie und digitale Unterschriften

Es scheint selbst für einen Laien möglich zu sein, ein relativ sicheres konventionelles Kryptosystem zu entwerfen. Ein möglicher Ansatz wäre z.B., einen Pseudo-Zufallsgenerator für einen Streamcipher zu entwerfen, der aus einem Anfangszustand (dem geheimen Schlüssel) durch Verwendung vieler komplizierter Transformationen eine Bitfolge erzeugt, die berechenmässig unmöglich zu analysieren ist. (Diese Bemerkung soll nicht dahingehend missverstanden werden, dass der Autor eine solche ad-hoc Entwurfstrategie für die Praxis empfiehlt.)

Im Gegensatz dazu ist der Entwurf von Public-Key Kryptosystemen eine Kunst, die nur durch Verwendung spezieller mathematischer Strukturen zum Ziel zu führen scheint. Die Anzahl vorgeschlagener Public-Key Systeme ist auch dementsprechend klein, und vor der Entdeckung des ersten solchen Systems war es völlig unklar, ob sie überhaupt existieren können.

Ein Public-Key Kryptosystem ist ein asymmetrisches Verschlüsselungssystem. Ein Benutzer, der geheime Nachrichten empfangen möchte, kann ein Paar von Schlüsseln bestehend aus "Public Key" und "Secret Key" generieren und den Public Key (wie der Name besagt) veröffentlichen. Jedermann kann mit Hilfe des Public Key Nachrichten für diesen Benutzer verschlüsseln, aber nur dieser kann mit Hilfe des geheimen Schlüssels die Nachrichten entschlüsseln. Ein mechanisches Analogon eine Public-Key Kryptosystems ist ein Vorhängeschloss, das man einfach schliessen kann, aber nur mittels des Schlüssels wieder öffnen kann.

Ein solches System scheint das in Abschnitt 2 erwähnte Schlüsselverteilproblem in grossen Netzwerken zu lösen, da die Kommunikationspartner nicht mehr paarweise geheime Schlüssel kennen müssen. Stattdessen können sich zwei Benutzer (oder Computerapplikationen) gegenseitig ihre Public Keys über einen unsicheren Kanal zusenden und die anschliessend ausgetauschten Nachrichten, nur für den jeweiligen Empfänger verständlich, verschlüsseln.

Allerdings stellen sich zwei Authentifikationsprobleme: Erstens muss ein Benutzer sicherstellen können, dass der empfangene Public Key tatsächlich vom behaupteten Sender stammt und nicht von einem Gegener eingespiesen wurde, und zweitens müssen auch die Nachrichten selbst authentisiert werden, da jedermann mit Hilfe des Public Keys Nachrichten verschlüsseln kann und es deshalb für den Empfänger nicht möglich ist, zu entscheiden, woher eine empfangene Nachricht kommt. Diese beiden Probleme können mit Hilfe von digitalen Unterschriften gelöst werden. Zur Lösung des ersten Problems zertifiziert z.B. eine vertrauenswürdige Instanz die Public Keys der Benutzer mit Hilfe von digitalen Unterschriften, zur Lösung des zweiten Problems unterschreibt der Sender einer Nachricht diese digital. Ein mechanisches Analogon einer digitalen Unterschrift ist ein Schaukasten, in dem nur mit Hilfe des Schlüssels eine Nachricht angebracht werden kann, welche dann aber für jedermann lesbar ist.

3.1. Ein Public-Key Kryptosystem basierend auf Graphen

Das folgende Beispiel aus [10] wird zu illustrativen Zwecken präsentiert, obwohl die vorgestellte Version selbst nicht kryptographisch sicher ist und das System in der Praxis auch in einer sicheren Version wohl nie verwendet wird. Das Beispiel soll aber illustrieren, dass man für die Errichtung eines Public-Key Systems eine spezifische mathematische Struktur benötigt, und auch dass sehr unterschiedliche mathematische Werkzeuge in der Kryptographie Anwendung finden können.

Gegeben sei ein Graph $G = (V, E)$ mit Knotenmenge V und Kantenmenge $E \subseteq V \times V$. Man stelle sich vor, der Graph repräsentiere die Plätze und Strassen in einer Stadt in südlichen Gefilden. Deshalb ist es wichtig, dass man von jedem Platz aus höchstens eine Strasse zu einem andern Platz gehen muss, um an einen Glacestand zu kommen. Die Stadt soll mit möglichst wenigen Glaceständen (an gewissen Plätzen) ausstattet werden, sodass die obige Bedingung erfüllt ist.

Dieses Problem ist in der Graphentheorie bekannt als das Problem der minimalen dominierenden Menge eines Graphen $G = (V, E)$. Eine dominierende Menge ist eine Teilmenge V' von V sodass für jeden Knoten $u \in V - V'$ ein Knoten $v \in V'$ existiert mit $(u, v) \in E$. Mit andern Worten, jeder Knoten im Graph ist entweder in V' oder ist mit einer Kante mit einem Knoten in V' verbunden. Dieses Problem ist NP-vollständig (siehe [12], S. 190) und es wird vermutet, dass es auch im generischen Fall schwierig ist.

In dem im folgenden beschriebenen Verfahren ist der Public Key ein Graph (siehe Figur 1, rechts) und der geheime Schlüssel ist eine dominierende Menge des Graphen. Der Graph wird konstruiert, indem man zuerst eine Menge von Sternen erzeugt, die paarweise keine gemeinsamen Knoten enthalten (siehe Figur 1, links). Jeder Stern besteht aus einem Zentrum und einer Menge von sternförmig mit dem Zentrum verbundenen Knoten. Diese Menge von Sternen wird anschliessend um beliebige Kanten zwischen Endpunkten verschiedener Sterne zu einem Graphen erweitert, für den die Zentren der Sterne eine dominierende Menge darstellen.

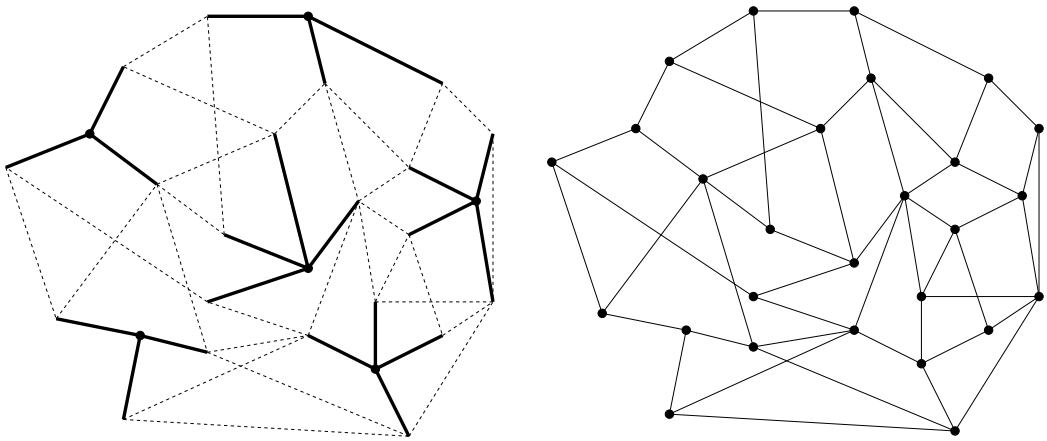

Figure 1. Konstruktion von geheimem Schlüssel und Public Key.

Sei $\{0, \ldots, M-1\}$ der Nachrichtenraum. Als Beispiel kann man $M = 10$ betrachten, d.h. der Klartext ist eine dezimal geschriebene Zahl und jede Ziffer $m \in \{0, \ldots, M-1\}$ wird unabhängig chiffriert, indem den Knoten des Graphen (ausser dem letzten) zufällige Werte $w_1, \ldots, w_{|V|-1}$ zwischen 0 und $M-1$ zugewiesen werden und dem letzten Knoten die Zahl zugewiesen wird, welche die gesamte Summe aller Knotenwerte (modulo M) gleich der Nachricht m macht: $w_{|V|} = m - \sum_{i=1}^{|V|-1} w_i \pmod{M}$. Für jeden Knoten wird jetzt die Summe aller Werte in seiner Nachbarschaft (inkl. sich selbst) berechnet, und diese Liste von $|V|$ Werten wird als Chiffrat übertragen.

Wer den geheimen Schlüssel kennt, kann die Nachricht entschlüsseln, indem er die Summe modulo M der Sternmittelpunkte und damit genau die Summe $m \equiv \sum_{i=1}^{|V|} w_i \pmod{M}$ berechnet. Obwohl das Problem der dominierenden Menge eines Graphen vermutlich sehr schwierig ist, ist dieses System unsicher. Der Leser überlege sich, wie es durch Lösen eines linearen Gleichungssystems gebrochen werden kann. Der Grund ist, dass der Public Key ein sehr spezieller Graph ist, der eine Sternzerlegung besitzt, und dass diese effizient gefunden werden kann. In ähnlicher Weise beruhen einige der verwendeten Kryptosysteme auf vermuteterweise schwierigen mathematischen Problemen, aber es ist oft nicht klar, ob die zu lösenden speziellen Instanzen des Problems nicht trotzdem einfach zu lösen sind. Einige andere Systeme beruhen zwar auf allgemeinen Instanzen eines vermutlich schwierigen Problems wie z.B. der Ganzzahlfaktorisierung, aber es ist oft nicht bewiesen, dass tatsächlich dieses Problem gelöst werden muss, um das System zu brechen.

3.2. Modulares Potenzieren und endliche Gruppen

Das oben beschriebene Public-Key Kryptosystem kann ohne grosse mathematische Exkurse jedem Mittelschüler erklärt werden. Die in der Praxis verwendeten Systeme beruhen aber auf stärkeren mathematischen Strukturen, z.B. auf gewissen endlichen Grup-

pen und im speziellen auf Berechnungen mit ganzen Zahlen, wobei jeweils nur der Rest $R_n(s)$ des Resultates s bei der Division durch einen bestimmten Modulus n benötigt wird. Man spricht von "Rechnen modulo n". Es ist einfach zu beweisen, dass bei einer Rechnung, die nur Additionen und Multiplikationen verwendet, beliebige Zwischenresultate durch ihren Rest modulo n ersetzt werden können, ohne den Rest des Schlussresultates zu verändern. Z.B. gilt $R_9(67 \cdot 101 + 89 \cdot 123) = R_9(4 \cdot 2 + 8 \cdot 6) = R_9(8 + 48) = 2$, was im Prinzip einer Anwendung der bekannten Neunerprobe entspricht.

Wir sind im speziellen an modularen Potenzen interessiert, d.h. $R_n(b^e)$ für gegebene Basis b, Exponenten e und Modulus n. In der Regel sind b, e und n sehr grosse Zahlen (z.B. mit 200 Dezimalstellen), und es stellt sich natürlich die Frage nach der effizienten Berechnung von $R_n(b^e)$.

Zahlen dieser Grössenordnung können im Computer in analoger Weise zu der bekannten ziffernweisen Methode mit Bleistift und Papier multipliziert und dividiert werden. Die Laufzeit dieses trivialen Algorithmus' wächst quadratisch in der Anzahl Stellen der beiden Zahlen. Es gibt aber erstaunlicherweise auch schnellere Algorithmen (siehe z.B. [15]). Der Algorithmus von Karatsuba sei hier kurz beschrieben, da er ein einfaches aber verblüffendes Beispiel eines effizienten rekursiven Algorithmus' ist.

Die zu multiplizierenden Zahlen a und b seien durch k Ziffern in Basis B dargestellt. (B ist in einer Computerimplementation in der Regel eine Zweierpotenz, z.B. 2^{16} oder 2^{32}.) Karatsuba's Idee besteht darin, die Zahlen in zwei Blöcke halber Länge aufzuteilen. Falls k gerade ist, schreiben wir also

$$a = a_1 B^{k/2} + a_2 \quad \text{und} \quad b = b_1 B^{k/2} + b_2,$$

wobei a_1, a_2, b_1 und b_2 Zahlen mit höchstens $k/2$ Ziffern sind. Das Produkt ab kann wie folgt geschrieben werden:

$$\begin{aligned} ab &= a_1 b_1 B^k + (a_1 b_2 + a_2 b_1) B^{k/2} + a_2 b_2. \\ &= a_1 b_1 B^k + (a_1 b_1 + a_2 b_2 + (a_1 - a_2)(b_2 - b_1)) B^{k/2} + a_2 b_2. \end{aligned}$$

Additionen und Subtraktionen sind schnelle Operationen und können (asymptotisch) vernachlässigt werden. Der obige Ausdruck für ab enthält nur 3 verschiedene Produkte von $k/2$-ziffrigen Grössen. Die Laufzeit $T(k)$ für die Multiplikation k-ziffriger Zahlen ist also gegeben durch $T(k) \approx 3T(k/2)$ und die Lösung dieser rekursiven Gleichung liefert $T(k) \approx c \cdot k^d$ für $d = \log_2 3 = 1.585$ und eine gewisse Konstante c, was wesentlich schneller ist als die quadratisch wachsende triviale Methode. Eine für sehr grosse Zahlen noch wesentlich schnellere Methode beruht auf der schnellen diskreten Fourier-Transformation über einem endlichen Körper und hat asymptotische Laufzeit $T(k) = c \cdot k \log k \log\log k$ (siehe [15]) für eine Konstante c. Diese Laufzeiten können auch für die Division erreicht werden.

Ein weiteres Problem bei der Berechnung von $R_n(b^e)$ ist der grosse Exponent. Offensichtlich ist die schrittweise Berechnung $R_n(b^2)$, $R_n(b^3)$, $R_n(b^4), \ldots$ völlig ineffizient. Statt Einerschritten kann man aber viel grössere Schritte nehmen, wobei wir im folgenden das Symbol $R_n(.)$ der Einfachheit halber nicht mehr schreiben. Der folgende Algorithmus mit Variable a berechnet die e-te Potenz von b für einen k-bit Exponenten e mit binärer Darstellung $e_{k-1} \ldots e_1 e_0$, (d.h. $e = \sum_{i=0}^{k-1} e_i 2^i$):

```
a := 1;
for i := k − 1 downto 0 do begin
    a := a ∗ a;
    if e_i = 1 then a := a ∗ b;
end;
```

Die Anzahl Multiplikationen beträgt $k + w(e) - 2 \leq 2k - 2$, wobei $w(e)$ die Anzahl "1" in der binären darstellung von e ist. Für die meisten Exponenten existiert aber eine noch schnellere Methode. So erfordert z.B. die Berechnung von b^{23} mit obiger Methode 7 Multiplikationen (die binäre Darstellung von 23 ist 10111 und es werden der Reihe nach die Werte $b^2, b^4, b^5, b^{10}, b^{11}, b^{22}$ und b^{23} berechnet), während die folgende Sequenz nur 6 Multiplikationen benötigt: $b^2, b^3, b^5, b^{10}, b^{20}, b^{23}$.

Eine mit 1 beginnende Folge von natürlichen Zahlen, in welcher jede Zahl die Summe zweier vorhergehender (nicht notwendigerweise verschiedener) Zahlen ist und deren letztes Element e ist, heisst Additionskette für e [15]. Die kürzeste Additionskette für 23 ist tatsächlich $1, 2, 3, 5, 10, 20, 23$. Ein interessantes Problem ist, für eine gegebene Zahl e kurze Additionsketten zu finden. Wird die absolut kürzeste Additionskette verlangt, so ist das Problem NP-vollständig und gilt deshalb als sehr schwierig. Die kürzesten Additionsketten sind für alle Zahlen bis etwa 10^6 bekannt.

Eine Menge G bildet mit der Operation $* : G \times G \to G$ eine *Gruppe*, falls $*$ assoziativ ist, G ein Element e (das Neutralelement) enthält sodass $g * e = e * g = g$ für alle $g \in G$ und falls für jedes $g \in G$ ein inverses Element g^{-1} existiert, sodass $g * g^{-1} = e$. Ist die Operation $*$ kommutativ, so keisst G Abelsche Gruppe. Die Menge $Z_n = \{0, \ldots, n-1\}$ bildet mit der Addition modulo n eine Gruppe. (In der Mathematik betrachtet man üblicherweise die Restklassen modulo n als die Elemente der Gruppe, aber für den Laien scheint die obige Betrachtung einfacher zu sein, da die Elemente der Gruppe Zahlen statt Mengen sind.)

Die Menge Z_n^* sei definiert als die Menge der Elemente von Z_n, die zu n teilerfremd sind: $Z_n^* = \{1 \leq x < n : ggT(x, n) = 1\}$. Z_n^* bildet zusammen mit der Multiplikation modulo n eine Gruppe. Die Existenz wie auch die effiziente Berechenbarkeit der inversen Elemente folgt aus dem verallgemeinerten Euklid'schen ggT-Algorithmus (siehe z.B. [33]). Man beachte, dass der oben beschriebene Potenzieralgorithmus sich für jede Gruppe, nicht nur für Z_n^*, anwenden lässt.

Falls n eine Primzahl ist, so ist $Z_n^* = \{1, \ldots, n-1\}$ und die Menge Z_n zusammen mit Addition und Multiplikation modulo n bildet deshalb einen endlichen Körper. Endliche Körper existieren genau für die Ordnungen, die eine Primzahlpotenz sind, und die Körper mit 2^m Elementen sind sehr wichtige Strukturen in der Codierungstheorie [5] und der Kryptologie. Zum Beispiel wird auf den Compact Discs ein sogannter Reed-Solomon Code [5] über einem Körper mit 256 Elementen verwendet.

3.3. Zyklische Gruppen, diskrete Logarithmen und das Diffie-Hellman Protokoll

Sei G eine (multiplikativ geschriebene) endliche Gruppe mit Neutralelement 1. Es folgt direkt aus den Gruppenaxiomen, dass für jedes $a \in G$ ein $i > 0$ existiert mit $g^i = 1$ (i heisst die Ordnung von a: $ord(a)$). Eine Gruppe G heisst *zyklisch*, falls ein $g \in G$ mit $ord(g) = |G|$ existiert (ein sogenannter Generator), welches also die ganze Gruppe generiert: $G = \{g^0, g^1, \ldots, g^{|G|-1}\}$ (wobei $g^0 = 1$). Jedes Element einer endlichen Gruppe generiert also per Definition eine zyklische Untergruppe. Das Theorem von Lagrange besagt, dass die Ordnung jeder Untergruppe von G die Gruppenordnung $|G|$ teilt und impliziert, dass $a^{|G|} = 1$ für alle $a \in G$. Man kann zeigen, dass Z_n^* genau dann zyklisch ist, wenn $n = 2, 4,$, eine ungerade Primzahlpotenz oder zweimal eine ungerade Primzahlpotenz ist. Dies ist ein interessantes Gesetz, das Mittelschüler experimentell selbst finden können, der Beweis (siehe z.B. [13]) ist allerdings nicht völlig trivial. Wir merken uns, dass im speziellen für eine Primzahl p die Gruppe Z_p^* zyklisch ist mit $|Z_p^*| = p - 1$.

In einer zyklischen Gruppe lässt sich jedes Element a als Potenz eines gegebenen Generators g schreiben, $a = g^i$, wobei i im Intervall $[0, \ldots, |G| - 1]$ eindeutig ist und der *diskrete Logarithmus* von a zur Basis g genannt wird. Im Gegensatz zu den reellen Zahlen, wo der Logarithmus mit beliebiger Präzision approximiert werden kann, ist das Berechnen von Logarithmen in vielen Gruppen ein (vermutlich) berechenmässig äusserst schwieriges Problem. Es ist z.B. kein effizienter Algorithmus bekannt, der diskrete Logarithmen in der Gruppe Z_n^* für $n \approx 10^{200}$ berechnet, ausser wenn $|Z_n^*|$ nur relativ kleine Primfaktoren besitzt. Der Leser überzeuge sich aber z.B. selbst, dass diskrete Logarithmen in der additiven Gruppe Z_n effizient berechnet werden können.

Das folgende einfache Protokoll von Diffie und Hellman [8], dessen Publikation eine Revolution in der Kryptologie auslöste und welches weltweit in unzähligen Anwendungen verwendet wird, beruht auf der Tatsache, dass Exponentieren selbst in einer grossen Gruppe effizient möglich ist, währenddem die inverse Operation sehr schwierig sein kann. Eine solche Funktion nennt man deshalb Einwegfunktion. Das Protokoll erlaubt zwei Partnern, nennen wir sie Alice und Bob, durch Kommunikation über einen unsicheren Kanal einen gemeinsamen geheimen Schlüssel zu generieren, den ein Gegner nicht berechnen kann, selbst wenn er die gesamte Kommunikation zwischen Alice und Bob abhören kann.

G sei eine Gruppe mit Generator g, in der der DL schwierig zu berechnen ist. G und g seien allgemein bekannt. Der Leser kann sich im folgenden $G = Z_p^*$ vorstellen, wobei p eine grosse Primzahl ist und also $|G| = p - 1$ gilt. Alice und Bob wählen je zufällig eine geheime Zahl x_A (resp. x_B) aus dem Intervall $[0, |G| - 1]$. Alice berechnet $y_A = g^{x_A}$ und Bob berechnet $y_B = g^{x_B}$. Nun tauschen Alice und Bob die Werte y_A und y_B über den unsicheren Kanal aus. Alice berechnet $K_{AB} = y_B^{x_A} = g^{x_A x_B}$ und Bob berechnet

$$K_{BA} = y_A^{x_B} = g^{x_B x_A} = K_{AB}.$$

Sie können nun K_{AB} als gemeinsamen geheimen Schlüssel verwenden, z.B. in einem konventionellen Kryptosystem.

Ein Gegner, der nur y_A und y_B, nicht aber x_A oder x_B kennt, kann K_{AB} nicht berech-

nen, da er nicht diskrete Logarithmen in G berechnen kann. Alice's Public Key ist also g^{x_A} und ihr geheimer Schlüssel ist x_A. Es war allerdings lange nicht bekannt, ob es zum Brechen des Systems wirklich notwendig ist, mindestens einen diskreten Logarithmus in G zu berechnen. Ein dazu praktisch äquivalentes Resultat wurde kürzlich in [23] durch Anwendung der Theorie der elliptischen Kurven über endlichen Körpern bewiesen. Einen guten Überblick über Algorithmen zur Berechnung diskreter Logarithmen gibt [24].

3.4. ElGamal Unterschriften

Diffie und Hellman schlugen in ihrem bahnbrechenden Artikel auch das Konzept der digitalen Unterschriften vor, ohne aber eine konkrete Realisierung zu zeigen, merkten aber nicht, dass die gleiche mathematische Struktur Z_n^* auf verschiedene Weise zu digitalen Unterschriften führen kann. Ein solches System, das sogenannte RSA-System, ist im nächsten Abschnitt beschrieben. Ein anderes, allerdings erst später entdecktes System [9], funktioniert wie folgt. Von diesem System gibt es auch einige Variationen, die zum Teil etwas effizienter sind, aber hier nicht diskutiert werden.

G sei wiederum eine Gruppe wie im Diffie-Hellman Protokoll. Im Gegensatz zum Diffie-Hellman Protokoll, in dem im Prinzip die Gruppenordnung $|G|$ nicht bekannt sein muss, spielt $|G|$ im folgenden eine wichtige Rolle und muss bekannt sein. Die Gruppe Z_p^* mit $|Z_p^*| = p - 1$ kann also verwendet werden.

Um später Nachrichten unterschreiben zu können, wählt Alice wiederum einen geheimen Schlüssel x_A und veröffentlicht $y_A = g^{x_A}$ als Unterschriftenverifikationsschlüssel (Public Key). Um eine Nachricht $m \in \{0, \ldots, |G| - 1\}$ zu unterschreiben, wählt Alice ein zufälliges $k \in \{0, \ldots, |G| - 1\}$ und berechnet $r = g^k$ sowie $s \in \{0, \ldots, |G| - 1\}$ sodass

$$m \equiv x_A r + ks \pmod{|G|}.$$

Die Unterschrift für m ist das Paar (r, s). Um diese Unterschrift zu verifizieren kann jedermann unter Verwendung von y_A überprüfen, ob folgende Gleichung (in G) erfüllt ist:

$$g^m = y_A^r \cdot r^s.$$

Wie beim Diffie-Hellman Protokoll würde das Brechen des Systems, d.h. das Fälschen von Unterschriften die Berechnung eines diskreten Logarithmus erfordern.

3.5. Das RSA-System und die Ganzzahlfaktorisierung

Die Euler'sche φ-Funktion ist definiert als $\varphi(n) = |Z_n^*|$. Es ist einfach zu zeigen, dass

$$\varphi(n) = n \cdot \prod_{i=1}^{r} \left(1 - \frac{1}{p_i}\right),$$

wobei $p_1, \ldots, p_r$ die verschiedenen Primfaktoren von n sind. Es folgt aus dem Lagrange-Theorem, dass

$$x^{\varphi(n)} \equiv 1 \pmod{n} \tag{1}$$

für alle $x \in Z_n^*$. Der Spezialfall, wenn n prim ist, war schon Fermat bekannt: $x^{n-1} \equiv 1 \pmod{n}$ für alle $0 < x < n$.

Im Jahre 1977, also ein Jahr nach der Entdeckung von Diffie und Hellman, publizierten Rivest, Shamir und Adleman vom MIT das folgende elegante Verfahren, das sowohl als Public-Key Kryptosystem als auch als digitales Unterschriftensystem verwendet werden kann. Seine Sicherheit beruht auf der vermuteten berechenmässigen Schwierigkeit, grosse Zahlen in ihre Primfaktoren zu zerlegen.

Ein Benutzer, nehmen wir wiederum Alice, wählt als geheimen Schlüssel zufällig zwei grosse Primzahlen p und q und berechnet $n = pq$ (z.B. mit je etwa 100 Dezimalstellen). Das Problem, solch grosse Primzahlen zu generieren, wird in Abschnitt 3.7 diskutiert. Ferner wählt Alice einen Exponenten e teilerfremd zu $\varphi(n) = (p-1)(q-1)$ (z.B. die Wahl $e = 3$ ist zulässig) und berechnet d als multiplikatives Inverses von e modulo $\varphi(n)$, d.h.

$$ed \equiv 1 \pmod{\varphi(n)} \tag{2}$$

mittels des erweiterten Euklid'schen ggT-Algorithmus [33]. Alice veröffentlicht als Public Key das Paar (n, e).

Wenn Bob eine Nachricht $x \in Z_n^*$ sicher an Alice senden möchte, verschlüsselt er sie mittels einer Exponentiation modulo n mit Exponent e und sendet das Chiffrat y an Alice:

$$y \equiv x^e \pmod{n}.$$

Um y zu dechiffrieren exponentiert sie y mit dem geheimen d:

$$x \equiv y^d \pmod{n}.$$

Die Korrektheit dieser Dechiffriertransformation folgt direkt aus (1) und (2):

$$y^d \equiv x^{ed} \equiv x^{k \cdot \varphi(n)+1} \equiv x \pmod{n}.$$

Die Chiffrier- und Dechiffrieroperationen funktionieren übrigens auch für nicht zu n teilerfremde x, sodass die Bedingung $x \in Z_n^*$ gar nicht geprüft werden muss. Das RSA-System kann gebrochen werden durch Faktorisierung von n, es ist aber nicht bewiesen, dass es nicht auf andere Art eventuell viel schneller gebrochen werden kann.

3.6. Ganzzahlfaktorisierung

Betrachten wir also kurz das Problem der Ganzzahlfaktorisierung, welches allgemein als schwierig, d.h. nicht effizient lösbar gilt. Die grösste je faktorisierte Zahl einer allgemeinen Form besitzt 120 Dezimalstellen, wozu eine Variante des sogennten quadratischen Siebes und massive Rechenzeit verwendet wurde [18]. Im Bereich von 120 Stellen wächst der Zeitaufwand dieses Algorithmus etwa um einen Faktor 10 pro 10 zusätzlichen Stellen. Die Laufzeit des asymptotisch (aber nicht für 120 Stellen) schnellsten bekannten Faktorisierungsalgorithmus [19] ist

$$C_1 e^{C_2 k^{1/3} (\ln k)^{2/3}},$$

wobei k die Anzahl Stellen der zu faktorisierenden Zahl ist und C_1 und C_2 Konstanten sind.

Es existieren spezielle Faktorisierungsalgorithmen für Zahlen von spezieller Form. Die sogenannte $(p-1)$-Methode von Pollard findet Primfaktoren p einer gegebenen Zahl n effizient, falls $p-1$ nur relativ kleine Primfaktoren enthält, was "smooth" genannt wird. Man setzt eine Akkumulatorvariable u auf einen zufälligen Wert a, zählt einen Zähler i hoch und potenziert u jeweils mit q wenn i eine Potenz einer Primzahl q ist. Die Sequenz der q's ist also $2, 3, 2, 5, 7, 2, 3, 11, 13, 2, 17, 19, \ldots$ und die Folge von berechneten Potenzen ist $a^2, a^6, a^{12}, a^{60}, a^{420}, \ldots$. Falls zu einem bestimmten Zeitpunkt der Exponent von a in u ein Vielfaches von $p-1$ ist, was relativ bald eintreten wird wenn $p-1$ nur kleine Primfaktoren und Primzahlpotenzen enthält, so ist $u \equiv 1 \pmod{p}$ und deshalb gilt $ggT(u-1, n) = p$ (ausser in extremen Ausnahmefällen). Durch regelmässiges Testen dieses ggT während der Exponentiation kann p gefunden werden.

Dieses Verfahren nützt also die Tatsache aus, dass die Ordnung einer Untergruppe von Z_n^* "smooth" ist, wobei die Untergruppe nicht explizit bekannt ist und vom unbekannten Primfaktor p abhängt. Lenstra verallgemeinerte 1987 diese Idee [20], indem er vorschlug, elliptische Kurven über Z_n^* als Gruppe zu verwenden. Im Vergleich zum Pollard-Verfahren, bei dem nur eine einzige Gruppe (nämlich eine mit Ordnung $p-1$) zur Verfügung steht, gibt es eine sehr grosse Anzahl von elliptischen Kurven modulo p, und deren Ordnungen sind ungefähr gleichverteilt im Intervall $[p-2\sqrt{p}, p+2\sqrt{p}]$. Lenstra's Verfahren besitzt deshalb gegenüber der Pollard'schen $(p-1)$-Methode den grossen Vorteil, dass die elliptische Kurve variiert werden kann, bis eine gefunden ist, deren Ordnung modulo p "smooth" ist und somit p gefunden wird.

Eine wichtige Anwendung von elliptische Kurven ist die Verwendung als Diffie-Hellman Gruppe, da in elliptischen Kurven der diskrete Logarithmus im allgemeinen noch viel schwieriger ist als in Z_p^*. Einen guten Überblick über Faktorisierungsalgorithmen geben [25] und [7].

3.7. Primzahlerzeugung

In den beschriebenen wie auch in vielen weiteren Public-Key Systemen werden grosse Primzahlen benötigt. Während die Primzahl für das Diffie-Hellman System öffentlich bekannt ist und im Prinzip weltweit die gleiche sein kann, ist es beim RSA-System äusserst wichtig, dass jeder Benutzer die Primzahlen zufällig wählt. Man kann also nicht einfach eine Liste bekannter grosser Primzahlen verwenden.

Eine zufällige Primzahl (aus einem bestimmten Intervall) kann im Prinzip erzeugt werden, indem man fortlaufend zufällige Zahlen wählt und auf Primheit testet, bis ein solcher Test erfolgreich ist. Das Primzahlentheorem besagt, dass die Dichte der Primzahlen in der Grössenordnung von N etwa $1/\ln N$ ist. Es ist also z.B. im Mittel etwa jede 115-te ungerade 100-stellige Zahl eine Primzahl, sodass es bei zufälliger Wahl nicht allzu lange dauert, bis eine Primzahl gefunden ist. Das Problem ist aber, dass es bis heute keinen wirklich effizienten Primzahltest gibt. Zwar wurde kürzlich der erste Primzahltest (basierend auf Abelschen Varietäten) vorgeschlagen [1], dessen Laufzeit polynomial in der Anzahl Stellen der getesteten Zahl ist, aber dieser Algorithmus ist hoffnungslos ineffizient.

Dennoch stellt er einen Durchbruch in der Theorie dar.

In vielen kryptographischen Implementationen werden deshalb die Primzahlen mit sogenannten Pseudo-Primzahltests generiert. Das Fermat-Theorem besagt z.B., dass n sicher keine Primzahl ist, wenn es ein zu n teilerfremdes a gibt mit $a^{n-1} \equiv 1 \pmod{n}$. Man ist versucht, zu vermuten, dass durch Testen genügend vieler Basen a eine zusammengesetzte Zahl mit grosser Sicherheit entdeckt werden kann. Leider gibt es aber Zahlen n, die sogenannten Carmichael-Zahlen, welche zusammengesetzt sind aber trotzdem obige Gleichung für alle zu n teilerfremden a erfüllen. Der Leser überlege selbst, welche Bedingung solche Zahlen erfüllen müssen und prüfe, dass 561 die kleinste solche Zahl ist.

Ein besserer Test ist der sogenannte Miller-Rabin Test. Sei n eine zu testende Zahl mit $n-1 = 2^u v$ mit ungeradem v. Die Zahl n besteht den Test für die Basis b genau dann wenn entweder

$$b^v \equiv 1 \pmod{n}$$

oder

$$b^{2^i v} \equiv -1 \pmod{n}$$

für irgend ein i mit $0 \leq i < u$. Man kann zeigen, dass jede zusammengesetzte Zahl n diesen Test für höchstens $1/4$ aller Basen b im Intervall $\in [1, n-1]$ bestehen kann. Deshalb ist die Wahrscheinlichkeit, dass eine zusammengesetzte Zahl durch t Anwendungen des Tests mit zufällig gewählten Basen nicht als solche entlarvt wird, höchstens $(1/4)^t$. Man kann zwar auf diese Weise nicht beweisen, dass eine Zahl eine Primzahl ist, aber man kann genügende Gewissheit erlangen.

In [22] wird aber ein Algorithmus vorgeschlagen, der beweisbare Primzahlen generiert und erst noch schneller ist als der Miller-Rabin Test. Dort wird auch ein Überblick über die interessante Problematik der Primzahlgenerierung gegeben.

4. Zero-Knowledge Beweisprotokolle und Benutzeridentifikation

In diesem Abschnitt beschreiben wir ein Protokoll, das Alice erlaubt, zu beweisen, dass sie ein Geheimnis kennt, ohne irgendwelche Information darüber wegzugeben. Falls sie dieses Protokoll mit Bob durchführt, um ihn zu überzeugen, so ist die gesamte Kommunikation zwischen den beiden derart, dass Bob sie mit exakt der gleichen Wahrscheinlichkeitsverteilung selbst, ohne Interaktion mit Alice, erzeugen könnte. Sie hängt also scheinbar gar nicht vom Geheimnis ab.

Ein wichtiges Anwendungsszenario ist die Benutzeridentifikation in Computersystemen, vor allem in Netzwerken. Alice kann selbst über ein völlig unsicheres Netz und selbst gegenüber einem überhaupt nicht vertrauenswürdigen System beweisen, dass sie Alice ist, ohne dass weder ein die Leitung abhörender Gegner noch das prüfende System anschliessend betrügen können, d.h. sich als Alice ausgeben können.

Das erste solche Protokoll wurde 1986 von Fiat und Shamir vorgeschlagen [11] und funktioniert wie folgt. Es existiert ein öffentlich bekannter RSA-Modulus $n = pq$, von dem aber niemand (ausser vielleicht eine vertrauenswürdige Instanz) die Faktorisierung kennt. Alice wählt als Geheimnis zufällig ein $x_A \in Z_n^*$ und veröffentlicht als Public Key

$y_A = x_A^2$ (alle Berechnungen sind modulo n). Man kann zeigen, dass Wurzelziehen modulo n einfach ist, wenn die Faktorisierung von n bekannt ist, dass es aber andernfalls nicht einfacher ist als n zu faktorisieren.

Um Bob zu beweisen, dass sie x_A kennt, wählt sie eine zufällige Zahl $r \in Z_n^*$ und sendet ihm $s = r^2$. Dann wählt Bob zufällig ein Bit $b \in \{0, 1\}$ und fordert Alice auf, $t = r \cdot x_A^b$ zu senden. Bob prüft dann, ob $t^2 = y_A^b \cdot s$.

Ein Gegner kann in diesem Protokoll höchstens mit Wahrscheinlichkeit $1/2$ betrügen. Könnte er nämlich für beide Werte von b richtig antworten, so könnte er ja x_A als Quotienten der beiden möglichen Antworten berechnen. Die Grösse x_A ist aber nur Alice bekannt. Durch k-malige unabhängige Wiederholung des Protokolls kann also die Betrugswahrscheinlichkeit auf $1/2^k$ verkleinert werden, sodass Bob mit beliebiger Sicherheit von Alice's Identität überzeugt sein kann. Man beachte, dass für die Wahl $b = 0$ Alice's Antwort gar nicht von x_A abhängt. Trotzdem ist die Überlegung falsch, die Wahl $b = 0$ nütze Bob nichts und er sollte immer $b = 1$ wählen; ein Gegner könnte betrügen, wenn er zu Beginn des Protokolls wüsste, dass $b = 1$ ist.

Bei genügend grossem k ist also Bob überzeugt, dass die überprüfte Person mit an Sicherheit grenzender Wahrscheinlichkeit x_A kennt und deshalb Alice ist. Trotzdem erhält Bob überhaupt keine Information über x_A, die er nicht schon vorher besass (deshalb der Name Zero-Knowledge). Natürlich könnte Bob x_A finden, wenn er n faktorisieren könnte, aber dies gilt vor und nach dem Protokoll gleichermassen. Formal ist ein Protokoll zero-knowledge wenn die gesamte Kommunikation zwischen den beteiligten Parteien vom Empfänger des Beweises selbst mit der gleichen Wahrscheinlichkeitsverteilung simuliert werden könnte, sogar wenn er eine vom eigentlichen Protokoll abweichende (Betrugs-)Strategie verwendet, um Information über x_A zu bekommen. Im Fiat-Shamir Protokoll sieht Bob im Fall $b = 0$ lediglich eine zufällige Zahl und deren Quadrat modulo n, und dies könnte er auch selbst generieren. Ebenso könnte Bob im Fall $b = 1$ selbst ein zufälliges t wählen und $s = t^2/y_A$ berechnen.

Zero-knowledge Protokolle würden z.B. einem misstrauischen Mathematiker im Prinzip auch erlauben, einem Kollegen zu beweisen, dass er den Beweis für eine offene mathematische Vermutung kennt, ohne jegliche Information darüber wegzugeben, wie der Beweis aussieht. Es wäre also denkbar, dass in Zukunft die Fields Medaille an jemand vergeben würde, der lediglich durch Zero-knowledge Protokolle jenseits aller vernünftigen Zweifel bewiesen hat, ein genialer Mathematiker zu sein, ohne aber je den Beweis eines Theorems publiziert zu haben. Es sei bemerkt, dass auch der Autor diese Vision für der Mathematik unwürdig hält.

5. Weitere Kuriositäten

Die Kryptologie kann mit Fug und Recht als Kabinett von Kuriositäten und Paradoxa bezeichnet werden, wobei natürlich normalerweise die Anwendung im Vordergrund steht. In diesem Artikel konnte leider nur die Spitze des Eisbergs gekratzt werden. Einige weitere interessante Protokolle seien im folgenden aber kurz erwähnt, wobei eine Diskussion der dahinterliegenden Mathematik aus Platzgründen leider unterbleiben muss.

Oblivious Transfer ist ein (in der Anwendung übrigens wichtiges) Protokoll, welches

einem Sender Alice erlaubt, eine Nachricht an einen Empfänger Bob zu senden, so dass dieser die Nachricht mit Wahrscheinlichkeit p (z.B. $p = 50\%$) erhält, mit der Restwahrscheinlichkeit $1 - p$ aber absolut keine Information über die Nachricht erhält, und so dass Alice keine Information darüber bekommt, welcher der beiden Fälle eingetreten ist. Dies könnte z.B. wie folgt implementiert werden, wobei die Sicherheit dieses Vorschlags nicht erwiesen ist: Alice generiert einen RSA-Modulus $n = pq$ und sendet ihn Bob. Bob wählt zufällig ein $r > n/2$ aus Z_n^* und sendet Alice $s = r^2$. Alice berechnet die vier Wurzeln von s in Z_n^* und wählt zufällig eine der beiden Wurzeln, die grösser als $n/2$ sind. Diese sei t. Dann sendet Alice $t \cdot g$ an Bob, wobei $g \in Z_n^*$ das Geheimnis sei. Bob kann nur eine der zwei Wurzeln kennen, und mit Wahrscheinlichkeit 50% ist $t = r$, in welchem Fall er g berechnen kann.

Eine interessante Thematik ist digitales Geld. Reine Information, d.h. eine Zahl, ohne physische Bindung an einen materiellen Gegenstand wie ein Stück Papier, repräsentiert einen Wert. Solches Geld kann also z.B. bei der Bank abgehoben werden, in einem Geschäft ausgegeben werden, und vom Geschäft bei der Bank wieder einbezahlt werden. Da das Geld lediglich Information ist, kann damit z.B. über ein Kommunikationsnetz bezahlt werden. Digitales Geld kann fälschungs- und diebstahlsicher implementiert werden (z.B. durch Speicherung in einer PIN-geschützten Chipkarte) und es ist sogar verlustsicher, da man zu hause auf dem PC eine Sicherheitskopie speichern kann. Natürlich zeigt sich hier auch ein Problem, nämlich dass das mehrfache Ausgeben des Geldes, durch den ursprünglichen Besitzer oder durch einen damit Bezahlten, verhindert werden muss. Interessanterweise kann digitales Geld sogar anonym realisiert werden, sodass also die Bank nicht herausfinden kann, wer wo bezahlt hat. Um diese Anonymität zu realisieren benötigt man sogenannte blinde Unterschriften, die einer Bank erlauben, ein Dokument digital zu unterschreiben, ohne es je zu sehen. Das mechanische Analogon einer blinden Unterschrift ist ein verschlossenes Couvert, in dem sich die Meldung und ein Stück Kohlepapier befinden. Jemand kann nun das Couvert aussen unterschreiben und hinterlässt auf der Meldung, die der Empfänger anschliessend entfernen kann, seine durchgepauste Unterschrift.

Eine weitere Thematik ist das sogenannte Secret Sharing. Dabei geht es darum, einen geheimen Schlüssel so auf eine Menge von Personen zu verteilen, dass nur genau spezifizierte Teilmengen von Personen den Schlüssel rekonstruieren können, dass aber alle andern Teilmengen von Personen keine Information über den Schlüssel besitzen, selbst unter Verwendung von unendlichen Computerressourcen. Ein harmloses Anwendungsbeispiel ist ein Banksafe, der z.B. nur von mindestens zwei Direktoren, einem Direktor und zwei Vizedirektoren, oder von fünf Vizedirektoren geöffnet werden können soll. Ein wichtigeres und reales Anwendungsgebiet ist die Sicherung von Atomwaffen durch die USA, wobei genau spezifiziert ist, welche Gruppen von Personen aus einer Menge, die den Präsidenten, den Vizepräsidenten und einige Generäle umfassen kann, die Waffen auslösen können sollen.

Es existieren des weiteren Protokolle, die es einer Menge sich gegenseitig misstrauender Personen erlauben, ein Spiel (z.B. Poker) über ein Telefonnetz zu spielen, d.h., ohne dass sie sich gegenseitig kontrollieren können. Eine erste Frage ist z.B., wie die Karten gemischt werden können.

Andere Protokolle erlauben, Abstimmungen über eine Datenleitung (z.B. das Telefon-

netz) durchzuführen auf eine Art, die jedem Stimmenden garantiert, dass seine Stimme einerseits gezählt worden ist und andererseits aber trotzdem geheim bleibt, selbst gegenüber der Auswertezentrale.

Sollte der Leser Interesse an der Kryptologie entwickelt haben (was natürlich der Zweck dieses Artikels ist), so sei er als ersten Einstieg auf das Buch von Schneier [29] verwiesen, welches allerdings mathematisch nicht sehr tief geht, dafür aber viele Protokolle auf einem intuitiven Niveau beschreibt und eine sehr umfassende Literaturliste enthält.

Literaturverzeichnis

[1] L.M. Adleman and M.A. Huang, Primality testing and abelian varieties over finite fields, *Lecture Notes in Mathematics*, Vol. 1512, Springer Verlag, 1992.

[2] F.L. Bauer, Kryptologie, Berlin: Springer-Verlag, 1993.

[3] C.H. Bennett, F. Bessette, G. Brassard, L. Salvail and J. Smolin, Experimental quantum cryptography, *Journal of Cryptology,* Vol. 5, No. 1, pp. 3-28, 1992.

[4] Bennett, C. H., Brassard, G. and Ekert, A. K., Quantum cryptography, *Scientific American*, October 1992, pp. 26-33. (Erschien im December 1992 als deutsche Übersetzung in *Spektrum der Wissenschaft*, pp. 96-104).

[5] R.E. Blahut, *Theory and Practice of Error Control Codes*, Reading, MA: Addison-Wesley, 1984.

[6] R.E. Blahut, *Principles and Practice of Information Theory*, Reading, MA: Addison-Wesley, 1987.

[7] D.M. Bressoud, *Factorization and Primality Testing*, Berlin: Springer-Verlag, 1989.

[8] W. Diffie and M.E. Hellman, New directions in cryptography, *IEEE Transactions on Information Theory*, Vol. 22, No. 6, pp. 644-654, 1976.

[9] T. El-Gamal, A public key cryptosystem and a signature scheme based on the discrete logarithm, *IEEE Transactions on Information Theory*, Vol. 31, No. 4, pp. 469-472, 1985.

[10] M. Fellows and N. Koblitz, Kid Crypto, *Advances in Cryptology - CRYPTO '92*, Lecture Notes in Computer Science, Vol. 740, pp. 371-389, Berlin: Springer-Verlag, 1993.

[11] A. Fiat and A. Shamir, How to prove yourself: practical solution to identification and signature problems, *Advances in Cryptology - CRYPTO '86*, Lecture Notes in Computer Science, Vol. 263, pp. 186-194, Berlin: Springer-Verlag, 1987.

[12] M.R. Garey and D.S. Johnson, *Computers and Intractability – A Guide to the Theory of NP-Completeness*, New York: W.H. Freeman and Co., 1979.

[13] K. Ireland and M. Rosen, *A classical introduction to modern number theory*, 2nd ed., Berlin: Springer-Verlag, 1990.

[14] D. Kahn, *The code breakers, the story of secret writing*, MacMillan, New York, 1967.

[15] D.E. Knuth, *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, 2nd edition, Addison-Wesley, 1981.

[16] N. Koblitz, *A Course in Number Theory and Cryptography*, Berlin: Springer-Verlag, 1987.

[17] E. Kranakis, *Primality and Cryptography*, Stuttgart: Teubner, and New York: John Wiley & Sons, 1986.

[18] A.K. Lenstra and H.W. Lenstra, Algorithms in number theory, Chapter 12 in *Handbook of Theoretical Computer Science*, J. van Leeuwen (ed.), MIT Press and Elsevier Science Publishers, 1990.

[19] A.K. Lenstra, H.W. Lenstra, M.S. Manasse and J.M. Pollard, The number field sieve, *Proc. 22nd ACM Symposium on Theory of Computing*, pp. 564-572, 1990.

[20] H.W. Lenstra, Jr., Factoring integers with elliptic curves, *Annals of Mathematics*, Vol. 126, pp. 649-673, 1987.

[21] U.M. Maurer, Secret key agreement by public discussion from common information, *IEEE Transactions on Information Theory*, pp. 733-742, May 1993.

[22] U.M. Maurer, Fast generation of prime numbers and secure public-key cryptographic parameters, Tech. Report No. 201, Dept. für Informatik, ETH Zurich, Nov. 1993.

[23] U.M. Maurer, On the equivalence of computing discrete logarithms and breaking the Diffie-Hellman protocol, Preprint, 1994.

[24] K. McCurley, The discrete logarithm problem, in *Cryptology and computational number theory*, C. Pomerance (ed.), Proc. of Symp. in Applied Math., Vol. 42, pp. 49-74, American Mathematical Society, 1990.

[25] C. Pomerance, Factoring, in *Cryptology and computational number theory*, C. Pomerance (ed.), Proc. of Symp. in Applied Math., Vol. 42, pp. 27-47, American Mathematical Society, 1990.

[26] M.O. Rabin, Probabilistic algorithm for testing primality, *Journal of Number Theory*, Vol. 12, pp. 128-138, 1980.

[27] H. Riesel, *Prime numbers and computer methods for factorization*, Boston, Basel, Stuttgart: Birkhäuser, 1985.

[28] R.L. Rivest, A. Shamir, and L. Adleman, A method for obtaining digital signatures and public-key cryptosystems, *Communications of the ACM*, Vol. 21, No. 2, pp. 120-126, 1978.

[29] B. Schneier, *Applied Cryptography*, John Wiley & Sons, Inc., New York, 1994.

[30] C.P. Schnorr, Efficient identification and signatures for smart cards, *Advances in Cryptology – CRYPTO '89*, Lecture Notes in Computer Science, Vol. 435, pp. 239-252, Berlin: Springer-Verlag, 1990.

[31] C.E. Shannon, *A mathematical theory of communication*, Bell Syst. Tech. J., vol. 27, no. 3, pp. 379-423 and 623-656, July 1948.

[32] G.J. Simmons, (Ed.), *Contemporary Cryptology – The Science of Information Integrity*, IEEE Press, 1992.

[33] H.C.A. van Tilborg, *An introduction to cryptology*, Kluwer Academic Publishers, 1988.

[34] D. Welsh, *Codes and Cryptography*, Oxford Science Publications, 1988.