Albrecht Beutelspacher

Kryptologie

Eine Einführung in die Wissenschaft vom Verschlüsseln, Verbergen und Verheimlichen

10. Auflage

SACHBUCH

Springer Spektrum

Albrecht Beutelspacher

Kryptologie

Eine Einführung in die Wissenschaft vom Verschlüsseln, Verbergen und Verheimlichen

10. Auflage

SACHBUCH

Springer Spektrum

Kryptologie

Albrecht Beutelspacher

Kryptologie

Eine Einführung in die Wissenschaft vom Verschlüsseln, Verbergen und Verheimlichen

10., aktualisierte Auflage

Springer Spektrum

Albrecht Beutelspacher
Mathematisches Institut
Justus-Liebig-Universität Gießen
Gießen, Deutschland

ISBN 978-3-658-05975-0 ISBN 978-3-658-05976-7 (eBook)
DOI 10.1007/978-3-658-05976-7

Springer Spektrum

Gedruckt auf säurefreiem und chlorfrei gebleichtem Papier.

Springer Spektrum ist eine Marke von Springer DE. Springer DE ist Teil der Fachverlagsgruppe Springer Science+Business Media
www.springer-spektrum.de

Ohne alle Geheimniskrämerei, aber nicht ohne hinterlistigen Schalk,
dargestellt zum Nutzen und Ergötzen des allgemeinen Publikums.

Vorwort

Aut prodesse volunt aut delectare poetae
aut simul et iucunda et idonea dicere vitae.
(Nützen oder erfreuen wollen die Dichter,
oder, was zugleich erfreulich und nützlich fürs Leben ist, sagen)
(Horaz).

Seit es mit Sprache begabte Lebewesen gibt, gibt es auch vertrauliche Mitteilungen. Das sind Mitteilungen, die nur für eine einzige Person oder nur für einen klar definierten Personenkreis bestimmt sind, und von denen Außenstehende keine Kenntnis erhalten sollen.

Wie kann eine Nachricht „sicher" übermittelt werden? Also so, dass nur der berechtigte Empfänger die Nachricht dechiffrieren kann? Fast noch wichtiger: Wie kann man erreichen, dass die Nachricht wirklich beim Empfänger ankommt, und zwar genauso, wie man sie losgeschickt hat?

Es gibt grundsätzlich verschiedene Möglichkeiten, diese Probleme zu lösen. Die erste Methode besteht darin, die Existenz der Nachricht zu verheimlichen. Man könnte die vertrauliche Nachricht zum Beispiel mit unsichtbarer Tinte schreiben. Oder man könnte in einem unverfänglichen Brief gewisse Buchstaben markieren, indem man zum Beispiel mit einer Stecknadel ein kleines Loch darunter macht; die markierten Buchstaben ergeben die eigentliche Nachricht. Man spricht von Methoden der Steganographie.

Man kann auch versuchen, das Problem organisatorisch zu lösen. Etwa, indem man die Mitteilung durch eine vertrauenswürdige Person überbringen lässt. Zu allen Zeiten haben heimlich Verliebte solche Methoden eingesetzt– und fast alle klassischen Tragödien zeugen vom letztlichen Scheitern dieser Bemühungen.

Eine ganz andersartige Methode besteht darin, vertrauliche Nachrichten zu verschlüsseln. In diesem Fall verheimlicht man nicht ihre Existenz. Im Gegenteil: Man übermittelt die Nachricht über einen unsicheren Kanal, aber so „chiffriert", dass niemand – außer dem wirklichen Empfänger – die Nachricht „dechiffrieren" kann. Dies ist eine ganz perfide Herausforderung des Geg-

ners; solche Herausforderungen wurden in der Regel auch angenommen – und nicht selten wurde der Spieß umgedreht.

Wir werden uns in diesem Buch hauptsächlich mit dieser letzten Methode, also der Verschlüsselung der Nachrichten zum Zwecke der Geheimhaltung beschäftigen.

Ein zweiter Schwerpunkt des Buches ist die Integrität und Authentifikation. Hier geht es nicht darum, eine Nachricht gegen unberechtigtes Lesen zu schützen, sondern vor unberechtigter Änderung. Dieses Problem hat in den letzten Jahren große praktische Bedeutung erworben.

Bis vor einem halben Jahrhundert waren die Militärs die einzigen, die sich professionell mit Kryptologie beschäftigt haben. Nur im militärischen Bereich gab es genügend Motivation – und ausreichende Mittel –, um die damaligen Chiffriermaschinen, diese ausgeklügelten mechanischen Wunderwerke, zu entwickeln, zu bezahlen und zu benützen. Besonders berühmt war die ENIGMA (griechisch für „Rätsel"), die im 2. Weltkrieg von der deutschen Wehrmacht benutzt wurde. Systematische Angriffe auf die ENIGMA wurden bereits vor dem Krieg in Polen und dann während des zweiten Weltkriegs im Britischen Dechiffrierzentrum in Bletchley Park unternommen. Den Briten gelang es dabei nicht nur, das ENIGMA-System zu knacken, sondern sie konnten diese Tatsache auch bis zum Ende des zweiten Weltkriegs vor den Deutschen geheim halten. Eine andere Maschine, der Geheimschreiber T-52 von Siemens & Halske, der zur Übermittlung streng geheimer Nachrichten eingesetzt wurde, blieb während dieser Zeit sicher.

Es gibt eine interessante Verbindung zwischen diesen Angriffen auf Chiffriermaschinen und den Anfängen der Computerentwicklung. Während des zweiten Weltkriegs entwickelten die Engländer elektromechanische und elektronische Maschinen, um die deutschen verschlüsselten Nachrichten zu knacken. Die berühmteste dieser Maschinen, die Röhrenrechenanlage COLOSSUS, kann als der erste digitale Computer angesehen werden. Es ist bemerkenswert, dass der englische Mathematiker Alan M. Turing (1912–1954), der später als der Vater der theoretischen Informatik berühmt wurde, zwar eine entscheidende Rolle im Dechiffrierteam von Bletchley Park gespielt hat, nicht aber an der Entwicklung des COLOSSUS beteiligt war.

Die Tatsache, dass die Kryptologie bei der Geburt der modernen Computer Pate stand, hat Symbolcharakter. Mit der überwältigenden Verbreitung der elektronischen Datenverarbeitung seit den 60-er Jahren des 20. Jahrhunderts ist die Kryptologie auf neue Füße gestellt worden. Dies hat verschiedene Gründe. Einige seien hier genannt:

Beim Versuch, ein gegnerisches System zu brechen, müssen Unmengen von Daten, zum Beispiel Buchstabenketten und Zahlenkolonnen verarbeitet werden: Man muss Daten vergleichen, Mittelwerte, Standardabweichungen und vieles andere mehr berechnen – alles Dinge, die ein Computer sehr viel schneller und besser kann als der Mensch. Die Konsequenz ist, dass Kryptosysteme, die heute mit Erfolg eingesetzt werden sollen, wesentlich komplexer sein müssen als ihre Vorgänger vor zwei oder drei Generationen.

Andererseits ermöglicht moderne Hard- und Software die Implementierung von komplexen und anspruchsvollen mathematischen Algorithmen. Mit diesen erreicht man einen Grad von Sicherheit, zu dem es in der Geschichte keine Parallele gibt: Ein kleiner Zuwachs in der Komplexität eines Algorithmus führt zu einem überdimensionalen Anwachsen der Ressourcen, die zum Brechen des Systems benötigt werden. Der Witz der modernen Kryptologie ist, dass der Computer nicht nur die Ursache vieler Probleme, sondern gleichzeitig der Schlüssel zur ihrer Lösung ist.

Durch das Vordringen elektronischer Datenverarbeitung und insbesondere von elektronischer Kommunikation in immer mehr Bereiche öffneten sich gänzlich neue Aufgabenfelder für die Kryptologie. Neben den „klassischen" Anwendungen im Militär- und Behördenwesen stellten sich ganz neuartige Herausforderungen an die Kryptologie. Große neuartige Anwendungen wie Mobilfunk, Internetkommunikation oder elektronischer Zahlungsverkehr sind ohne moderne Sicherheitsfunktionen unvorstellbar.

Die Gespräche, die Sie über Ihr Handy führen, können prinzipiell abgehört werden – jedenfalls zwischen dem Handy und der ersten Basisstation. Folglich müssen die Gespräche so chiffriert werden, dass ein Abhörer nur sinnlose Geräusche hören kann.

Wenn man den Internetverkehr verschlüsseln möchte, muss man mit einer riesigen Menge von Teilnehmern rechnen. Daher kann ein Schlüsselmanagement wie es noch vor fünfzig Jahren üblich war (je zwei Teilnehmer, die miteinander kommunizieren wollen, tauschen einen geheimen Schlüssel aus), schlechterdings nicht funktionieren. Hier setzt man die Methoden der Public-Key-Kryptographie ein.

Dass Geldüberweisungen elektronisch getätigt werden und nicht über papierene Überweisungsträger erfolgen, hat sich weitgehend durchgesetzt. Stichworte sind „Homebanking" und „electronic cash". Für solche Anwendungen wird ein elektronischer Ersatz für die herkömmliche handschriftliche Unterschrift benötigt. In vielerlei Hinsicht ist die so genannte elektronische Signatur besser als die vertraute handschriftliche Unterschrift.

Es gehört nicht viel Prophetengabe dazu, vorauszusagen, dass die Kryptologie, die sich erst in den letzten Jahrzehnten als seriöse Wissenschaft etabliert hat, auch in der Zukunft ihren rasanten Aufschwung beibehalten wird.

Hier ist ein Wort zur NSA und anderen Geheimdiensten angebracht. In der Tat ist es schockierend, wie viele Daten die NSA abhören und wie viele Kommunikationsverbindungen sie kontrollieren kann. Es scheint so zu sein, dass die Geheimdienste mit ihrer übermächtigen Computerpower alle Sicherheitssysteme aushebeln können. Hier soll nichts beschönigt oder verharmlost werden, zumal wir nicht hinter die Kulissen der Geheimdienste blicken können. Aber: Die einzige Chance, die wir Bürger haben, liegt in der Kryptographie. Gute kryptographische Algorithmen, die in verlässlicher Hard- und Software realisiert sind (hier liegt allerdings der Haken!) können auch scheinbar übermächtigen Gegnern Paroli bieten!

Wir werden auf dieses brisante Thema im Verlauf dieses Buches immer wieder zu sprechen kommen

Jeder, der mit solchen oder ähnlichen Anwendungen zu tun hat, wird zustimmend nicken: „Selbstverständlich brauchen wir Sicherheit! Aber – warum soll die Kryptologie das Allheilmittel sein? Gibt es nicht auch andere Methoden, um Sicherheit zu erreichen?“ Natürlich gibt es andere Methoden! Denken Sie zum Beispiel an die über Jahrhunderte entwickelten ausgefeilten Techniken, die dazu dienen, unsere Banknoten sicher zu machen: Spezialpapier, komplexe (manchmal sogar schöne) Bilder, Präzisionsdruck, Wasserzeichen, Silberdraht, und vieles andere mehr.

Also: Warum Kryptologie?

Die Antwort ist einfach: Kryptologie ist besser! Ein Grund dafür ist, dass Kryptologie eine mathematische Disziplin ist. Das mag übertrieben klingen, ist es aber nicht: Die Mathematik liefert die theoretische Rechtfertigung für die Stärke eines Verfahrens. Mit Mathematik kann man – im Idealfall – beweisen, dass ein kryptographischer Algorithmus ein gewisses Sicherheitsniveau hat. Und wenn die Sicherheit einmal mathematisch bewiesen ist, ist kein Zweifel mehr möglich, dass dieser Algorithmus wirklich sicher ist. Man muss sich dann nicht mehr auf (sich mitunter widersprechende) Expertenmeinungen verlassen, man braucht sich bei der Einschätzung der Sicherheit nicht auf die „heutige Technologie“, die morgen ganz anders sein kann, zu berufen usw.

Ich muss allerdings gestehen, dass solche Beweise bislang nur in sehr wenigen Fällen gelungen sind. Dennoch: Mathematik ist ein vertrauenswürdiges Instrument, um Kryptosysteme systematisch zu untersuchen (das heißt zu entwerfen und zu analysieren). Das ist der Grund, weshalb kryptologische Mechanismen im Zweifel anderen Sicherheitsmechanismen vorzuziehen sind: In dubio pro mathematica!

Die Wissenschaft, die sich mit all diesen Problemen beschäftigt, heißt Kryptologie oder Kryptographie. In den sechs Kapiteln dieses Buches werde ich ihnen die Themen vorstellen, die meiner Meinung nach wesentlich für das Verständnis der modernen Kryptologie sind. Wir werden also den Teil der Kryptologie behandeln, der zur mathematischen Allgemeinbildung gehört. Mein Ziel ist es, die Grundgedanken dieses Gebiets darzulegen. Das kann ich nicht leisten, ohne wenigsten ab und zu ein System im Detail zu behandeln. Aber ich habe versucht, einen lesbaren Text zu schreiben, der weitgehend ohne formalen Ballast auskommt.

Das erste Kapitel hat zwei Ziele. Zunächst betrachten wir einige monoalphabetische Algorithmen über dem natürlichen Alphabet, wie etwa die Cäsar-Chiffre. Es wird sich herausstellen, dass all diese Chiffrierungen relativ leicht zu brechen sind. Bei der Darstellung dieser Algorithmen werden wir uns zwanglos die grundlegenden kryptologischen Begriffe und Bezeichnungen klar machen.

Das zweite Kapitel ist polyalphabetischen Chiffrierungen über dem natürlichen Alphabet gewidmet. Diese sind komplizierter aufgebaut, und man benötigt daher auch präzisere Methoden, um sie zu brechen. Zwei solche Methoden, nämlich den Kasiski-Test und den Friedman-Test werden wir detailliert besprechen.

Das dritte Kapitel ist ein theoretisches Sahnehäubchen. Dort werden Sie nicht nur eine Erklärung des Begriffs „sicher“, sondern auch ein perfektes, also sogar theoretisch sicheres Chiffriersystem (das sogenannte One-Time-Pad) finden. Die zweite Hälfte dieses Kapitels dient dem Studium der „rückgekoppelten Schieberegister“, auf denen sehr viele moderne Algorithmen beruhen.

Im vierten Kapitel werden wir uns mit den Diensten „Integrität“ und „Authentizität“ beschäftigen. In diesem Gebiet der Kryptographie versucht man nicht, Daten geheim zu halten, sondern vielmehr, ihre Unversehrtheit zu garantieren und Gewissheit über den Datenursprung zu erhalten. Diese Problemstellung hat in den letzten Jahren der reinen Geheimhaltung der Daten den Rang abgelaufen und ist dafür verantwortlich, dass die Kryptologie nicht mehr auf den abgeschotteten Bereich der militärischen Anwendungen beschränkt ist, sondern sich im rauen Wind der freien Wirtschaft bewähren kann. An einem alltäglichen Beispiel wird die Bedeutung der Datenintegrität klar: Ich kann es zur Not verschmerzen, wenn ein Unbefugter erfährt, wie viel Geld mir mein Verlag als Honorar für dieses Buch jährlich überweist; mindestens einer der Beteiligten würde aber ziemlich unfreundlich reagieren, wenn der Unbefugte an den Überweisungen etwas verändern kann, sei es den Betrag, sei es die Kontonummer!

In diesem Kapitel werden wir auch die geheimnisvollen „Zero-Knowledge-Algorithmen“ kennen lernen. Dabei geht es um folgende Frage: Können

Sie mich davon überzeugen, ein bestimmtes Geheimnis zu haben, ohne mir auch nur das Geringste davon zu verraten? Diese Algorithmen haben in den letzten Jahren großes Interesse gefunden. Schließlich werden wir Chipkarten behandeln, die sich als das Werkzeug zur Realisierung von kryptographischen Diensten für jedermann als ideales Werkzeug anbieten.

Im fünften Kapitel werden wir die zukunftsweisenden Public-Key-Systeme („asymmetrische" Systeme) vorstellen, deren Einführung durch Diffie und Hellman 1976 eine Revolution der Kryptologie war. Dies zeigt sich zuletzt auch darin, dass seitdem die Kryptologie ein unverzichtbarer Bestandteil der Mathematik und der Informatik geworden ist. Die Eleganz der Public-Key-Algorithmen ist allerdings weit mehr als ein Spielzeug für Mathematiker: Ihre Erfindung war entscheidend durch praktische Probleme motiviert. Wir werden sehen, dass man mit solchen Algorithmen wichtige praktische Probleme auf sehr befriedigende Art und Weise lösen kann.

Im abschließenden sechsten Kapitel studieren wir ein Problem, das am Rande der Kryptologie liegt, nämlich Anonymität. In vielen rechnergestützten Systemen wird Sicherheit vor allem dadurch erreicht, dass alle relevanten Vorgänge aufgezeichnet und ausgewertet werden. Damit sind all diese Ereignisse rekonstruierbar, nichts bleibt verborgen: Der Computer spielt in gewisser Weise die Rolle Gottes: Er weiß alles. Frage: Ist es möglich, ein elektronisches System zu entwerfen (beispielsweise für elektronisches Bezahlen), das grundsätzlich nicht allwissend ist, aber dennoch die notwendige Sicherheit bietet? Anders gefragt: Widersprechen sich Sicherheit und Anonymität? Wir werden zwei Systeme vorstellen, bei denen sich diese beiden Qualitäten vereint sind. Insbesondere werden wir diskutieren, ob es ein elektronisches Äquivalent zum üblichen Münzgeld geben kann.

Sie merken: Das sind zum großen Teil neue, aufregende, sehr praxisbezogene Themen. Wenn Sie fürchten, dass alles sehr kompliziert und undurchschaubar wird, dann kann ich Ihnen sagen: Keine Angst: Die Kryptologie ist ein Glücksfall, da man gerade die neuen und zukunftsweisenden Dinge ziemlich anschaulich erklären kann. Ich habe versucht, alles möglichst verständlich, klar und – hoffentlich – unterhaltsam darzustellen.

Es ist nicht notwendig, die Kapitel der Reihe nach zu lesen. Erschrecken Sie nicht, wenn Ihnen die eine oder andere Stelle zunächst kryptisch vorkommt. In den allermeisten Fällen ist der folgende Text auch ohne Kenntnis dieser „schwierigen" Stelle verständlich.

Mein Rat: Überblättern Sie ruhig die eine oder andere Stelle – und tun Sie das guten Gewissens!

Am Ende jeden Kapitels finden sich Übungsaufgaben, insgesamt weit über 100. Alle Übungsaufgaben werden Ihnen, so hoffe ich, Spaß machen; die meisten sind einfach zu lösen. [Hinweis: Die schwierigeren Übungsaufgaben enthalten einen Hinweis zu ihrer Lösung.]

Einige wenige, etwas schwierigere Aufgaben sind durch das Symbol * gekennzeichnet. Sie werden auch Programmieraufgaben finden. Durch diese können Sie sich überzeugen, dass die dargestellten Verfahren auch wirklich funktionieren. Keine dieser Aufgaben, die man an dem „Klammeraffen" @ erkennt, ist besonders schwierig. Manche erfordern einige Zeit – wie das beim Programmieren so üblich ist.

Einige Aufgaben fordern Sie heraus, einen Geheimtext zu entschlüsseln. Wenn Sie kontrollieren wollen, ob Sie auf der richtigen Spur sind, haben Sie die Möglichkeit, im Anhang bei *Entschlüsselung der Geheimtexte* nachzuschauen.

Eine Erfahrung will ich Ihnen nicht vorenthalten: Ich habe von mehr als einer hübschen jungen Dame gehört, die es anregend fand, mit diesem Buch in die Badewanne zu steigen und sich dort der Lektüre hinzugeben. Ein schöneres Kompliment kann ich mir kaum vorstellen! Ich hoffe, dass auch Sie Spaß bei der Lektüre dieses Buches haben.

Mein Dank gilt allen, die dieses Buchprojekt unterstützten, kritisch begleiteten und förderten. Es sind zu viele, als dass ich sie hier alle nennen könnte.

An erster Stelle danke ich meinen nächsten Angehörigen, Monika, Christoph und Maria. Sie mussten nicht nur häufig als Versuchskaninchen herhalten, sondern haben mir auch mehrere Male großzügig Urlaub zum Bücherschreiben gewährt.

Ferner gilt mein Dank meinen Kollegen A, C, F, I, J, L, M, R, U – in alphabetischer Reihenfolge, wobei einige Buchstaben mehrfach zu zählen sind und mindestens einer fett zu drucken gewesen wäre. Sie haben das Entstehen dieses Buches auf mannigfaltige Weise gefördert: durch akribische Kritik, durch konstruktive Vorschläge, durch aufmunternde Gespräche, durch inspirierende Sitzungen, durch schnelles Beschaffen von Material usw. usw.

Ein besonderer Dank gebührt Frau Dr. Ute Rosenbaum, die sich in der letzten Phase der Herstellung dieses Buches engagiert und effizient alle möglichen technischen und nichttechnischen Probleme gelöst hat.

Dem Verlag Springer Spektrum danke ich für die langjährige problemlose, freundliche und geduldige Zusammenarbeit.

Inhaltsverzeichnis

1

Cäsar oder Aller Anfang ist leicht!

Ibich habibebi dibich,
Lobittebi, sobi liebib.
Habist aubich dubi mibich
Liebibä Neibin, vebirgibib.
Nabih obidebir febirn
Gobitt seibi dibir gubit.
Meibin Hebirz habit gebirn
Abin dibir gebirubiht
(Joachim Ringelnatz).

Es nicht immer leicht, jung und verliebt zu sein. Die klassische Literatur lehrt uns: Wenn immer eine Julia ihrem geliebten Romeo eine vertrauliche Botschaft übermitteln möchte, dann gibt es fast immer einen missgünstigen Bösewicht, der das junge Glück dadurch zu stören versucht, dass er den Brief abfängt (siehe Abb. 1.1).

Vielleicht möchte der Missgünstling die Nachricht „nur" lesen. Das Rezept gegen einen solchen *passiven Angriff* ist Verschlüsselung, das Thema dieses und der folgenden drei Kapitel. Gegen einen *aktiven Angreifer*, also einen Bösewicht, der die Nachricht zu verfälschen trachtet, ist ein anderes Kraut gewachsen: Romeo und Julia sollten die Methoden aus Kap. 4 und 5 studieren, um dagegen gewappnet zu sein.

Die Moral von der Geschicht für Verliebte, aber auch für Diplomaten, Internetbenutzer, Börsenspekulanten – kurz für alle, die vertrauliche Nachrichten verschicken müssen, muss lauten, geeignete Gegenmaßnahmen einzusetzen, die dem Angreifer die Suppe versalzen. Eine Möglichkeit ist die Verschlüsselung: Julia ersetzt die Buchstaben ihres heimlichen Briefchens durch andere Buchstaben oder Zahlen oder Symbole.

Es ist natürlich keine Kunst, eine Nachricht so zu verunstalten, dass überhaupt kein Mensch mehr etwas damit anfangen kann. Die Herausforderung für die Kryptologie besteht vielmehr darin, die Nachricht so zu transformieren, dass niemand außer dem berechtigten Empfänger diese entziffern kann. Daher muss der Empfänger dem Angreifer etwas voraushaben. Mit Hilfe die-

A. Beutelspacher, *Kryptologie*, DOI 10.1007/978-3-658-05976-7_1,

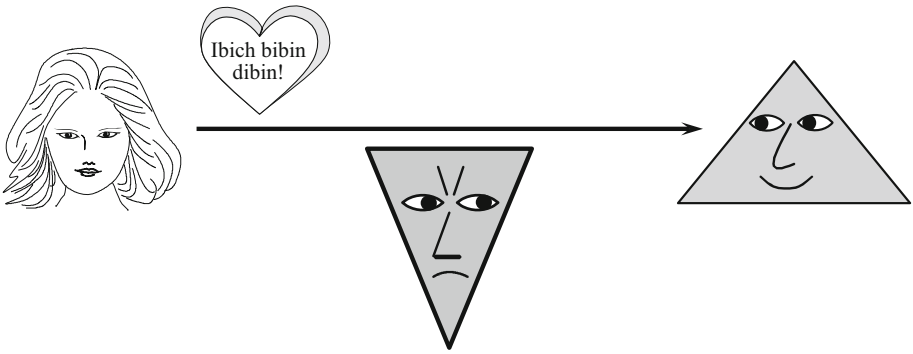


Abb. 1.1 Ein passiver Angreifer

ser Information kann der Empfänger entschlüsseln; sie darf keinem Angreifer zur Verfügung stehen, denn sonst könnte dieser die Nachricht genau so leicht wie der Empfänger entschlüsseln. Man nennt diese exklusive Information den *Schlüssel*. Die klassischen Verschlüsselungsverfahren sind so aufgebaut, dass Sender und Empfänger einen gemeinsamen geheimen Schlüssel besitzen, mit dem der Sender verschlüsselt und der Empfänger entschlüsselt. Da Sender und Empfänger über das gleiche Geheimnis verfügen, nennt man solche Verfahren *symmetrisch*. In Kap. 5 werden wir sehen, dass es auch asymmetrische Verschlüsselungsalgorithmen gibt: In solchen Systemen braucht nur der Empfänger einen geheimen Schlüssel.

In diesem Kapitel betrachten wir die in gewissem Sinne einfachsten symmetrischen Verschlüsselungsalgorithmen, nämlich solche, bei denen ein- und derselbe Buchstabe immer durch ein- und dasselbe Symbol ersetzt wird. Zum Beispiel könnte der Klartextbuchstabe e stets mit dem Geheimtextbuchstaben K chiffriert werden. Gegen Ende des Kapitels wird uns klar werden, dass solche Systeme in der Regel nicht allzu empfehlenswert sind.

Zunächst einige Worte zur Terminologie. Die Begriffe *Kryptologie* und *Kryptographie* sind aus den griechischen Wörtern $\kappa\rho\upsilon\pi\tau\omicron\sigma$ (geheim), $\lambda\omicron\gamma\omicron\sigma$ (das Wort, der Sinn), und $\gamma\rho\alpha\varphi\varepsilon\iota\nu$ (schreiben) gebildet. Beide bezeichnen die Kunst und die Wissenschaft, die sich damit beschäftigt, Methoden zur Verheimlichung von Nachrichten zu entwickeln. Manche Leute unterscheiden zwischen *Kryptographie*, der Wissenschaft von der Entwicklung von Kryptosystemen, *Kryptoanalyse*, der Kunst diese zu brechen und bezeichnen mit *Kryptologie* die Gesamtheit dieser Wissenschaften. Es besteht aber keine Gefahr von Missverständnissen, wenn man Kryptographie und Kryptologie

synonym benutzt. Ganz sicher ist der Bedeutungsunterschied dieser beiden Begriffe bei weitem nicht so groß wie bei Geologie und Geographie oder Philologie und Philosophie oder gar Astronomie und Astrologie.

Der Text, die Nachricht, die Buchstaben- oder Zeichenfolge, die wir übermitteln wollen, heißt der *Klartext*; wir werden den Klartext in der Regel durch kleine Buchstaben a, b, c, … darstellen. Die verschlüsselte Nachricht (also die Buchstaben- oder Zeichenfolge, die tatsächlich übermittelt wird) nennen wir den *Geheimtext* (in der älteren Literatur wird der Geheimtext auch *Kryptogramm* genannt); ihn werden wir in Großbuchstaben A, B, C, … schreiben. Den Verschlüsselungsvorgang nennen wir *Chiffrieren*, den Entschlüsselungsvorgang *Dechiffrieren*. Der Sender chiffriert also, während der Empfänger dechiffrieren muss, bevor er die Nachricht lesen kann.

Die Texte, die wir verschlüsseln werden, bestehen aus *Zeichen*; die Zeichen bilden insgesamt ein *Alphabet*. In den ersten beiden Kapiteln wird unser Alphabet meist das natürliche Alphabet {a, b, c, …, x, y, z} sein. Wir könnten aber als Zeichen z. B. auch die Zahlen 1, …, 26, die Bits 0 und 1 oder, wenn es für unsere Betrachtungen günstig ist, auch die binären Folgen der Länge 64 wählen. In diesem Fall ist das Alphabet die Menge

$$\{(a_1, ..., a_{64} | a_1 \in \{0,1\}\}.$$

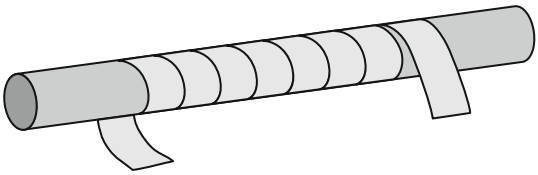
In der Tat arbeiten die heutigen Chiffrierverfahren auf Bits oder Folgen von Bits. In Abschn. 1.7 werden wir einige dieser Verfahren beschreiben.

Im Widerspruch zu der Ankündigung in der Überschrift beginnt die Geschichte der Kryptographie *vor* Cäsar.

1.1 Die Skytala von Sparta

Die Geschichte fängt vor ungefähr 2500 Jahren an. Wie wir von dem griechischen Historiker Plutarch wissen, benutzte die Regierung von Sparta folgende trickreiche Methode zur Übermittlung geheimer Nachrichten an ihre Generäle: Sender und Empfänger mussten beide eine so genannte *Skytala* (sprich: Skýtala) haben; das waren zwei Zylinder mit genau dem gleichen Umfang. Der Sender wickelte ein schmales Band aus Pergament spiralförmig um seine Skytala und schrieb dann der Länge nach seine Nachricht auf das Band (vergleiche Abb. 1.2). War das Band abgewickelt, konnte die Nachricht nur von einer Person gelesen werden, die einen Zylinder genau desselben Umfangs hatte – hoffentlich nur der Empfänger.

Abb. 1.2 Eine Skytala



Wir können uns eine Skytala-Verschlüsselung auch so vorstellen, dass man den Text spaltenweise (d. h. von oben nach unten) in einer bestimmten Anzahl von Zeilen aufschreibt.

Wir betrachten ein Beispiel in moderner Sprache. Stellen wir uns vor, wir hätten einen Papierstreifen abgefangen, auf dem wir die folgende Buchstabenfolge lesen:

S I H L T I T A D O C I U E L H S P R O E T T K G R D Z R I H A I Y E E S E P !

Die Skytala des Senders hat einen Umfang, den wir durch die Anzahl von Buchstaben ausdrücken können, die „einmal um die Skytala herum" passen. Mit anderen Worten: Die Zahl u ist die Anzahl der Zeilen, in die wir den Text anordnen. Zur Analyse probieren wir einfach verschiedene Umfänge u aus. Wenn wir u = 4 wählen, ergibt sich vollkommener Unsinn:

S T D U S E G R I S
I I O E P T R I Y E
H T C L R T D H E P
L A I H O K Z A E !

Wenn wir aber den Text in u = 5 Zeilen anordnen, wird die Botschaft klar:

S I C H E R H E
I T I S T D A S
H A U P T Z I E
L D E R K R Y P
T O L O G I E !

Algorithmus 1.1: Skytala-Verschlüsselung

Schlüssel:	Der Umfang u der Skytala bzw. die Anzahl der Zeilen.
Chiffrieren:	Man schreibt den Klartext zeilenweise in ein Schema mit genau u Zeilen; man erhält den Geheimtext, indem man den Text spaltenweise liest.
Dechiffrieren:	Man schreibt den Geheimtext spaltenweise in ein Schema mit genau u Zeilen; daraus erhält man den Klartext, indem man zeilenweise liest.

Die Skytala ist der Prototyp eine *Transpositionschiffre*; bei einer solchen bleiben die Buchstaben, was sie sind, aber nicht, wo sie sind. Ein Mathematiker würde eine Transpositionschiffre beschreiben als eine Permutation der Stellen des Klartextes. Viele populäre Algorithmen sind Transpositionschiffren. (Vergleiche zum Beispiel die Übungsaufgaben 6 und 26). Kapitel III des Buches [Smi71] enthält eine reiche Auswahl von „klassischen" Transpositionsalgorithmen.

Transpositionsalgorithmen sind ein wichtiger Baustein für moderne Algorithmen. Die andere Komponente sind die *Substitutionsalgorithmen*; bei diesen wird die Nachricht dadurch unlesbar gemacht, dass jeder Buchstabe zwar seine Position behält, aber durch einen anderen ersetzt wird.

Dies ist das Stichwort für den Auftritt Cäsars.

1.2 Verschiebechiffren

Einer der ersten, der kryptologische Techniken benutzt haben soll, war der römische Feldherr und Staatsmann C. Julius Cäsar (100 bis 44 v. Chr.). Bei Sueton (Caes. LVI) lesen wir:

> Exstant et [epistolae] ad Ciceronem, item ad familiares de rebus, in quibus, si qua occultius perferenda erant, per notas scripsit, id est sic structo litterarum ordine, ut nullum verbum effici posset; quae si qui investigare et persequi velit, quartam elementorum litteram, id est D pro A et perinde reliquas commutet.

Übersetzt lautet dies etwa

> Es existieren auch [Briefe von Cäsar] an Cicero und an Bekannte über Dinge, in denen er, wenn etwas vertraulich übermittelt werden musste, in Geheimschrift schrieb. D. h. er veränderte die Ordnung der Buchstaben derart, dass

kein einziges Wort mehr ausgemacht werden konnte. Wenn jemand das entziffern und den Inhalt erkennen wollte, so musste er den vierten Buchstaben des Alphabets, als D, für A einsetzen, und so mit den andern.

Die von Cäsar benutzte Chiffre erhält man, wenn man unter das Klartextalphabet das Geheimtextalphabet schreibt – aber um 23 Stellen nach rechts oder, was dasselbe bedeutet, um 3 Stellen nach links versetzt:

Klartext:	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Geheimtext:	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

Chiffriert wird, indem ein Klartextbuchstabe durch den darunter stehenden Geheimtextbuchstaben ersetzt wird. Zum Beispiel wird aus dem Wort „Klartext“ die scheinbar sinnlose Buchstabenfolge NODUWHAW. Die Dechiffrierung ist genauso einfach: Jeder Geheimtextbuchstabe wird in den darüber stehenden Klartextbuchstaben zurückübersetzt.

Sie fragen zu Recht, warum Cäsar das Geheimtextalphabet gerade um 3 Stellen verschoben hat. Die Antwort ist einfach: Dafür gibt es überhaupt keinen Grund! In der Tat kann man das Alphabet um eine beliebige Anzahl von Stellen verschieben. Sender und Empfänger müssen sich nur merken, um wie viele Stellen das Geheimtextalphabet verschoben wurde. Das können Sie auch daran erkennen, in welchen Buchstaben der Klartextbuchstabe a (oder e oder irgendein anderer Klartextbuchstabe) chiffriert wurde.

Da unser Alphabet aus 26 Buchstaben besteht, gibt es genau 26 solche Chiffrierungen; man nennt sie *Verschiebechiffren* (oder auch *additive* Chiffren – siehe Abschn. 1.4). Darunter befindet sich auch die *triviale* Chiffrierung a $\longmapsto$ A, b $\longmapsto$ B, …, z $\longmapsto$ Z, die wohl noch niemand zu Geheimhaltungszwecken verwendet hat.

Algorithmus 1.2: Cäsar-Verschlüsselung

Schlüssel:	Ein Buchstabe, der die Einstellung des Geheimtextalphabets angibt, z. B. Geheimtextäquivalent von **a** oder von **e**.
Chiffrieren:	Ein Buchstabe des Klartextalphabets wird verschlüsselt, indem er durch den unmittelbar darunter stehenden Buchstaben ersetzt wird.
Dechiffrieren:	Ein Buchstabe des Geheimtextalphabets wird entschlüsselt, indem er durch den unmittelbar darüber stehenden Buchstaben ersetzt wird.

Tabelle 1.1 gibt eine Übersicht über alle 26 Verschiebechiffren.

Tab. 1.1 Die 26 Verschiebechiffren

Klartextalphabet:	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Geheimtextalph.:	A	B	C	D	E	F	G	H	J	I	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
	B	C	D	E	F	G	H	J	I	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
	C	D	E	F	G	H	J	I	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
	D	E	F	G	H	J	I	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
	E	F	G	H	J	I	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
	F	G	H	J	I	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
	G	H	J	I	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
	H	J	I	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Natürlich ist es auch mit einer solchen Tabelle immer noch vergleichsweise mühsam, einen langen Text zu chiffrieren. Bereits 1470 hat das italienische Universalgenie Leon Battista Alberti (1404–1472), der völlig zu Recht „Vater der modernen Kryptologie“ [Kah67] genannt wird, eine Maschine erfunden, die das Verschlüsseln mechanisiert. Abbildung 1.3 zeigt eine „Maschine“, mit der man alle 26 Verschiebechiffren realisieren kann. Sie besteht aus zwei Kreisscheiben, auf denen jeweils das Alphabet in natürlicher Reihenfolge geschrieben ist. Die beiden Scheiben sind an ihren Mittelpunkten so aneinander befestigt, dass die innere gegenüber der äußeren gedreht werden kann. So kann man jede gewünschte Verschiebechiffre einstellen, und das Chiffrieren bzw. Dechiffrieren ist dann keine Kunst mehr: Chiffrieren heißt Lesen von außen nach innen, Dechiffrieren bedeutet Lesen von innen nach außen.

Es gibt noch eine weitere Darstellung der Verschiebechiffren. Wenn man nämlich statt Buchstaben Zahlen schreibt, wird das Ganze noch einfacher. Wir identifizieren dazu A mit 1, B mit 2, C mit 3, … und schließlich Z mit

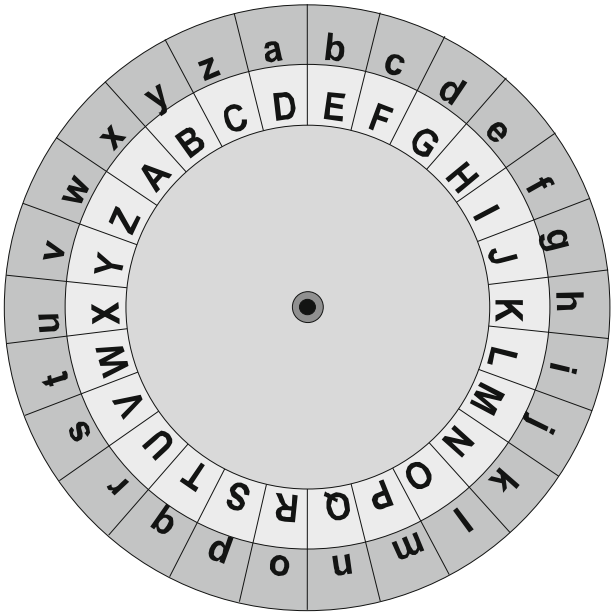


Abb. 1.3 Eine Chiffriermaschine

26. Eine Verschiebechiffre ist dann nichts anderes als die Addition einer festen Zahl.

Zum Beispiel entspricht der Verschiebchiffre, die wir oben betrachtet haben, die Addition der Zahl 3. Ein formales Problem entsteht dann, wenn wir etwa eine Addition der Art $24 + 3$ auszuführen haben. Das kann man sich so vorstellen, dass man zu 24 die Zahl 3 addiert, aber nach 26 wieder von vorne zu zählen anfängt. Das bedeutet, die Zählreihe lautet …, 24, 25, 26, 1, 2, 3, … Also ist in diesem Sinne $24 + 3 = 1$. Man kommt zu diesem Ergebnis auch, wenn man von der „normalen" Summe $24 + 3$ die Zahl 26 subtrahiert. Man schreibt $(24 + 3) \bmod 26$ und sagt, dass man „modulo 26" rechnet. (Man vergleiche dazu auch den Abschn. 1.4.)

Algorithmus 1.3: Cäsar-Verschlüsselung mit Zahlen

Schlüssel :	Eine Zahl t zwischen 0 und 25.
Chiffrieren:	Sei x die Nummer des Klartextbuchstaben, dann ist $(x + t) \bmod 26$ die Nummer des Geheimtextbuchstabens. Dabei ist a mod 26 diejenige Zahl, die sich als Rest bei der Division von a durch 26 ergibt.
Dechiffrieren:	Sei y die Nummer des Geheimtextbuchstabens. Dann ist $(y - t) \bmod 26$ die Nummer des Klartextbuchstabens.

Bereits an den Verschiebechiffren werden zwei äußerst wichtige Begriffe deutlich, die man gut unterscheiden muss. Es handelt sich um die Begriffe „Chiffrieralgorithmus“ und „Schlüssel“. Den *Chiffrieralgorithmus* können wir uns in obigem Beispiel durch die Tab. 1.1 oder durch die „Maschine“ aus Abb. 1.3 realisiert denken. Demgegenüber ist der *Schlüssel* beispielsweise die Anzahl der Stellen, um welche die beiden Alphabete verschoben wurden, oder die aktuelle Einstellung der Scheiben, oder das Geheimtextäquivalent des Buchstabens a, oder ähnliches. Der Schlüssel beschreibt genau, wie der Algorithmus in einer speziellen Situation verwendet wird.

Die Kommunikationspartner müssen sich grundsätzlich über den Chiffrieralgorithmus einigen und vor der Übertragung der Nachricht den Schlüssel vereinbaren bzw. austauschen.

Dazu einige Bemerkungen: Chiffrieralgorithmus und Schlüssel haben völlig verschiedene Funktionen und müssen daher klar unterschieden werden. Der Algorithmus ist in aller Regel ziemlich „groß“. Zum Beispiel sind viele Algorithmen durch eine mechanische Maschine realisiert oder beruhen auf einem mehr oder weniger komplexen öffentlich zugänglichen mathematischen Verfahren. Daraus ergibt sich, dass der Algorithmus de facto nicht geheim gehalten werden kann. Das bedeutet wiederum, dass die gesamte Sicherheit eines Kryptosystems auf der Geheimhaltung des Schlüssels beruht.

Diese Forderung mag übertrieben erscheinen, sie ist aber äußerst realistisch: Für jemanden, der eine Nachricht unberechtigterweise lesen will, ist es vergleichsweise einfach, in den Besitz des Algorithmus (z. B. einer „Maschine“) zu kommen. Dann weiß dieser böse Wicht bereits über den Chiffrieralgorithmus völlig Bescheid; er kennt aber (hoffentlich!) den aktuellen Schlüssel noch nicht.

Daraus ergibt sich zwingend, dass der Schlüssel auf „sicherem Wege“ übermittelt werden muss. Wozu dann das Ganze? Dann könnte man doch gleich die Nachricht auf diesem sicheren Kanal übertragen! Man hat die geheime Übermittlung der Nachricht auf die geheime Übermittlung des Schlüssels zurückgeführt. Mathematiker lieben es, ein Problem auf ein anderes zurückzuführen. Das ist aber nur dann eine gute Strategie, wenn das zweite Problem einfacher zu lösen ist. Ist es tatsächlich einfacher, den Schlüssel geheim zu übertragen als die Nachricht?

- Während die Nachricht in der Regel sehr lang ist, möchte man den Schlüssel so kurz wie möglich wählen. Der Schlüssel ist ein Buchstabe, ein Wort, eine Zahl, eine Folge von Bits usw. In den meisten Fällen hat der Schlüssel eine feste Länge, etwa 128 Bit. Der zusätzliche Aufwand für die sichere Übertragung des Schlüssels ist dann stark reduziert.

- Sender und Empfänger können den Zeitpunkt und den Modus der Schlüsselübergabe frei wählen. Der Schlüssel kann zum Beispiel Tage vorher vereinbart werden. Demgegenüber muss die Nachricht oft zu einem Zeitpunkt übertragen werden, der von den Kommunikationspartnern nicht beeinflusst werden kann. Man denke an politische Ereignisse, überraschende Börsenentwicklungen usw.
- Mit Hilfe der Public-Key-Systeme (siehe Kap. 5) kann man die Schlüssel sicher austauschen, um dann die Verschlüsselung mit Hilfe konventioneller Verfahren durchzuführen.

Hier muss noch auf eine andere Gefahr hingewiesen werden. Wenn der Schlüssel ausgetauscht ist, muss er sicher gespeichert werden. Es darf nicht möglich sein, den Schlüssel aus der Maschine auszulesen. Es ist klar, dass ein Schlüssel nur dann sicher gespeichert ist, wenn das Gerät durch physikalische Maßnahmen unausforschbar gemacht wird.

1.3 Kryptoanalyse

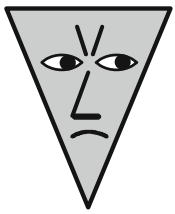
Nun ist der Zeitpunkt gekommen, die Fronten zu wechseln. Die Kryptologie beschäftigt sich nämlich nicht nur damit, Systeme zur Geheimhaltung zu entwerfen; eine ihrer zentralen Aufgaben besteht vielmehr darin, solche Systeme zu „knacken“ oder dies jedenfalls zu versuchen.

Spielen wir also die Rolle des Bösewichts. Etwas vornehmer wird das folgendermaßen ausgedrückt: Wir arbeiten als *Kryptoanalytiker* und führen eine *Kryptoanalyse* des Geheimtexts beziehungsweise des zu untersuchenden Kryptosystems durch. Der Entwickler eines Kryptosystems muss immer mit der Möglichkeit rechnen, dass der Algorithmus dem Gegner bekannt wird. Außerdem empfiehlt es sich, den Gegner nicht zu unterschätzen, sondern ihm höchste Intelligenz zuzubilligen; sein Name soll Mr. X sein (siehe Abb. 1.4).

Wir stellen uns vor, dass Mr. X den folgenden Geheimtext abgefangen hat:

MRNBNA CNGC RBC WRLQC VNQA PNQNRV

Abb. 1.4 Mr. X



Aufgrund gewisser Indizien ist er zu der Vermutung gelangt, dass dieser Geheimtext mit Hilfe einer Verschiebechiffre verschlüsselt wurde. Beispielsweise könnte er eine der oben beschriebenen Chiffriermaschinen „gefunden" haben. Einen solchen Text kann man nun prinzipiell auf zwei Weisen analysieren.

1. „Systematisches" Durchprobieren aller Möglichkeiten
Da es nur 26 Verschiebechiffren gibt, ist der Aufwand dafür nicht allzu groß. Mr. X kann den Aufwand aber noch erheblich reduzieren, indem er nicht auf den gesamten Text alle 26 Verschiebechiffren anwendet, sondern nur auf einen kleinen Textteil.

Betrachten wir zum Beispiel das „Wort" RBC. Probiert man alle „Verschiebungen" dieser Buchstabenfolge durch, so erkennt man leicht, dass unter allen möglichen Klartextäquivalenten das Wort *ist* als einziges einen Sinn ergibt. Daher ist es sehr wahrscheinlich, dass der Geheimtext durch eine Verschiebung um 9 Stellen gewonnen wurde. Mr. X verifiziert diese Vermutung, indem er den gesamten Text dechiffriert:

dieser text ist nicht mehr geheim.

Diese Methode, eine Verschiebechiffre zu brechen, ist deshalb so gut, weil die allermeisten Buchstabenkombinationen im Deutschen völlig bedeutungslos sind. Obwohl diese Beobachtung eine wichtige Grundlage für viele kryptoanalytische Verfahren ist, hat sie doch in der hier dargestellten Form einen gravierenden Nachteil. Diese Methode kann nämlich nur mit unangemessen hohem Aufwand automatisiert werden. Dies trifft insbesondere dann zu, wenn man den Test nur einen Teil des Textes, etwa jeden 20. Buchstaben anwendet. Wenn diese Methode in einem Rechner selbstständig ablaufen soll, so muss dieser alle, oder jedenfalls sehr viele, deutsche Wörter gespeichert haben. Obwohl das prinzipiell zwar möglich ist, würde man doch mit Kanonen nach Spatzen schießen. Dies kann man der zweiten Methode nicht vorwerfen.

Algorithmus 1.4: Systematische Schlüsselsuche (exhaustive key search)

Man kann jeden Algorithmus dadurch brechen, dass man alle Schlüssel systematisch auf einen gegebenen Geheimtext anwendet.

Man braucht im schlimmsten Fall k Versuche (wobei k die Anzahl aller Schlüssel ist), im Durchschnitt wird man k/2 Versuche brauchen.

Man kann diesen Angriff dadurch unmöglich machen, dass man die Anzahl der Schlüssel groß macht.

Tab. 1.2 Häufigkeiten der Buchstaben der deutschen Sprache

Buchstabe	Häufigkeit	Buchstabe	Häufigkeit
a	6,51 %	n	9,78 %
b	1,89 %	o	2,51 %
c	3,06 %	p	0,79 %
d	5,08 %	q	0,02 %
e	17,40 %	r	7,00 %
f	1,66 %	s	7,27 %
g	3,01 %	t	6,15 %
h	4,76 %	u	4,35 %
i	7,55 %	v	0,67 %
j	0,27 %	w	1,89 %
k	1,21 %	x	0,03 %
l	3,44 %	y	0,04 %
m	2,53 %	z	1,13 %

2. Statistische Analyse
Im Deutschen, wie in jeder natürlichen Sprache, kommen die Buchstaben nicht gleich häufig vor; vielmehr hat jeder Buchstabe eine charakteristische Häufigkeit.

Diese Häufigkeiten sind in Tab. 1.2 aufgelistet. Bei der Erstellung dieser Tabelle haben wir Leer- und Satzzeichen nicht berücksichtigt; die Umlaute ä, ö, ü wurden wie ae, oe, ue behandelt.

Solche Tabellen erhält man, indem man lange und verschiedenartige Texte auszählt. Es ist bemerkenswert, wie wenig die Häufigkeiten variieren (siehe Übungsaufgabe 10). Natürlich gibt es erklärbare Differenzen: Ein von Zitaten strotzender zoologischer Text über den Einfluss von Ozon auf die Zebras im Zentrum von Zaire wird eine andere Häufigkeitsverteilung haben als ein Traktat über die amourösen Aventüren des Balthasar Matzbach am Rande des Panamakanals.

Wir können die Buchstaben gemäß der Häufigkeit ihres Auftretens in vier Gruppen einteilen (siehe Tab. 1.3). In der ersten Gruppe sind die beiden häufigsten Buchstaben e und n, in der zweiten befinden sich die Buchstaben, deren Häufigkeit noch relativ groß, etwa 7 %, ist; in der dritten Gruppe sind die Buchstaben zusammengefasst, die eine kleine, aber noch merkliche Häufigkeit haben, während in der letzten Gruppe die vernachlässigbaren Buchstaben aufgeführt sind.

Was passiert nun, wenn wir einen (deutschen) Klartext chiffrieren? Nun, dann bleibt die Häufigkeitsverteilung der Buchstaben erhalten; allerdings sind

Tab. 1.3 Buchstabengruppen

Gruppe	Gesamthäufigkeit dieser Buchstaben
e, n	27,18 %
i, s, r, a, t	34,48 %
d, h, u, l, c, g, m, o, b, w, f, k, z	36,52 %
p, v, j, y, x, q	1,82 %

die einzelnen Häufigkeiten nicht mehr ihren „angestammten" Buchstaben zugeordnet. Zum Beispiel: Wenn der Buchstabe e in X chiffriert wird, so wird X der häufigste Geheimtextbuchstabe sein; wenn andererseits y mit S verschlüsselt wird, so wird S im Geheimtext fast nicht vorkommen.

Konkret geht Mr. X bei einer Kryptoanalyse so vor: Zunächst zählt er per Strichliste, wie oft die einzelnen Buchstaben im Geheimtext enthalten sind. In unserem Beispiel ergibt sich:

Buchstabe:	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Häufigkeit:	2	2	4	0	0	0	1	0	0	0	0	1	1	6	0	1	3	4	0	0	0	2	1	0	0	0.

Der Buchstabe mit der größten Häufigkeit ist N; also wird Mr. X vermuten, dass N der dem Klartextbuchstaben e entsprechende Buchstabe ist.

Achtung: Da alles auf Statistik beruht, ist nichts 100 %-ig sicher! Mr. X muss nach Bestätigungen seiner Vermutung Ausschau halten. Wenn seine Vermutung richtig ist, so liegt eine Verschiebung um 9 Stellen vor. Dann müsste R dem Klartextbuchstaben i entsprechen (das bestätigt seine Hypothese, da R relativ häufig vorkommt), und W müsste das Geheimtextäquivalent von n sein – das stimmt ihn bedenklich, da W nur einmal auftaucht. Andererseits kommen A, B, C relativ häufig vor (sie müssten r, s, t entsprechen), und die Äquivalente von x, y, z (nämlich G, H, I) treten praktisch nicht in Erscheinung.

Also wird Mr. X der nächstliegenden Vermutung nachgehen und so zu entschlüsseln, wie er zunächst vermutet hatte. Er erhält einen sinnvollen Klartext, und damit ist seine Vermutung endgültig bestätigt.

Algorithmus 1.5: Statistische Analyse der Cäsar-Chiffre

Man bestimmt den häufigsten Buchstaben im Geheimtext und stellt das Geheimtextalphabet so ein, dass dieser Buchstabe dem Klartext-e entspricht.

Dann entschlüsselt man mit dieser Einstellung.

Hier sind einige Bemerkungen angebracht. Das zweite Verfahren hat den unbestreitbaren Vorteil, dass es von einem Computer leicht allein durchgeführt werden kann. Da hier aber statistische Überlegungen die entscheidende Rolle spielen, muss man eine gewisse Vorsicht walten lassen. Insbesondere bei kurzen Texten kann die naive Suche nach dem häufigsten Buchstaben auf einen Holzweg führen. Wenn man aber noch die Häufigkeiten einiger anderer Buchstaben in Rechnung stellt, kann man Verfahren entwickeln, die auch bei sehr kurzen Texten erfolgreich sind (vergleiche Übungsaufgabe 17). Auf alle Fälle ist meine persönliche Erfahrung, dass ich bei der praktischen Kryptoanalyse begonnen habe, an die Verlässlichkeit von Statistik zu glauben.

1.4 Monoalphabetische Chiffrierungen

Eine Chiffrierung heißt *monoalphabetisch*, falls jeder Buchstabe des Alphabets stets zu demselben (Geheimtext-)Buchstaben chiffriert wird. Eine monoalphabetische Chiffrierung kann man also immer so darstellen, dass man unter das „Klartextalphabet“ ein „Geheimtextalphabet“ schreibt. Neben den Verschiebechiffren stellen die folgenden Chiffriermethoden monoalphabetische Chiffrierungen dar:

Klartextalph.:	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Geh.alphabet:	Q	W	E	R	T	Z	U	I	O	P	A	S	D	F	G	H	J	K	L	Y	X	C	V	B	N	M

Klartextalph.:	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Geh.alphabet:	#	∀	∃	&	ɜ	<	>	∞	♣	♦	♥	♠	±	≥	∂	≠	≡	≈	↵	ℵ	⊗	⊕	∅	∇	∏	√

Das letzte ∀ɜ♣↵≠♣ɜ♠ soll Sie nur daran erinnern, dass Klartext und Geheimtext nicht über demselben Alphabet definiert sein müssen. Ist dies jedoch der Fall, so entspricht jeder monoalphabetischen Chiffrierung eine Permutation der Buchstaben des Alphabets; umgekehrt kann man jeder Permutation der Buchstaben eine monoalphabetische Chiffrierung zuordnen.

Eine *Permutation* ist eine beliebige „Umordnung“ der Elemente einer Menge (Mathematiker sprechen von einer „bijektiven Abbildung“.) In obigem Fall ist die Umordnung die folgende: Der Buchstabe a kommt an die Stelle, an der vorher k stand, der Buchstabe b an die Stelle x, usw.

Eine n-elementige Menge hat genau $n! = n \cdot (n-1) \cdot \ldots \cdot 2 \cdot 1$ Permutationen (sprich: „n Fakultät“). Daher gibt es genau

$$26! = 26 \cdot 25 \cdot \ldots \cdot 2 \cdot 1 = 403.291\,461.126\,605.635\,584.000\,000,$$

also etwa $4 \cdot 10^{26}$ monoalphabetische Chiffrierungen über dem natürlichen Alphabet {a, b, c, . . . , z}.

Algorithmus 1.6: Monoalphabetische Verschlüsselung

Schlüssel:	Das Geheimtextalphabet ist eine beliebige Permutation des Alphabets.
Chiffrieren:	Ein Buchstabe des Klartextalphabets wird verschlüsselt, indem er durch den unmittelbar darunter stehenden Buchstaben des Geheimtextalphabets ersetzt wird.
Dechiffrieren:	Ein Buchstabe des Geheimtextalphabets wird verschlüsselt, indem er durch den unmittelbar darüber stehenden Buchstaben des Klartextalphabets ersetzt wird

In diesem Kapitel werden wir diese riesige Menge von Chiffrieralgorithmen studieren. Zunächst werden wir einige weitere Beispiele betrachten, um uns dann klar zu machen, dass monoalphabetische Chiffrierungen bei weitem nicht so sicher sind, wie ihre beeindruckende Zahl glauben machen möchte.

1.5 Tauschchiffren

Wenn man mit dem Computer verschlüsseln will, so muss man die Buchstaben durch Zahlen darstellen. Man identifiziert üblicherweise a (bzw. A) mit 1, b (bzw. B) mit 2, und so weiter; schließlich identifiziert man x mit 24, y mit 25 und z mit 0.

So kann man eine Verschiebechiffre besonders gut beschreiben: Eine Verschiebung um, sagen wir, t Stellen entspricht nämlich einer Addition der Zahl t. Konkret geht man dabei so vor:

- Zunächst wird der Klartextbuchstabe in die ihm entsprechende Zahl übersetzt,
- dann wird zu dieser Zahl die Zahl t addiert,
- vom Ergebnis betrachten wir nur den Rest, der sich beim Teilen durch 26 ergibt; dieser Rest wird wieder in einen Buchstaben zurückübersetzt. Man sagt dazu auch, man rechnet „modulo 26“.

So erhält man den zugehörigen Geheimtextbuchstaben.

Beispiel: Wir wollen den Klartextbuchstaben a mit einer Verschiebechiffre, die um 3 Stellen verschiebt, chiffrieren.

- a wird durch die Zahl 1 dargestellt,
- $1 + 3 = 4$,
- 4 ist die Darstellung des Geheimtextbuchstabens D.

Bei der Dechiffrierung von B geht man so vor:

- B entspricht der Zahl 2,
- $2 - 3 = -1$,
- der Rest von -1 bei Division durch 26 ist 25; dieser Rest entspricht dem Klartextbuchstaben y.

Mit dieser Methode kann man Buchstaben „addieren". Interessanter wird die Sache, wenn wir Buchstaben multiplizieren. Dies geschieht auf folgende Weise:

Um einen Buchstaben mit einer Zahl s zu multiplizieren, rechnen wir wieder modulo 26. Das heißt, wir multiplizieren die dem Buchstaben entsprechende Zahl mit s und betrachten den Rest bei Division durch 26. Dann ist der diesem Rest entsprechende Buchstabe das Ergebnis dieser „Multiplikation".

Wenn man den Wert eines jeden Klartextbuchstaben mit der Zahl 2 multipliziert, so erhält man

Klartextalph.:	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Geh.alphabet:	B	D	F	H	J	L	N	P	R	T	V	X	Z	B	D	F	H	J	L	N	P	R	T	V	X	Z

Wir sehen, dass jeweils zwei Buchstaben (zum Beispiel h und u) dasselbe „Produkt" (in unserem Beispiel P) ergeben. Daher können wir diese Substitution nicht als Chiffre verwenden. Für jede Chiffrierung muss nämlich die bislang zwar unausgesprochene, aber selbstverständliche Regel gelten, dass der Klartext mit Hilfe des Schlüssels eindeutig aus dem Geheimtext rekonstruierbar sein muss. Manche halten diese Regel für zu restriktiv; man kann sie aber abschwächen und zugleich begründen: Jeder Computer soll den Geheimtext mit Hilfe des Schlüssels dechiffrieren können!

Jetzt versuchen wir unser Glück nochmals und multiplizieren jeden Buchstaben mit der Zahl 3. Es ergibt sich

Klartextalph.:	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Geh.alphabet:	C	F	I	L	O	R	U	X	A	D	G	J	M	P	S	V	Y	B	E	H	K	N	Q	T	W	Z

In diesem Fall erhalten wir tatsächlich eine monoalphabetische Chiffrierung. Durch Ausprobieren (vergleiche Übungsaufgabe 21) sieht man ohne Mühe, dass man genau dann eine monoalphabetische Chiffrierung erhält,

wenn man mit einer der Zahlen 1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23 oder 25 multipliziert; die so erhaltenen Chiffrierungen nennt man *multiplikative* Chiffren.

Es gibt (inklusive der trivialen Chiffre) also genau 12 multiplikative Chiffrierungen; ihre Anzahl ist also noch kleiner als die der Verschiebechiffren. Daher werden wir von ihnen nur eine äußerst geringe kryptographische Sicherheit erwarten.

Man kann aber Verschiebechiffren und multiplikative Chiffren miteinander kombinieren. Dazu multiplizieren wir zunächst den Klartextbuchstaben mit einer Zahl s, fassen das Ergebnis wieder als Klartextbuchstaben auf und addieren dazu dann eine weitere Zahl t. Man erhält nach dieser Vorschrift wieder eine Chiffre, eine sogenannte *Tauschchiffre*, die wir mit [s, t] bezeichnen.

Der *Schlüssel* der Tauschchiffre [s, t] besteht aus dem Zahlenpaar (s, t). Natürlich muss bei jeder Tauschchiffre die Zahl s so gewählt sein, dass die Multiplikation mit s eine multiplikative Chiffre ist; s muss also eine der oben genannten Zahlen 1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25 sein.

Algorithmus 1.7: Tausch-Chiffre

Schlüssel:	Ein Paar (s, t) von natürlichen Zahlen ≤ 26, wobei s teilerfremd zu 26 ist.
Chiffrieren:	Sei x die Nummer des Klartextbuchstaben, dann ist $xs + t \bmod 26$ die Nummer des Geheimtextbuchstabens. Dabei ist a mod 26 diejenige Zahl, die sich als Rest bei der Division von a durch 26 ergibt.
Dechiffrieren:	Sei y die Nummer des Geheimtextbuchstabens. Dann ist $(y - t)s' \bmod 26$ die Nummer des Klartextbuchstabens, wobei s' die Zahl ist mit $s's \bmod 26 = 1$.

Die Anzahl der Tauschchiffren errechnet sich als die Anzahl aller Verschiebechiffren mal der Anzahl aller multiplikativen Chiffren; die Anzahl der Tauschchiffren ist demgemäß gleich $26 \cdot 12 = 312$. Diese Zahl ist schon so groß, dass man sich bei einer Kryptoanalyse „von Hand“ nicht ganz leicht tut, wenn man alle Möglichkeiten systematisch durchprobieren will. (Siehe aber die Übungsaufgaben 15, 16, 17.) Wir geben zur Lösung dieses Problems keinen speziellen Algorithmus an, da wir im übernächsten Abschnitt ohnedies alle monoalphabetischen Chiffrierungen analysieren werden.

1.6 Schlüsselwörter

Auf folgende Art und Weise erhält man eine riesige Menge von monoalphabetischen Chiffren: Der Schlüssel besteht aus zwei Komponenten, einem Schlüsselwort und einem Schlüsselbuchstaben. Zunächst macht man aus dem Schlüsselwort eine Buchstabenfolge, in der jeder Buchstabe nur einmal vorkommt. Dies wird dadurch erreicht, dass jeder Buchstabe bei seinem zweiten, dritten, … Auftreten gestrichen wird. Haben wir beispielsweise das Schlüsselwort

GEHEIMSCHRIFT

gewählt, so ergibt sich die Folge

GEHIMSCRFT.

Nun schreibt man diese Folge unter das Klartextalphabet, und zwar so, dass man genau unter dem Schlüsselbuchstaben beginnt. Haben wir in unserem Beispiel als Schlüsselbuchstaben e gewählt, so erhalten wir

Klartextalph.:	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Geh.alphabet:					G	E	H	I	M	S	C	R	F	T												

Anschließend schreibt man die restlichen Geheimtextbuchstaben in alphabetischer Reihenfolge auf, indem man nach dem letzten Schlüsselwortbuchstaben beginnt. In unserem Beispiel ergibt sich:

Klartextalph.:	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Geh.alphabet:	W	X	Y	Z	G	E	H	I	M	S	C	R	F	T	A	B	D	J	K	L	N	O	P	Q	U	V

Algorithmus 1.8: Schlüsselwortchiffre

Schlüssel:	Ein Wort. Doppelte Buchstaben werden beim zweiten oder späteren Auftreten entfernt. Dieses Wort wird als Anfang des Geheimtextalphabets benutzt; dieses wird dann durch die restlichen Buchstaben in natürlicher Reihenfolge aufgefüllt.
Variante:	Der Schlüssel besteht zusätzlich aus einem Buchstaben. Das Schlüsselwort wird beginnend unter diesen *Schlüsselbuchstaben* geschrieben.
Chiffrieren:	Ein Buchstabe des Klartextalphabets wird verschlüsselt, indem er durch den unmittelbar darunter stehenden Buchstaben ersetzt wird.

Dechiffrieren: Ein Buchstabe des Geheimtextalphabets wird verschlüsselt, indem er durch den unmittelbar darüber stehenden Buchstaben ersetzt wird.

Da man die Anzahl der Schlüsselwörter nicht genau angeben kann, kann man auch die Anzahl dieser Chiffrierungen nicht präzise bestimmen. Klar ist jedoch, dass ihre Zahl sehr groß ist. Formal könnte man sogar sagen, dass man jede monoalphabetische Chiffrierung mit Hilfe eines Schlüsselwortes darstellen kann – dieses wird allerdings in der Regel sehr seltsam aussehen.

1.7 Kryptoanalyse

Die „Philosophie" der modernen Kryptoanalyse wird durch das *Prinzip von Kerckhoffs* beschrieben; dies wurde erstmals in dem Buch *La cryptographie militaire* (1883) des niederländischen Philologen Jean Guillaume Hubert Victor François Alexandre Auguste Kerckhoffs von Nieuwenhof (1835 bis 1903) formuliert, wie sein Name in vollem barockem Glanze lautet.

Das Prinzip von Kerckhoffs

Die Sicherheit eines Kryptosystems darf nicht von der Geheimhaltung des Algorithmus abhängen. Die Sicherheit gründet sich nur auf die Geheimhaltung des Schlüssels.

Dies ist eine grundsätzliche Warnung an den Entwickler eines Kryptosystems. Er darf auf keinen Fall so blauäugig sein anzunehmen, Mr. X hätte keine Möglichkeit, Kenntnis des Algorithmus zu erlangen. Die Geschichte der Kryptographie ist voll von Beispielen, wo der Erfinder eines Kryptosystems sein Vertrauen darauf gründete, dass sein Algorithmus nie bekannt werden könnte.

Im Gegenteil: Das Ziel der modernen Kryptographie muss es sein, Systeme zu entwickeln, die auch dann noch sicher bleiben, wenn der Algorithmus lange Zeit öffentlich diskutiert wurde. Hervorragende Beispiele dafür sind der DES- und der AES-Algorithmus (vergleichen Sie dazu Übungsaufgabe 25).

Manche Experten gehen noch viel weiter und fordern, dass man nur solche Verschlüsselungsalgorithmen einsetzen sollte, die veröffentlicht und längere Zeit öffentlich diskutiert wurden. Dieses Argument hat etwas für sich, denn nach wie vor gilt für die meisten Algorithmen: Die Praxis ist nicht nur der härteste, sondern auch der beste Test!

Ein Kryptoanalytiker sieht sich von Fall zu Fall unterschiedlichen Voraussetzungen gegenüber. Man unterscheidet folgende Typen von Attacken:

- *Angriff mit bekanntem Geheimtext (known ciphertext attack)*
  Der Kryptoanalytiker kennt ein relativ großes Stück Geheimtext. Dies ist eine außerordentlich realistische Annahme, da es in der Regel ganz einfach ist, sich (beliebig lange) Stücke Geheimtext zu verschaffen.
- *Angriff mit bekanntem Klartext (known plaintext attack)*
  Der Kryptoanalytiker kennt ein (vergleichsweise kleines) Stück von zusammengehörigem Klartext/Geheimtext. Diese Hypothese ist realistischer als sie auf den ersten Blick erscheint. Denn oft „weiß" Mr. X, worum es geht, und kann also zumindest einige Schlagworte erraten. Außerdem finden sich in der Regel standardisierte Eröffnungs- und Schlussfloskeln, und so weiter.
- *Angriff mit frei gewähltem Klartext (chosen plaintext attack)*
  Wenn der Kryptoanalytiker Zugang zum Verschlüsselungsalgorithmus (mit dem aktuellen Schlüssel!) hat, so kann er, um den Schlüssel zu erschließen, auch selbstgewählte Stücke Klartext verschlüsseln und versuchen, aus dem erhaltenen Geheimtext Rückschlüsse auf die Struktur des Schlüssels zu ziehen. Er könnte etwa der Maschine besonders regelmäßige Klartexte unterjubeln, zum Beispiel Folgen aus demselben Buchstaben (aaa . . .), mit dem Auftrag, diese zu verschlüsseln.

Die Gefahr eines solchen Angriffs besteht darin, dass es gelingen könnte, das Verschlüsselungsgerät dazu zu bringen, scheinbar harmlose Nachrichten zu verschlüsseln, mit Hilfe derer dann Mr. X eine Nachricht verschlüsseln kann, die der Sender nie verschlüsselt hätte. Wie gefährlich dieser Angriff ist, wird besonders deutlich, wenn man daran denkt, dass manche Kryptogeräte nicht nur verschlüsseln, sondern auch „unterschreiben" können (siehe Kap. 5): Wenn der Algorithmus so schwach ist, dass er diesen Angriff zulässt, so könnte Mr. X unter diesen Umständen aus harmlos erscheinenden unterschriebenen Nachrichten ein brisantes, gültig unterschriebenes Dokument komponieren!

Jede monoalphabetische Chiffrierung einer natürlichen Sprache kann ziemlich leicht geknackt werden. Wir werden uns nun klarmachen, dass jede monoalphabetische Chiffrierung (einer natürlichen Sprache!) schon unter der äußerst schwachen (weil äußerst realistischen) Annahme einer known ciphertext attack zu knacken ist.

Wir werden hier allerdings nur einen „prinzipiellen" Algorithmus vorstellen. Sein Ziel ist es, Sie zu überzeugen, dass monoalphabetische Chiffriermethoden äußerst unsicher sind. Das Verfahren beruht auf der Tatsache, dass die _ehn haeu_i_sten _u_hsta_en i_ deuts_hen _ereits drei _ierte_ des _esa_tte_tes _i_den.

Tab. 1.4 Die häufigsten Bigramme der deutschen Sprache

Bigramm	Häufigkeit	Bigramm	Häufigkeit
en	3,88 %	nd	1,99 %
er	3,75 %	ei	1,88 %
ch	2,75 %	ie	1,79 %
te	2,26 %	in	1,67 %
de	2,00 %	es	1,52 %

Stellen wir uns vor, dass Mr. X einen Geheimtext von etwa 500 Buchstaben abgefangen hat, von dem er weiß, oder auch nur vermutet, dass er mit Hilfe einer monoalphabetischen Chiffrierung verschlüsselt wurde.

Schritt 1 Zunächst stellt Mr. X die Häufigkeiten der Buchstaben des Geheimtextes fest. Dadurch kann er das Äquivalent von e, das von n, sowie die nächst häufigsten Buchstaben i, s, r, a, t feststellen. Die einzelnen Buchstaben kann er dabei in der Regel noch nicht identifizieren; aber er weiß, welche Buchstaben der Menge der Klartextbuchstaben i, s, r, a, t entsprechen.

Schritt 2 Nun zählt Mr. X die *Bigramme*, das heißt Paare aufeinander folgender Buchstaben. Die häufigsten Bigramme der deutschen Sprache sind in Tab. 1.4 aufgelistet.

Damit kann er nun die nächsthäufigsten Buchstaben isolieren. Zum Beispiel hat das Paar er eine sehr große Häufigkeit, während alle anderen Kombinationen der kritischen Buchstaben mit e ziemlich selten vorkommen (ea und et sind wirklich sehr selten (unter 0,5 %), und auch es kommt mit signifikant geringerer Häufigkeit vor). Das Paar ei könnte eine Konkurrenz sein; diese kann man aber dadurch ausschalten, dass man das inverse Paar testet: Nur bei diesem Paar ist es so, dass es sowohl in Originalreihenfolge als auch in umgekehrter Reihenfolge praktisch gleich häufig vorkommt. Damit kann Mr. X die zunächst ununterscheidbaren zweithäufigsten Buchstaben i, s, r, a, t isolieren. Ferner kann er die Buchstaben c und h daran erkennen, dass sie zwar ein extrem häufiges Paar bilden, aber einzeln betrachtet nur ganz selten auftauchen.

So kann er viele der häufigsten Buchstaben ziemlich zweifelsfrei identifizieren; dies sind die Buchstaben e, n, i, s, r, a, t, h, c, die zusammen mehr als zwei Drittel eines Textes ausmachen. Wir halten fest, dass dies völlig automatisch von einem Computer durchgeführt werden kann!

Schritt 3 Nun lässt Mr. X den Computer die erkannten Buchstaben im gesamten Text übersetzen. Mit anderen Worten: Der Computer entschlüsselt die

bekannten Teile des Textes. Dieser wird auf dem Bildschirm angezeigt, wobei die nicht entschlüsselten Buchstaben zweckmäßigerweise durch Leerzeichen ersetzt werden.

In aller Regel ist dieser Text noch nicht, oder jedenfalls nur äußerst mühsam zu entziffern. Weitere Buchstaben kann der intelligente Mr. X dann leicht raten! Dies tut er auch und lässt sich den Text jeweils probeweise mit den geratenen zusätzlichen Identifikationen anzeigen. Nach zwei oder drei Schritten ist er zu einem sehr gut lesbaren Text gelangt.

Algorithmus 1.9: Kryptoanalyse einer monoalphabetischen Chiffre über dem natürlichen Alphabet

Durch eine Häufigkeitsanalyse bestimmt man die Geheimtextäquivalente von e und n.

Eventuell kann man hier schon weitere Buchstaben erfolgreich raten.

Durch die Häufigkeitsanalyse der Bigramme kann man weitere Buchstaben bestimmen.

Jetzt kann man so viele Buchstaben raten, dass der Text entzifferbar ist.

Wir fassen zusammen: Monoalphabetische Chiffrierungen über natürlichen Sprachen sind bemerkenswert unsicher. (Eine natürliche Sprache hat wenig Buchstaben, die ziemlich ungleichmäßig verteilt sind.) Heutzutage verwendet man deshalb entweder monoalphabetische Chiffrierungen über nichtnatürlichen Sprachen oder polyalphabetische Chiffrierungen.

1.8 Moderne monoalphabetische Algorithmen

Die populärste monoalphabetische Chiffrierung ist der *DES*, der *Data Encryption Standard*. Dieser Algorithmus wurde im Wesentlichen von IBM entwickelt und 1977 standardisiert. (Für eine Geschichte dieses Algorithmus siehe [FR94].) Der DES verschlüsselt nicht Buchstaben, sondern die Symbole 0 und 1, und zwar jeweils 64 auf einen Streich. (Wird der DES benutzt, um gewöhnlichen Text zu chiffrieren, müssen die Buchstaben zuvor in Bitfolgen übersetzt werden. Eine Methode, dies zu tun, ist der sogenannte ASCII-Code, den wir in Kap. 5 kennen lernen werden.) Es stellt sich heraus, dass der DES ein monoalphabetischer Algorithmus über dem „Alphabet“ $\{(a_1, \ldots, a_{64}) \mid a_i \in \{0,1\}\}$ ist. Das heißt, dass die Elemente des Klar- und des Geheimtextalphabets die binären Folgen der Länge 64, die sogenannten *Blöcke* sind. Man spricht auch von einer *Blockchiffre*. Die Schlüssel sind die binären Folgen aus 56 Bits; daher gibt es genau $2^{56} \approx 7 \cdot 10^{16}$ verschiedene Schlüssel.

Der DES-Algorithmus wurde von Anfang an vollständig publiziert – es war der erste Algorithmus der Geschichte, bei dem das der Fall war. In Übungsaufgabe 25 ist ein Teil des Textes zitiert. Das gesamte Dokument ist in [BP82] abgedruckt. Der DES wurde vor allem im Bankenbereich erfolgreich eingesetzt. Der im Jahre 1990 vorgestellte Angriff von Eli Biham und Adi Shamir [BS91] vermittelt die Einsicht, dass der DES ein nach hervorragenden Prinzipien entwickelter Algorithmus, ja ein sehr guter Algorithmus ist. Die Zielrichtung der Arbeit von Biham und Shamir ist eine Analyse des DES. Aber eines ihrer Ergebnis ist, dass viele DES-ähnlichen Algorithmen durch diese geistreiche Analyse erledigt sind, nicht aber der DES selbst!

Allerdings hatten seit Veröffentlichung des DES viele Wissenschaftler vor der ihrer Meinung nach zu kurzen Schlüssellänge gewarnt. Und sie hatten Recht! Im Frühjahr 1999 wurde der DES durch eine vollständige Schlüsselsuche gebrochen. Das kann man sich so vorstellen: Es war ein DES-verschlüsselter Geheimtext gegeben. Auf diesen wurden solange mögliche Schlüssel angewandt, bis sich bei der Entschlüsselung ein sinnvoller Klartext ergab. Die ganze Prozedur dauerte damals gut 22 Stunden; heute würde man das noch viel schneller schaffen. Aber Achtung! Der DES wurde nicht geknackt, weil man eine interne Schwäche gefunden und diese ausgenützt hätte, sondern nur dadurch, dass man die stumpfsinnigste denkbare Attacke ausgeführt hat, nämlich das systematische Durchprobieren der Schlüssel. Eine Konsequenz daraus ist, dass jeder Algorithmus mit einer Schlüssellänge von 56 Bit oder weniger so geknackt werden kann. Solche Verfahren sollte man grundsätzlich nicht verwenden!

Was tun? Die Forscher haben in zwei Richtungen gearbeitet. Zum einen haben sie den DES verbessert und aus ihm den Triple-DES gemacht, zum andern haben sie neue Algorithmen entwickelt. Über beides soll noch kurz berichtet werden.

Beim *Triple-DES* verwendet man insgesamt zwei DES-Schlüssel und wendet den DES-Algorithmus dreimal an. Der Trick des Triple-DES besteht darin, den Klartext zunächst unter dem ersten Schlüssel k_1 mit DES zu verschlüsseln, dann das Ergebnis mit Hilfe des zweiten Schlüssels k_2 zu entschlüsseln und schließlich dieses Ergebnis mit wieder Hilfe der ersten Schlüssels k_1 zu verschlüsseln. In einer Formel sieht das so aus:

$$c = DES(DES^{-1}(DES(m, k_1), k_2), k_1).$$

Wundern Sie sich nicht: es geht nicht einfacher. Der gesamte Schlüssel besteht aus 112 Bits, und diese können auf absehbare Zeit nicht mit Hilfe einer systematischen Attacke ausprobiert werden. Der Triple-DES ist bis heute ein häufig eingesetzter Verschlüsselungsalgorithmus.

Im Herbst 2000 hat das amerikanische Standardisierungsinstitut NIST einen neuen Algorithmus angekündigt, den AES (Advanced Encryption Standard, siehe [DR01]). Dieser ist wie der DES eine Blockchiffre, unterscheidet sich aber ansonsten grundlegend vom DES: Die Schlüssellänge des AES ist 128 Bit und kann sogar bis 256 Bit gesteigert werden. Er operiert auf Blöcken von je 128 Bit.

Der AES ist außerordentlich klar strukturiert. Die Daten werden in Bytes eingeteilt, und diese sind in Spalten zu je vier Bytes organisiert. Damit wird sowohl die 8 Bit-Struktur von Chipkartenchips als auch die 32 Bit Architektur von PCs unterstützt. Der AES arbeitet in Runden, und jede Runde besteht aus einer Substitution, bei der jedes Byte durch ein anderes ersetzt wird, einer Verschiebung der Zeilen, einer Transformation der Spalten und der Addition des Rundenschlüssels. Jede einzelne Operation kann mathematisch genau erfasst werden; man hofft, durch die Mischung der vier inkompatiblen Operationen eine hohe Komplexität zu erhalten.

Die durchsichtige mathematische Struktur wird unterschiedlich bewertet: Während die einen sagen, dass man nur so eine Chance hat, die Sicherheit des Algorithmus zu verifizieren, warnen die anderen, dass die mathematische Einfachheit prinzipiell auch die Möglichkeit von Angriffen eröffnet. In jedem Fall muss man aber anerkennen, dass beim AES – im Gegensatz zum DES – die Designkriterien von vornherein offen gelegt wurden, und nicht nur eine Implementierungsvorschrift bekannt gegeben wurde.

1.9 Übungsaufgaben

1 Wie lautet der Klartext, der zu folgendem, mittels einer Skytala chiffrierten Geheimtext gehört?
I S A D T P I H E H N N C S D I R O O I L T A I H T A E A S
N F E Z C S W E S S N I S F I U K T U J S E S T C R C K E !

2 Basteln Sie sich eine Skytala. [Anstelle eines Zylinders kann man zum Dechiffrieren auch ein ebenes steifes Stück Pappe benutzen.]
   (a) Probieren Sie die Skytala aus.
   (b) Beschreiben Sie diese Chiffriermethode, indem Sie die Wörter „Algorithmus“ und „Schlüssel“ benutzen.
   (c) Wie groß ist ungefähr die Anzahl der Schlüssel?
   (d) Beurteilen Sie die Sicherheit dieses Algorithmus!

3 Beschreiben die Permutation, durch die man die Skytala-Chiffre aus Abschn. 1.1 (d. h. ein Text aus 4 Buchstaben wird spaltenweise in fünf Zeilen geschrieben) darstellen kann. („Der erste Buchstabe bleibt fest, der zweite

wird an die Stelle hmhm geschrieben, der dritte an die Stelle hmhmhm, …“)

4 Wenn man den Skytala-Algorithmus durchführt, geht er in der Regel nicht auf. Das heißt: In dem Schema mit u Zeilen bleiben am Ende einige Plätze leer. Man könnte diese Plätze mit auffüllen, indem man dort überall den Buchstaben x hinschreibt. Ist das gut oder wäre das ein wertvoller Hinweis für Mr. X?

5 Zeigen Sie, dass es genau n! $(= n \cdot (n-1) \cdot \ldots \cdot 2 \cdot 1)$ Permutationen einer n-elementigen Menge gibt.

6 Die *Gartenzaun-Chiffre* kann am besten durch ein Beispiel beschrieben werden:

Klartext:	I			A			N			E			G			T			S		
		S			D			S			I			G			T			R			L			O			I			H			U			?
			T				E				U				A				R				M															

Geheimtext: I A N E G T S S D S I G T R L O I H U ? T E U A R M.

(a) Beschreiben Sie diesen Algorithmus in Worten.
(b) Ist dies eine Transpositions- oder eine Substitutionschiffre?

7 Und Cäsar sprach: SBKF SFAF SFZF.

8 Basteln Sie aus zwei Papierscheiben die in Abb. 1.3 dargestellte Chiffriermaschine.

9 Warum nennt man die Verschiebechiffren auch „additive“ Chiffren?

10 Bestimmen Sie folgende Zahlen: 30 mod 26, 60 mod 26, 90 mod 26, −10 mod 26.

11 @Erstellen Sie ein Programm, das eine monoalphabetische Chiffre realisiert. Ein einzugebender Klartext soll mittels eines vom Benutzer zu wählenden Schlüssels chiffriert werden.

12 *Suchen Sie (sinnvolle) deutsche Wörter, die mittels einer Verschiebechiffre auseinander hervorgehen. [Hinweis: Bereits Paare aus Wörtern mit 3 Buchstaben sind eine Leistung!]

13 Ermitteln Sie die Häufigkeiten der Buchstaben in mindestens fünf verschiedenen deutschen Texten mit 1000 Buchstaben.

14 Suchen Sie einen Klartext von mindestens 50 Buchstaben, bei dem e nicht der häufigste Buchstabe ist. [Das ist zu hart für mich als Autor!]

Zusatzinformation: Es ist kaum zu glauben, aber es gibt eine alte literarische Tradition, deren Ziel es ist, Texte ohne einen bestimmten, vorher gewählten Buchstaben zu verfassen. Solche Texte werden *lipogrammatisch* (auch *leipogrammatisch*) genannt. Der erste lipogrammatische Schriftsteller soll der Grie-

che Lasos von Achaia gewesen sein, der im sechsten vorchristlichen Jahrhundert lebte. Die lipogrammatische Idee war fast immer in der Geschichte eine Herausforderung an die Schriftsteller; so gab es zum Beispiel Versuche, die 24 Gesänge von Homers Ilias lipogrammatisch umzudichten: den ersten Gesang ohne den Buchstaben α, den zweiten ohne β, ... und den letzten schließlich ohne ω.

Ein Höhepunkt lipogrammatischer Literatur ist zweifellos der 1969 erschienene Roman *La Disparition* des französischen Schriftstellers George Perec: Keines seiner etwa 85.000 Wörter enthält den Buchstaben e. Mindestens genauso bewundernswert ist, dass dieser Roman lipogrammatisch übersetzt wurde: Unter dem Titel *Anton Voyls Fortgang* hat Eugen Helmle dieses Kunststück vollbracht. (Ist es Zufall, dass sowohl der Name des Autors als auch der seines Übersetzers vor e's geradezu strotzt?) Der interessierte Leser sei auf das äußerst lehrreiche Nachwort von [Per86] hingewiesen.

15 Zählen Sie die Bigramme in mindestens zwei verschiedenen Texten, die aus je 1000 Buchstaben bestehen.

16 @Entwerfen Sie ein Programm, das die Häufigkeiten von Buchstaben eines einzugebenden Textes bestimmt.

17 @Entwerfen Sie einen Algorithmus, der einen mittels einer Verschiebechiffre verschlüsselten Geheimtext von 30 Buchstaben sicher knackt, wenn er nur die Buchstabenhäufigkeiten kennt.

18 (a) Dechiffrieren Sie W K D U I Q. Benützen Sie die Tatsache, dass eine Tauschchiffre [7,t] benutzt wurde und der Klartext ein deutscher Mädchenname ist.

(b) Dechiffrieren Sie P Z U V F H. Benützen Sie die Tatsache, dass eine Tauschchiffre [s,3] benutzt wurde und der Klartext ein deutscher Mädchenname ist.

19 (a) Zeigen Sie, dass bei jeder multiplikativen Chiffrierung m auf M und z auf Z abgebildet wird.

(b) Durch wie viele korrespondierende Klartext/Geheimtext-Buchstabenpaare ist eine Tauschchiffre eindeutig festgelegt?

20 @Benutzen Sie die Ergebnisse der vorigen Aufgabe, um einen Algorithmus zum Brechen einer Tauschchiffre bei bekanntem Geheimtext zu entwerfen.

21 (a) Genau dann ergibt die Multiplikation mit t eine Chiffrierung, wenn t und 26 teilerfremd sind (d. h., wenn der größte gemeinsame Teiler von t und 26 gleich 1 ist).

(b) Wie viele multiplikative Chiffrierungen besitzt ein Alphabet mit 25 (bzw. 27, bzw. 29) Buchstaben?

22 @Entwerfen Sie einen Algorithmus, der mit Hilfe einer Schlüsselwortchiffre chiffriert. (Planen Sie das Programm so, dass das Schlüsselwort und der Schlüsselbuchstabe vom Benutzer gewählt werden kann.)

23 *Knacken Sie folgenden Geheimtext, von dem ich Ihnen nur verrate, dass er mit Hilfe einer Schlüsselwortchiffrierung mit deutschsprachigem Schlüsselwort erstellt wurde:

YZBY YZBKJYNGJBO ZB XZY MZHHYBHWNUKI LCA
LYGHWNTJYHHYTB, LYGVYGOYB JBX LYGNYZATZWNYB
CNBY UTTY OYNYZABZHRGUYAYGYZ, UVYG BZWNI CNBY
NZBIYGTZHIZOYB HWNUTR, XUGOYHIYTTI SJA BJISYB
JBX YGOCYISYB XYH UTTOYAYZBYB DJVTZRJAH.

24 (a) Was sind besonders häufige bzw. besonders seltene Bigramme im Deutschen? Überlegen Sie zuerst und verifizieren Sie Ihre Überlegungen anschließend.

(b) Verbessern Sie Schritt 2 des Vorschlags für eine Kryptoanalyse eines monoalphabetisch verschlüsselten Textes.

25 Lesen Sie sorgfältig das folgende offizielle Dokument, das der standardisierte Text für den DES ist. Überprüfen Sie, ob die allgemeinen Bemerkungen richtig sind. Wo ist das Prinzip von Kerckhoffs beschrieben?

Federal Information

Processing Standards Publication 46
1977 January 15
ANNOUNCING THE
DATA ENCRYPTION STANDARD

Name of Standard: Data Encryption Standard (DES).
Category of Standard: Operations, Computer Security.

Explanation: The Data Encryption Standard (DES) specifies an algorithm to be implemented in electronic hardware devices and used for the cryptographic protection of computer data. This publication provides a complete description of a mathematical algorithm for encrypting (enciphering) and decrypting (deciphering) binary coded information. Encrypting data converts it to an unintelligible form called cipher. Decrypting cipher converts the data back to its original form. The algorithm described in this standard specifies both enciphering and deciphering operations which are based on a binary number called a key. The key consists of 64 binary digits („0"s or „1"s) of which 56 bits are used directly by the algorithm and 8 bits are used for error detection.

Binary coded data may be cryptographically protected using the DES algorithm in conjunction with a key. Each member of a group of authorized users of encrypted computer data must have the key that was used to encipher the data in order to use it. This key, held by each member in common, is used to decipher the data received in cipher form from other members of the group. The encryption algorithm specified in this standard is commonly known among those using the standard. The unique key chosen for use in a particular application makes the results of encrypting data using the algorithm unique. Selection of a different key causes the cipher that is produced for any given set of inputs to be different. The cryptographic security of the data depends on the security provided for the key used to encipher and decipher the data.

Data can be recovered from cipher only by using exactly the same key used to encipher it. Unauthorized recipients of the cipher who know the algorithm but do not have the correct key cannot derive the original data algorithmically. However, anyone who does have the key and the algorithm can easily decipher the cipher and obtain the original data. A standard algorithm based on a secure key thus provides a basis for exchanging encrypted computer data by issuing the key used to encipher it to those authorized to have the data.

Applications: Data encryption (cryptography) may be utilized in various applications and in various environments. In general, cryptography is used to protect data while it is being communicated between two points or while it is stored in a medium vulnerable to physical theft. Communication security provides protection to data by enciphering it at the transmitting point and deciphering it at the receiving point. File security provides protection to data by enciphering it when it is recorded on a storage medium and deciphering it when it is read back from the storage medium. In the first case, the key must be available at the transmitter and receiver simultaneously during communication. In the second case, the key must be maintained and accessible for the duration of the storage period.

Qualifications: The cryptographic algorithm specified in this standard transforms a 64-bit binary value into a unique 64-bit binary value based on a 56-bit variable. If the complete 64-bit input is used (i. e., none of the input bits should be predeterminated from block to block) and if the 56-bit variable is randomly chosen, no technique other than trying all possible keys using known input and output for the DES will guarantee finding the chosen key. As there are over 70,000,000,000,000,000 (seventy quadrillion) possible keys of 56 bits, the feasibility of deriving a particular key in this way is extremely unlikely in typical threat environments. Moreover, if the key is changed frequently, the risk of this event is greatly diminished. However, users should be aware that it is theoretically possible to derive the key in fewer trials (with a correspondingly lower probability of success depending on the number of keys tried) and should be cautioned to change the key as often as practical. Users must change the key and provide it a high level of protection in order to minimize the potential risks of its unauthorized computation or acquisition.

When correctly implemented and properly used, this standard will provide a high level of cryptographic protection to computer data.

Viele „Geheimschriften“, die (nicht nur) Kinder begeistert ausprobieren, haben das Prinzip, dass die Buchstaben gleich bleiben, aber gemäß mehr oder weniger trickreicher Muster ihre Plätze wechseln. Mit anderen Worten: Es handelt sich um eine Transpositionschiffre. Ein Beispiel soll hier dargestellt werden.

26 Für diese Verschlüsselung brauchen Sender und Empfänger identische Schablonen. Eine solche Schablone ist ein Rechteck, aus dem einige quadratische Löcher ausgeschnitten sind. Der Sender legt die Schablone auf ein Blatt Papier und schreibt seine Nachricht in der „richtigen“ Reihenfolge durch die Löcher, und zwar einen Buchstaben pro Loch. Danach entfernt er die Schablone und füllt die leeren Stellen mit beliebigen Buchstaben auf. Falls seine Nachricht mehr Buchstaben hat als die Schablone Löcher, muss er die Schablone mehrfach anlegen. Die verschlüsselte Nachricht wird dann dem Empfänger geschickt, der sie leicht lesen kann, indem er mit seiner Schablone die verwirrenden Buchstaben ausblendet (vergleiche Abb. 1.5).

(a) Probieren sie diese Verschlüsselungsmethode aus. [Hinweis: Für die Buchstaben zur Verschleierung benutze ich Buchstaben normaler Sätze, verteile sie aber zufällig in dem Rechteck. Dies hat einen kryptologischen Vorteil: Die Buchstaben haben die „richtige“ Häu-

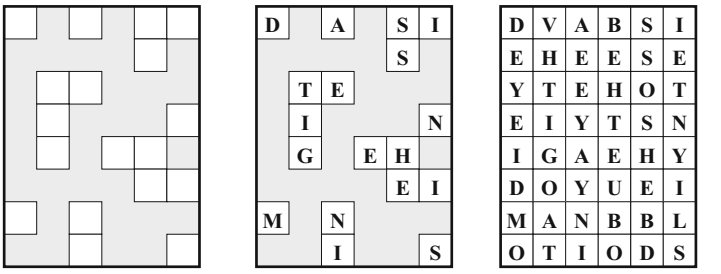


Abb. 1.5 Mit einer Schablone verschlüsseln

figkeitsverteilung. Der andere Vorteil ist psychologischer Art: Sie können auf diese Weise Gedanken formulieren, die später niemand – nicht einmal Sie selbst – wieder lesen können!]

(b) Ist dieser Algorithmus eine Transpositions- oder eine Substitutionschiffre?

(c) Welches sind die Schlüssel? Schätzen Sie die Anzahl der Schlüssel ab!

(d) Unter welchen Umständen hat Mr. X eine Chance, einen so verschlüsselten Text zu knacken?

(e) Entschlüsseln Sie das folgende Kryptogramm:

H	K	A	C	P	P
T	E	B	M	Y	O
U	B	I	H	F	O
R	R	T	S	E	T
E	H	V	D	A	D
L	A	E	Y	Y	T
O	A	Y	S	M	R
H	E	O	C	O	U

27 Lesen Sie Edgar Allan Poe's Erzählung *Der Goldkäfer*. Bei der Lösung dieses Krimis spielt die Analyse einer monoalphabetischen Chiffre eine entscheidende Rolle. Das entscheidende Kryptogramm sieht wie folgt aus. Jedes Zeichen entspricht einem Buchstaben der englischen Sprache.

53‡‡†305))6*;4826)4‡.)4‡);806*;48†8¶60))85;1‡(;:‡
8†83(88)5†;46(;88*96*?;8)*‡(;485);5*†2:*‡(;4956
2(5–4)8¶8*;4069285);)6†8)4‡‡;1(‡9;48081;8:8‡1;4
8†85;4)485†528806*81(‡9;48;(88;4(‡?34;48)4‡;161;:
188;‡?;

2

Wörter und Würmer oder Warum einfach, wenn's auch kompliziert geht?

Ein Wort, ein Satz –: aus Chiffren steigen erkanntes Leben, jäher Sinn (Gottfried Benn).

In diesem Kapitel stehen *polyalphabetischen* Chiffrierungen im Mittelpunkt. Das Ziel bei der Entwicklung solcher Verfahren ist, die Häufigkeiten der Geheimtextbuchstaben möglichst stark anzugleichen, um einem Angreifer die Arbeit möglichst schwer zu machen. Daher wird ein Klartextbuchstabe nicht stets zu demselben Geheimtextbuchstaben verschlüsselt. Eine polyalphabetische Chiffrierung kann also nicht einfach durch ein Klartextalphabet und ein darunter geschriebenes Geheimtextalphabet beschrieben werden.

Die Zuordnung eines Klartextbuchstabens zu einem Geheimtextbuchstaben darf aber auch nicht willkürlich erfolgen. Die *De*chiffrierung muss der strengen Regel der *Eindeutigkeit* genügen; sonst ist keine Dechiffrierung möglich. Anders gesagt: Wenn die Chiffrierung nicht eindeutig wäre, so befände sich der Empfänger prinzipiell in keiner besseren Lage als Mr. X!

Es gibt grundsätzlich zwei Arten, die Häufigkeiten der Geheimtextzeichen einander anzugleichen. Einerseits kann man jedem Klartextbuchstaben nicht nur ein Geheimtextzeichen, sondern eine ganze Menge davon zuordnen. Man spricht von einer *homophonen* Chiffre. Solche Algorithmen werden im folgenden Abschnitt dargestellt. Andererseits kann man aber auch die Geheimtextalphabete wechseln, also viele Geheimtexte im Wechsel verwenden. Die Untersuchung dieser im eigentlichen Sinne polyalphabetischen Verfahren ($\pi o \lambda \upsilon$ (griech.) = viel) wird den größten Teil des Kapitels ausmachen. Wir werden uns im Detail mit der Verschlüsselung nach Vigenère beschäftigen.

A. Beutelspacher, *Kryptologie*, DOI 10.1007/978-3-658-05976-7_2,

Tab. 2.1 Eine homophone Chiffre

Buchstabe	Zugeordnete Zeichen	**Buchstabe**	Zugeordnete Zeichen
a	10 21 52 59 71	**n**	30 35 43 62 63 67 68 72 77 79
b	20 34	**o**	02 05 82
c	28 06 80	**p**	31
d	04 19 70 81 87	**q**	25
e	09 18 33 38 40 42 53 54 55 60 66 75 85 86 92 93 99	**r**	17 36 51 69 74 78 83
f	00 41	**s**	15 26 45 56 61 73 96
g	08 12 97	**t**	13 32 90 91 95 98
h	07 24 47 89	**u**	29 01 58
i	14 39 46 50 65 76 88 94	**v**	37
j	57	**w**	22
k	23	**x**	44
l	16 03 84	**y**	48
m	27 11 49	**z**	64

2.1 Verschleierung der Häufigkeiten

Wie kann man erreichen, dass alle Geheimtextzeichen mit der gleichen Wahrscheinlichkeit auftreten? Ganz einfach: Bei einer *homophonen* Chiffre ordnet die Chiffriervorschrift jedem Buchstaben nicht nur ein Zeichen, sondern eine feste *Menge* von Zeichen (in unserem Beispiel: Ziffernpaare) zu, und zwar so, dass die folgenden Bedingungen erfüllt sind:

- Um das Dechiffrieren eindeutig zu machen, dürfen die Mengen, die zu verschiedenen Klartextbuchstaben gehören, kein gemeinsames Element besitzen. Man spricht von *disjunkten* Mengen.
- Die Anzahl der Geheimtextzeichen, die zu einem Klartextbuchstaben gehören, entspricht der Häufigkeit dieses Buchstabens. Wir verwenden hierzu die Häufigkeiten aus Tab. 1.2.

Im folgenden Beispiel einer homophonen Chiffre sind die Geheimtextzeichen die 100 Paare 00, . . . , 99 von Ziffern, die den Buchstaben gemäß Tab. 2.1 zugeordnet sind.

Beim *Chiffrieren* ordnet man einem Klartextbuchstaben zufällig ein dazugehöriges Geheimtextzeichen zu. Der Empfänger kann dann mit obiger Tabelle einfach *dechiffrieren*: 23520127 6429 97845929346663 04597396, 9945 5682 86886200712847 141513!

Algorithmus 2.1: Homophone Chiffrierung

Chiffrieren:	Um einen Buchstaben zu verschlüsseln, wählt man zufällig eines der Zeichen, das diesem Buchstaben zugeordnet ist.
Dechiffrieren:	Um ein Zeichen zu dechiffrieren, sucht man den Buchstaben, der zu dem Zeichen gehört.

Da die Zeichen beim Chiffrieren zufällig gewählt werden, kommt jedes Zeichen (in unserem Fall also jedes Ziffernpaar) gleich häufig vor (daher der Name „homophon" = gleich lautend). Ein potentieller Kryptoanalytiker sieht sich also vor eine wesentlich schwierigere Aufgabe gestellt als beim Brechen einer monoalphabetischen Chiffrierung.

Allerdings sollte der Systementwickler nicht zu früh jubeln, denn eine *Kryptoanalyse* ist auch hier möglich. Die Analyse basiert auf der Beobachtung, dass zwar die Häufigkeiten der Geheimtextzeichen, also der Ziffernpaare gleich sind, dass man aber aus der Betrachtung von *Paaren* von Geheimtextzeichen sehr wohl Information gewinnen kann. Vergleichen Sie dazu Tab. 1.4. Wir diskutieren zwei Beispiele.

Wenn Mr. X ein Geheimtextäquivalent des Buchstabens **c** betrachtet, also etwa die Zahl 28, so wird er feststellen, dass nur ganz bestimmte Geheimtextzeichen als unmittelbare Nachfolger von 28 in Frage kommen. Dies sind die Zahlen 07, 24, 23, 47, 89, also die Geheimtextäquivalente der Buchstaben **h** und **k.** Damit „weiß" er bereits, welche Zeichen den Buchstaben **h** oder **k** entsprechen.

Wenn Mr. X ein Geheimtextäquivalent des Buchstabens **e**, also etwa 99, ins Auge fasst, so stellt er fest, dass gewisse Geheimtextzeichen als Vorgänger und als Nachfolger von 99 vorkommen – und zwar praktisch gleich häufig. Dies müssen dann die Geheimtextäquivalente des Buchstabens **i** sein.

Diese Andeutungen sind natürlich noch längst keine Kryptoanalyse. Sie sollen Ihnen nur zeigen, dass Mr. X auch einem auf den ersten Blick scheinbar unknackbaren Geheimtext nicht völlig hilflos gegenübersteht.

2.2 Die Vigenère-Chiffre

Die Vigenère-Verschlüsselung (sprich: Wischen*ä*hr) wurde im Jahre 1586 von dem französischen Diplomaten Blaise de Vigenère (1523–1596) der Öffentlichkeit zugänglich gemacht. Die Grundidee ist, verschiedene monoalphabetische Chiffrierungen im Wechsel zu benützen. Diese Idee ist so natürlich, dass

Variationen der Vigenère-Chiffre mehrfach (wieder-)erfunden wurden. Zwei der wichtigeren Vorgänger waren Johannes Trithemius (1462–1516), dessen Bücher *Poligraphia* (1518) und *Steganographia* (1531) posthum veröffentlicht wurden, und Giovanni Battista Della Porta (1538–1615), der Erfinder der *Camera obscura*, der 1558 in seinem Buch *Magia naturalis* einen polyalphabetischen Algorithmus veröffentlichte, der große Ähnlichkeit mit der Vigenère-Chiffre aufweist.

In diesem Kapitel werden wir uns hauptsächlich mit der Vigenère-Verschlüsselung, der bekanntesten unter allen „periodischen" polyalphabetischen Algorithmen, beschäftigen, und zwar aus zwei Gründen.

- Die Vigenère-Verschlüsselung ist der Prototyp für viele Algorithmen, die bis heute praktisch eingesetzt werden.
- Bei der Kryptoanalyse werden wir zwei außerordentlich wichtige Methoden kennen lernen, den Kasiski-Test und den Friedman-Test.

Um nach dem *Algorithmus von Vigenère* chiffrieren zu können, braucht man zwei Dinge: Das *Vigenère-Quadrat* und ein *Schlüsselwort* (siehe Abb. 2.1).

Das *Vigenère-Quadrat* besteht aus 26 Alphabeten, die auf folgende Weise untereinander geschrieben sind. Das erste Alphabet ist das gewöhnliche Alphabet, das zweite das um einen Buchstaben nach links verschobene, das dritte das um zwei Buchstaben verschobene und so weiter. Mit anderen Worten: Das Vigenère-Quadrat besteht aus den 26 Verschiebechiffren in natürlicher Reihenfolge. Das erkennen wir auch daran, dass in der ersten Spalte das Alphabet steht. Das heißt: Zu jedem Buchstaben gibt es eine Zeile, die mit diesem Buchstaben beginnt.

Das Schlüsselwort kann jede beliebige Buchstabenfolge sein; für unser Demonstrationsbeispiel wählen wir das Wort VENUS. Wir schreiben dieses Schlüsselwort Buchstabe für Buchstabe über den Klartext (ohne Zwischenräume), und zwar so lange, bis die Länge des Klartexts erreicht ist:

Schlüsselwort:	V	E	N	U	S	V	E	N	U	S	V	E	N	U	S	V
Klartext:	**p**	**o**	**l**	**y**	**a**	**l**	**p**	**h**	**a**	**b**	**e**	**t**	**i**	**s**	**c**	**h**

Bei der *Chiffrierung* wird ein Klartextbuchstabe mit Hilfe des über ihm stehenden Schlüsselwortbuchstabens verschlüsselt. Genauer gesagt bestimmt der Schlüsselwortbuchstabe das Alphabet, d. h. die *Zeile* im Vigenère-Quadrat, mit dem dieser Klartextbuchstabe chiffriert wird.

Noch genauer: Um den ersten Geheimtextbuchstaben zu erhalten, müssen wir in dem Alphabet, das mit V beginnt, nachsehen, was in der *Spalte* **p** steht; dies ist der Buchstabe **K**.

Klartext:	**a**	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**	**j**	**k**	**l**	**m**	**n**	**o**	**p**	**q**	**r**	**s**	**t**	**u**	**v**	**w**	**x**	**y**	**z**
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Abb. 2.1 Das Vigenère-Quadrat

Noch ein Beispiel gefällig? Voilà: Um den zweiten Buchstaben zu verschlüsseln, suchen wir in der Zeile E den Buchstaben in der Spalte **o;** dieser ist **S**. Und so weiter. Insgesamt ergibt sich:

Schlüsselwort:	V	E	N	U	S	V	E	N	U	S	V	E	N	U	S	V
Klartext:	**p**	**o**	**l**	**y**	**a**	**l**	**p**	**h**	**a**	**b**	**e**	**t**	**i**	**s**	**c**	**h**
Geheimtext:	**K**	**S**	**Y**	**S**	**S**	**G**	**T**	**U**	**U**	**T**	**Z**	**X**	**V**	**M**	**U**	**C**

Es ist klar, dass die Vigenère-Chiffe Mr. X vor erheblich größere Probleme stellt als eine monoalphabetische Chiffrierung. Die Häufigkeit der Buchstaben ist viel gleichmäßiger verteilt. Dies erkennt man schon an unserem kurzen Beispiel. Die beiden Klartextbuchstaben **a** werden in verschiedene Geheimtextbuchstaben (**S** und **U**) verschlüsselt, während der Geheimtextbuchstabe **S** von verschiedenen Klartextbuchstaben, nämlich **o, y, a** herkommt.

Algorithmus 2.2: Vigenère-Chiffrierung

Chiffrieren:	Um einen Buchstaben zu verschlüsseln, bestimmt man den über ihm stehenden Schlüsselwortbuchstaben. Der Geheimtextbuchstabe ist der Schnittpunkt derjenigen Zeile, die mit dem Schlüsselwortbuchstaben beginnt, mit derjenigen Spalte, deren erstes Element der Klartextbuchstabe ist.
Dechiffrieren:	Um einen Geheimtextbuchstaben zu entschlüsseln, bestimmt man zunächst den über ihm stehenden Schlüsselwortbuchstaben. Dann sucht man das Geheimtextalphabet, das mit diesem Schlüsselwortbuchstaben beginnt. Mit diesem Alphabet entschlüsselt man; d. h. man sucht in diesem Alphabet den Geheimtextbuchstaben und geht zu dem darüber liegenden Klartextbuchstaben.

2.3 Kryptoanalyse

Mit heutigen Methoden kann auch ein Vigenère-chiffrierter Text geknackt werden. Denn ein genügend langer Geheimtext weist viele statistisch erfassbare Regelmäßigkeiten auf, die es einem ermöglichen, das Schlüsselwort zu erschließen. Der erste veröffentlichte Angriff stammt von dem preußischen Infanteriemajor Friedrich Wilhelm Kasiski (1805–1881), der diesen 1863 publiziert hat. Eine zweite Methode geht auf Colonel William Frederick Friedman (1891–1969) zurück. Beide Methoden dienen dazu, die Schlüsselwortlänge zu bestimmen. Da beide Tests auch über die spezielle Vigenère-Analyse hinaus weitreichende und grundlegende Bedeutung haben, sollen beide Methoden hier im Detail vorgestellt werden.

Angenommen, Mr. X hat den folgenden Text (Abb. 2.2) abgefangen, von dem er weiß oder vermutet, dass er Vigenère-chiffriert ist:

2.3.1 Der Kasiski-Test

Obwohl diese wirkungsvolle Methode zur Analyse polyalphabetischer Algorithmen zuerst von Kasiski *veröffentlicht* wurde, muss man erwähnen, dass der englische Mathematiker Charles Babbage (1792–1871), der unter anderem berühmt ist für seine Konzeption eines Vorgängers des modernen Computers, umfangreiche, allerdings unveröffentlichte Untersuchungen über Kryptographie durchgeführt hat. Insbesondere hatte er den Kasiski-Test bereits 1854 entwickelt, also neun Jahre vor Kasiski. Für eine detaillierte Darstellung siehe [Fra84].

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

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

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

Abb. 2.2 Ein Geheimtext, der mit der Methode von Vigenère verschlüsselt wurde

Wir illustrieren das Verfahren an einem mit der Methode von Vigenère verschlüsselten Geheimtext (siehe Abb. 2.2). Der Test beruht auf folgender Idee: Wenn im Klartext zwei Folgen aus gleichen Buchstaben auftreten (zum Beispiel zweimal das Wort **ein**), so werden im Allgemeinen die entsprechenden Folgen im Geheimtext verschieden ausfallen; denn schon der jeweils erste Buchstabe der beiden Folgen wird in der Regel verschieden verschlüsselt. Wenn aber die beiden Anfangsbuchstaben der Folgen mit Hilfe desselben Schlüsselwortbuchstabens verschlüsselt werden, so sind die beiden Geheimtextbuchstaben gleich. In diesem Fall werden auch die jeweils zweiten Buchstaben der Klartextfolgen mit demselben Schlüsselwortbuchstaben verschlüsselt; also ergeben sich auch im Geheimtext die gleichen Buchstaben. Das heißt also: Wenn die beiden Anfangsbuchstaben der Klartextfolgen mit demselben Schlüsselwortbuchstaben verschlüsselt werden, so bestehen die entsprechenden Geheimtextfolgen aus den gleichen Buchstaben. Siehe Abb. 2.3.

Wann tritt der Fall auf, dass zwei Buchstaben mit demselben Schlüsselwortbuchstaben verschlüsselt werden? Nun, genau dann, wenn das Schlüsselwort zwischen sie genau einmal, genau zweimal, genau dreimal, … „passt". Mit anderen Worten: Genau dann, wenn der *Abstand* der beiden Klartextbuchstaben ein Vielfaches der Schlüsselwortlänge ist (siehe Abb. 2.3).

Um den Abstand zwischen zwei Folgen zu bestimmen, geht man so vor: Man zählt die Anzahl der Buchstaben zwischen den jeweils ersten Buchstaben der Folgen, wobei man den ersten Buchstaben nicht mitzählt. Zum Beispiel haben die Folgen, die mit dem Buchstaben Nr. 11 beziehunsgsweise mit dem Buchstaben Nr. 26 beginnen, den Abstand 15.

Schlüsselwort: V E N U S V E N U S V E N U S V
Klartext: ... **e i n** ... ... **e i n** ...
Geheimtext: ... **R C F** ... ... **W D R** ...
Im Allgemeinen werden Klartextfolgen aus gleichen Buchstaben in Geheimtextfolgen aus verschiedenen Buchstaben chiffriert.

Schlüsselwort: V E N U S V E N U S V E N U S V
Klartext: ... **e i n** ... ... **e i n** ...
Geheimtext: ... **Y A I** ... ... **Y A I** ...

Wenn aber die Anfangsbuchstaben der beiden Folgen unter dem gleichen Schlüsselwortbuchstaben stehen, dann bestehen auch die entsprechenden Geheimtextfolgen aus gleichen Buchstaben.

Abb. 2.3 Der Kasiski-Test

Tab. 2.2 Auswertung der Folgen aus gleichen Buchstaben

Folge	Abstand	Primfaktorzerlegung des Abstands
TNE	50	$2 \cdot 5 \cdot 5$
FCRV	265	$5 \cdot 53$
NEVHJAOVWU	90	$2 \cdot 3 \cdot 3 \cdot 5$
VWU	75	$3 \cdot 5 \cdot 5$

Wir fassen zusammen: Wenn zwei Klartextfolgen aus gleichen Buchstaben einen Abstand haben, der ein Vielfaches der Schlüsselwortlänge ist, so entsprechen ihnen im Geheimtext Folgen aus gleichen Buchstaben.

Nun dreht Mr. X den Spieß um: Er sucht zunächst im Geheimtext Folgen aus gleichen Buchstaben (siehe Abb. 2.4). Er vermutet, dass ihr Abstand „wahrscheinlich“ ein Vielfaches der Schlüsselwortlänge ist. Diese Wahrscheinlichkeit folgt dem Gesetz „je länger, je lieber“: Gleiche Buchstaben sagen, wie wir wissen, gar nichts über die Schlüsselwortlänge aus, und auch Paare aus gleichen Buchstaben können sich „zufällig“ ergeben. Aber aus Folgen von drei oder mehr gleichen Buchstaben kann Mr. X schon ziemlich zuverlässig auf die Schlüsselwortlänge schließen. In unserem Beispiel erkennt er die Struktur, die in Abb. 2.4 zu erkennen ist.

Der größte gemeinsame Faktor ist 5. Also könnte ein (zu) optimistischer Kryptoanalytiker frohlocken und sagen, „Ich weiß, dass die Schlüsselwortlänge 5 ist“. In der Tat funktioniert der Kasiski-Test in der Praxis sehr gut.

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

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

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

Abb. 2.4 Folgen aus gleichen Buchstaben

Algorithmus 2.3: Kasiski-Test (Bestimmen der Schlüsselwortlänge)

Man sucht gleiche Folgen im Geheimtext und bestimmt deren Abstand. Dieser ist (vermutlich) ein Vielfaches der Schlüsselwortlänge.

Wenn der Kryptoanalytiker Mr. X allerdings vorsichtig ist, wird er nur sagen, „Dies ist ein starkes Indiz dafür, dass die Schlüsselwortlänge 5 ist." Warum tut Mr. X gut daran, vorsichtig zu sein? Es gibt dafür zwei Gründe.

1. Es könnte sein, dass zufällig (!) zwei Geheimtextfolgen aus drei oder mehr gleichen Buchstaben vorhanden sind, die einen nicht durch 5 teilbaren Abstand haben. Dann würde sich als größter gemeinsamer Teiler 1 ergeben! (In unserem Beispiel tritt dieser Fall tatsächlich auf: Die Folge **O I S** kommt zweimal vor, und zwar mit Abstand $26 = 2 \cdot 13$.) Das heißt, dass man den größten gemeinsamen Teiler nicht „blind" ausrechnen darf, sondern dass man ihn „mit Gefühl" bestimmen muss. Offensichtliche Ausreißer muss man unberücksichtigt lassen.
2. Gerade deswegen könnte man auf die Idee kommen, die Schlüsselwortlänge könnte nicht 5, sondern 10, 15 oder 30 sein, denn die Faktoren 2 und 3 kommen auch recht häufig vor. Mit anderen Worten: Der Kasiski-Test liefert einem die Schlüsselwortlänge bis auf *Vielfache (*oder *Teiler).*

Auch aus diesem Grund präsentieren wir noch eine zweite Methode; diese ergibt die *Größenordnung* der Schlüsselwortlänge. Eine Kombination beider Methoden lässt einen dann kaum mehr in die Irre gehen.

2.3.2 Der Friedman-Test

Dieses Verfahren wurde 1925 von William Friedman entwickelt, „der als größter Kryptologe aller Zeiten gilt“ [Fra82]. Bei diesem Test fragt man sich, *mit welcher Chance ein willkürlich aus einem Klartext herausgegriffenes Buchstabenpaar aus gleichen Buchstaben besteht*. Die Antwort darauf wird durch den Koinzidenzindex gegeben.

Stellen wir uns dazu zunächst eine beliebige Buchstabenfolge der Länge n vor. Sei n_1 die Anzahl der **a**'s, n_2 die Anzahl der **b**'s, … und n_{26} die Anzahl der **z**'s.

Wir bestimmen die Anzahl der Paare, bei dem beide Buchstaben gleich **a** sind. Wir verlangen *nicht*, dass es sich um Bigramme, also um aufeinanderfolgenden Buchstaben handelt. Für die Auswahl des ersten **a**'s gibt es nach Definition genau n_1 Möglichkeiten, für die Auswahl des zweiten **a**'s dann noch $n_1 - 1$ Möglichkeiten. Da es auf die Reihenfolge der Buchstaben nicht ankommt, ist die Anzahl der gesuchten Paare gleich $n_1(n_1 - 1)/2$.

Also ist die Anzahl der Paare, bei dem beide Buchstaben gleich sind, d. h. bei denen beide gleich **a** oder beide gleich **b** … oder beide gleich **z** sind, gleich

$$\frac{n_1(n_1-1)}{2} + \frac{n_2(n_2-1)}{2} + \ldots + \frac{n_{26}(n_{26}-1)}{2} = \sum_{i=1}^{26} \frac{n_i(n_i-1)}{2}.$$

Die *Chance*, ein Paar aus gleichen Buchstaben zu erwischen, lässt sich daraus nach der Melodie „Anzahl der günstigen Fälle durch Anzahl der möglichen Fälle“ wie folgt berechnen:

$$\frac{\sum_{i=1}^{26} \frac{n_i(n_i-1)}{2}}{n(n-1)/2} = \frac{\sum_{i=1}^{26} n_i(n_i-1)}{n(n-1)}.$$

Diese Zahl heißt der (Friedmansche) *Koinzidenzindex* und wird mit I bezeichnet:

$$I = \frac{\sum_{i=1}^{26} n_i(n_i-1)}{n(n-1)}.$$

Friedman selbst bezeichnete diese Zahl mit κ (griechisches Kappa); daher findet man für die Methode, die wir im Folgenden vorstellen, manchmal auch den Namen *Kappa-Test*.

Nun nähern wir uns diesem Koinzidenzindex von einer anderen Seite. In vielen Fällen kennt Mr. X nämlich nicht nur den Geheimtext, sondern weiß von vornherein auch etwas über die Verteilung der Buchstaben.

Nehmen wir an, er *wüsste*, dass im Text der Buchstabe **a** mit der Wahrscheinlichkeit p_1 auftritt, der Buchstabe **b** mit der Wahrscheinlichkeit p_2, ..., und schließlich der Buchstabe **z** mit der Wahrscheinlichkeit p_{26}. Mr. X weiß dies, wenn es sich um einen deutschen Text handelt. Dann kann er die konkreten Werte für die Wahrscheinlichkeiten p_i beispielsweise der Tab. 1.2 entnehmen.

Stellen wir uns nun zwei willkürlich herausgegriffene Buchstaben unseres Textes vor. Die Wahrscheinlichkeit, dass der erste dieser Buchstaben gleich **a** ist, ist p_1. Auch die Wahrscheinlichkeit dafür, dass der zweite Buchstabe gleich **a** ist, ist p_1. Da beide Buchstaben willkürlich gewählt wurden, ist die Wahrscheinlichkeit dafür, dass an beiden Stellen der Buchstabe **a** steht, gleich p_1^2.

Entsprechendes gilt auch für die anderen Buchstaben. Somit ist die Wahrscheinlichkeit dafür, dass an zwei beliebig herausgegriffenen Stellen der *gleiche* Buchstabe steht (also entweder der Buchstabe **a**, der Buchstabe **b** oder ...), gleich

$$p_1 \cdot p_1 + p_2 \cdot p_2 + \ldots + p_{26} \cdot p_{26} = \sum_{i=1}^{26} p_i^2.$$

Wir fassen zusammen: Wenn wir die Wahrscheinlichkeiten der Buchstaben kennen, können wir von vornherein den Koinzidenzindex ausrechnen. Es gilt:

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

Diese Zahl hängt natürlich von den Wahrscheinlichkeiten $p_1, \ldots, p_{26}$ ab. Wir betrachten zwei Beispiele.

- Man kann den *Koinzidenzindex der deutsche Sprache* bestimmen, indem man einfach die Wahrscheinlichkeiten der Buchstaben aus Tab. 1.2 in die Formel einsetzt:

$$\sum_{i=1}^{26} p_i^2 = 0{,}0651^2 + 0{,}0189^2 + 0{,}0306^2 + \ldots + 0{,}0113^2 = 0{,}0762.$$

Dies bedeutet, dass ein zufällig gewähltes Buchstabenpaar mit einer 7,62 %-igen Chance aus gleichen Buchstaben besteht. Mit anderen Worten: Etwa jedes 13-te Buchstabenpaar besteht aus gleichen Buchstaben.

- Stellen wir uns nun andererseits einen völlig zufälligen Text vor, also einen Text, in dem die Buchstaben bunt durcheinandergewürfelt sind. Darin sind die Buchstaben gleichverteilt, also kommt jeder Buchstabe mit derselben Wahrscheinlichkeit $p_i = 1/26 \approx 0{,}0385$ vor. In diesem Fall ergibt sich als *Koinzidenzindex einer zufälligen Buchstabenfolge*

$$\sum_{i=1}^{26} p_i^2 = \sum_{i=1}^{26} \frac{1}{26^2} = 26 \cdot \frac{1}{26^2} = \frac{1}{26} \approx 0{,}0385.$$

In einem solchen sinnlosen Buchstabensalat haben wir also nur eine etwa halb so große Chance, ein Paar aus gleichen Buchstaben zu treffen: Nur etwa jedes 26-te Buchstabenpaar besteht aus gleichen Buchstaben.

Was, zum Kuckuck, hat dies alles mit Chiffrieren und der Kryptoanalyse der Vigenère-Chiffre zu tun? Diese Frage soll sogleich beantwortet werden – allerdings werden wir zur Vigenère-Verschlüsselung erst später kommen.

Lassen Sie uns für einen Augenblick zu *mono*alphabetischen Chiffrierungen zurückkehren. Da eine monoalphabetische Chiffrierung im Grunde nur eine Permutation der Buchstaben ist, bleibt die Häufigkeitsverteilung der Buchstaben erhalten. Die Häufigkeiten der einzelnen Buchstaben werden zusammen mit den Buchstaben permutiert. Zum Beispiel gehört die Häufigkeit 0,174 nicht mehr zu dem Buchstaben **e**, sondern zu dem entsprechenden Geheimtextbuchstaben.

Allgemein kann man beweisen, dass der Koinzidenzindex (oder gleichwertig $\sum p_i^2$) *größer* wird, wenn der Text *un*regelmäßiger wird, und *kleiner* wird, je *gleichmäßiger* der Text ist. Der Wert 0,0385 ist das absolute Minimum für den Koinzidenzindex. Diese Behauptung soll in Übungsaufgabe 15 bewiesen werden.

Also bleibt bei einer monoalphabetischen Chiffrierung der Koinzidenzindex gleich, während er bei einer polyalphabetischen sinkt. Denn polyalphabetische Chiffrierungen sind ja gerade dazu gemacht, die Häufigkeiten der einzelnen Buchstaben einander anzugleichen.

Daraus leiten wir einen Test ab, der uns sagt, ob ein vorgelegter Geheimtext von einer monoalphabetischen Chiffrierung herkommt oder nicht.

Algorithmus 2.4: Test, ob ein Geheimtext monoalphabetisch verschlüsselt wurde

Man berechnet den Koinzidenzindex. Wenn dieser ungefähr 0,0762 ist, so ist die Chiffrierung wahrscheinlich monoalphabetisch. Wenn der Koinzidenzindex deutlich kleiner ist, dann ist der Text nicht monoalphabetisch chiffriert.

Nun verwenden wir den Koinzidenzindex, um die Schlüsselwortlänge eines Vigenère-chiffrierten Geheimtextes zu berechnen. Das Ziel ist es, den Koinzidenzindex dieses Textes zu berechnen – ohne den Text zu kennen.

Da ein polyalphabetischer Algorithmus verwendet wurde, ist der Koinzidenzindex kleiner als 0,0762. Aber um wie viel kleiner? Antwort: Das kommt darauf an. Genauer gesagt: Es kommt auf die Länge des Schlüsselworts an.

Um aus dem Koinzidenzindex die Länge des Schlüsselworts ausrechnen zu können, müssen wir diese Länge bezeichnen. Sei h die Länge des Schlüsselworts, das heißt die Anzahl seiner Buchstaben. Wir nehmen an, dass es aus lauter verschiedenen Buchstaben besteht.

Die Formel herzuleiten, ist eigentlich nicht schwierig, aber einigermaßen aufwändig. Wenn Sie im Augenblick keine Lust auf die Herleitung von Formeln haben, so bitte ich Sie, die folgende Seite getrost zu überblättern und erst zu Beginn des Abschn. 2.3.3 wieder aufzumerken.

Unsere Strategie ist die folgende: Wir tun so, als ob wir die Länge h des Schlüsselwortes kennen würden, berechnen daraus den Koinzidenzindex – und drehen dann die Formel um.

Wir stellen uns also vor, dass Mr. X einen Geheimtext hat, von dem er weiß, dass er Vigenère-verschlüsselt ist, und von dem er zumindest ahnt, dass das Schlüsselwort aus genau h Buchstaben besteht.

Daher kann Mr. X die Buchstaben des Geheimtexts isolieren, die mit dem ersten Schlüsselwortbuchstaben verschlüsselt wurden. Das sind die Buchstaben Nr. 1, Nr. h+1, Nr. 2h+1, … Entsprechend kann er die Buchstaben isolieren, die mit dem zweiten Schlüsselwortbuchstaben verschlüsselt wurden; es handelt sich um die Buchstaben Nr. 2, h+2, 2h+2, … Und so weiter.

Mr. X systematisiert dies, indem er den Geheimtext zeilenweise in h Spalten schreibt. Dann befinden sich in der ersten Spalte die Buchstaben Nr. 1, Nr. h+1, Nr. 2h+1 und so fort., also all diejenigen Buchstaben, die mit Hilfe des ersten Schlüsselwortbuchstabens chiffriert wurden. Entsprechend befinden sich in der zweiten Spalte diejenigen Buchstaben Nr. 2, h+2, 2h+2, …,

Erster Schlüsselwort-buchstabe	Zweiter Schlüsselwort-buchstabe	Dritter Schlüsselwort-buchstabe	...	h-ter Schlüsselwort-buchstabe
1	2	3	...	h
h+1	h+2	h+3	...	2h
2h+1	2h+2	2h+3	...	3h
3h+1	3h+2	3h+3	...	4h
...	...	...	...	...

Abb. 2.5 Nummern der Buchstaben eines Vigenère-verschlüsselten Textes

also diejenigen, die mit Hilfe des zweiten Schlüsselwortbuchstabens verschlüsselt wurden. Und so weiter. Aus Abb. 2.5 wird dieses Vorgehen deutlich.

Wenn man dieses Schema sorgfältig betrachtet, kann man den Koinzidenzindex ausrechnen.

Erste Beobachtung: Die Wahrscheinlichkeiten In jeder Spalte stehen Buchstaben, die durch dieselbe monoalphabetische Chiffrierung (sogar durch dieselbe Verschiebe-Chiffre) gewonnen wurden. Nach dem Algorithmus 2.4, dem Test, ob ein Text monoalphabetisch verschlüsselt wurde, ist die Chance, in einer Spalte ein Paar aus gleichen Buchstaben zu treffen, gleich 0,0762.

Nun betrachten wir die Paare aus Buchstaben, die in verschiedenen Spalten stehen. Wenn die Schlüsselwortlänge groß ist, stehen die Buchstaben eines solchen Paares in keinem inneren Zusammenhang. Da die zugehörigen Verschlüsselungsalphabete „zufällig" gewählt wurden (die Buchstaben des Schlüsselworts sind alle verschieden!), kann ein solches Paar nur zufällig aus gleichen Buchstaben bestehen.

Die Wahrscheinlichkeit dafür ist wesentlich niedriger als 0,0762, etwa 0,0385. Sie ist exakt 0,0385, wenn das Schlüsselwort eine zufällige Buchstabenfolge ist. Falls nicht, ist diese Wahrscheinlichkeit etwas höher.

Zweite Beobachtung: Die Anzahlen Wir zählen die Anzahl der Buchstabenpaare aus gleichen Spalten und die Anzahl der Paare aus verschiedenen Spalten.

Wenn der Geheimtext insgesamt n Buchstaben hat, so stehen in jeder Spalte genau n/h Buchstaben.

Bemerkung: Auf die Betrachtung von Rundungsfehlern verzichten wir hier grundsätzlich; der Text möge so lang sein, dass diese Fehler nicht ins Gewicht fallen.

Um den ersten Buchstaben zu wählen, gibt es genau n Möglichkeiten. Ist dieser Buchstabe gewählt, so liegt auch die Spalte, in der dieser sich befindet, fest. In dieser Spalte gibt es noch $n/h - 1$ andere Buchstaben, also $n/h - 1$ Möglichkeiten, den zweiten Buchstaben zu wählen. Also ist die Anzahl der

Paare von Buchstaben, die sich *in derselben Spalte* befinden, gleich

$$n \cdot \left(\frac{n}{h} - 1\right) / 2 = \frac{n\,(n-h)}{2h}.$$

Nun zu den Paaren aus Buchstaben aus verschiedenen Spalten. Wieder wählen wir zunächst den ersten Buchstaben (n Möglichkeiten). Da es genau $n - n/h$ Buchstaben außerhalb der Spalte gibt, die durch den ersten Buchstaben festgelegt ist, ergibt sich als die Anzahl der Paare von Buchstaben *aus verschiedenen Spalten*

$$n \cdot \left(n - \frac{n}{h}\right) / 2 = \frac{n^2 \cdot (h-1)}{2h}.$$

Nun fassen wir die Beobachtungen zusammen: Die erwartete *Anzahl* A von Paaren aus gleichen Buchstaben ist

$$A = \frac{n \cdot (n-h)}{2h} \cdot 0{,}0762 + \frac{n^2 \cdot (h-1)}{2h} \cdot 0{,}0385.$$

Also ist die *Wahrscheinlichkeit*, ein Paar aus gleichen Buchstaben zu treffen, gleich

$$\begin{aligned}\frac{A}{n\,(n-1)\,/\,2} &= \frac{n-h}{h\,(n-1)} \cdot 0{,}0762 + \frac{n\,(h-1)}{h\,(n-1)} \cdot 0{,}0385 \\ &= \frac{0{,}0377n + h\,(0{,}0385 - 0{,}0762)}{h\,(n-1)}.\end{aligned}$$

Wir wissen, dass der Koinzidenzindex eine sehr gute Annäherung an diese Zahl ist; daher gilt

$$I = \frac{0{,}0377n}{h\,(n-1)} + \frac{0{,}0385n - 0{,}0762}{n-1}.$$

Durch Umformen und Auflösen nach h ergibt sich daraus die wichtige, auf Friedman zurückgehende Formel für die Länge h des Schlüsselworts:

Algorithmus 2.5: Friedman-Test (Bestimmen der Schlüsselwortlänge)

Man bestimmt den Koinzidenzindex I und setzt ihn in folgende Formel ein:

$$h = \frac{0{,}0377n}{I \cdot (n-1) - 0{,}0385n + 0{,}0762}.$$

Diese Formel sieht vielleicht schwierig aus, und jedenfalls war ihre Herleitung langwierig. Es ist jedoch einfach, sie in einem konkreten Fall anzuwenden. Man braucht nämlich nur die Länge n des Textes und die Häufigkeiten n_i der Buchstaben – und mit diesen lächerlich wenigen Daten liefert uns die Zauberformel dann die Länge h des Schlüsselworts!

Nun ernten wir die Früchte unserer Arbeit, indem wir diese Theorie auf unser Beispiel vom Beginn dieses Abschnitts anwenden. Per Strichliste zählen wir die n_i's; es ergibt sich $n = 368$ und

$$\sum_{i=1}^{26} n_i\,(n_i - 1) = 6468.$$

Also ist

$$I = \frac{6468}{135.056} \approx 0{,}048.$$

Somit handelt es sich mit großer Wahrscheinlichkeit um eine polyalphabetische Chiffrierung. Mit der Formel aus Algorithmus 2.5 berechnen wir nun die Schlüsselwortlänge; es ergibt sich $h \approx 3{,}937$. Dies deutet zusammen mit den Ergebnissen des Kasiski-Tests darauf hin, dass die Schlüsselwortlänge tatsächlich 5 (und nicht 10, 15 oder 20) ist.

2.3.3 Bestimmung des Schlüsselworts

Nachdem nun die Schlüsselwortlänge bestimmt ist, geht es darum, das Schlüsselwort selbst zu erkennen. Das ist aber nicht mehr schwierig.

Wenn der Kryptoanalytiker Mr. X die Schlüsselwortlänge h kennt, so weiß er, dass die Buchstaben, die unter dem ersten Schlüsselwortbuchstaben stehen (das sind die Buchstaben Nr. 1, h+1, 2h+1, …) durch dieselbe monoalphabetische Chiffrierung, ja sogar durch dieselbe Verschiebe-Chiffrierung gewonnen worden sind. Es genügt in der Regel also, das Äquivalent des Buchstabens **e** zu finden.

Entsprechendes gilt für die Buchstaben, die unter dem zweiten Schlüsselwortbuchstaben stehen (also die Buchstaben Nr. 2, h+2, 2h+2, …). Und so weiter.

In unserem Beispiel ist $h = 5$. Von den 74 Buchstaben des „ersten" monoalphabetischen Teiltextes sind 13 gleich **F**. Daher entspricht **e** dem Buchstaben **F.** Ein Blick auf das Vigenère-Quadrat zeigt sofort, dass der erste Schlüsselwortbuchstabe **B** ist.

Auf diese Weise kann man leicht das Schlüsselwort erkennen und damit den Text entschlüsseln (vergleiche Übungsaufgabe 6).

2.4 Schlussbemerkungen

Wir haben gesehen, dass jede Vigenère-Chiffrierung zu knacken ist – sogar mit relativ einfachen Mitteln. Jede? – Nein, natürlich nur solche, bei denen das Schlüsselwort ziemlich kurz ist.

Konsequenterweise betrachten wir jetzt Vigenère-Chiffrierungen mit langem Schlüsselwort. Um uns nicht durch nebensächliche Diskussionen ablenken zu lassen (was heißt „lang"?), behandeln wir von vornherein die längstmöglichen Schlüsselwörter: Unsere Schlüsselwörter sollen so lang wie der Klartext sein. Wir stellen zwei Tricks vor, die beide dazu dienen, Mr. X die Freude an den oben beschriebenen Methoden zu vergällen.

1. Trick Man könnte versuchen, als Schlüssel„wort" den Text eines Buches zu verwenden. Ein solcher Schlüssel hat bestimmt den Vorteil, ohne große Probleme übermittelt werden zu können. Zum Beispiel muss der Empfänger nur die Information „Fontane: Irrungen, Wirrungen" haben, um den Geheimtext mit dem größten Vergnügen mittels des folgenden „Wortes" dechiffrieren zu können:

> An dem Schnittpunkte von Kurfürstendamm und Kurfürstenstraße, schräg gegenüber dem „Zoologischen", befand sich in der Mitte der siebziger Jahre noch eine große, feldeinwärts sich erstreckende Gärtnerei, deren kleines, dreifenstriges, in einem Vorgärtchen um etwa hundert Schritte zurückgelegenes Wohnhaus, trotz aller Kleinheit und Zurückgezogenheit, von der vorübergehenden Straße sehr wohl erkannt werden konnte. Was aber sonst noch zu dem Gesamtgewese der Gärtnerei gehörte, ja die recht eigentliche Hauptsache derselben ausmachte, war durch eben dies kleine Wohnhaus wie durch eine Kulisse versteckt …

Bei der Verwendung eines solchen Schlüssels laufen alle Methoden zur Bestimmung der Schlüsselwortlänge selbstverständlich ins Leere. Da aber der Schlüssel ein deutschsprachiger Text ist, schlagen statistisch signifikante Daten der Sprache auf den Geheimtext durch, so dass diese Chiffre nicht als sicher bezeichnet werden kann. Der erste, der diese Schwäche erkannte, war ebenfalls Friedman. Deshalb gehen wir noch einen Schritt weiter.

2. Trick Beim 1. Trick konnte Mr. X immer noch Statistik benutzen, weil das Schlüsselwort statistisch erfassbare Widerhaken aufwies. Deshalb wählen

wir nun als Schlüsselwort eine praktisch unendlich lange, völlig bunt gewürfelte Folge von Buchstaben, an der sämtliche statistische Tests widerstandslos abgleiten. Das ist eine (Buchstaben-)Zufallsfolge, die man sich z. B. erzeugen kann durch Werfen eines fairen 26-seitigen Buchstabenwürfels. Eine solche Folge nennen wir einen *Buchstabenwurm*.

Ein derartiger Wurm hat die Eigenschaft, dass man aus keinem noch so langen Stück auch nur einen einzigen weiteren Buchstaben vorhersagen kann. Verschlüsselt man einen Klartext mit Hilfe eines Buchstabenwurms, so hat offenbar auch der Geheimtext keine statistisch signifikanten Ansatzpunkte mehr, bei denen Mr. X für seine Kryptoanalyse einhaken könnte. Auch wenn er ein noch so langes Stück Geheimtext (ja sogar ein beliebig langes zusammengehöriges Stück Klartext-Geheimtext) kennt, kann er keinen einzigen (weiteren) Buchstaben bestimmen. Ein solches System ist sogar *theoretisch sicher*! Mit anderen Worten: Es bietet *perfekte Sicherheit*.

Diese perfekten Systeme werden wir im folgenden Kapitel genauer unter die Lupe nehmen.

2.5 Übungsaufgaben

1 Konstruieren Sie (auf dem Papier) eine „Maschine“, mit deren Hilfe Sie die in Abschn. 2.1 angegebene Chiffrierung vollziehen können. Konstruieren Sie entsprechend eine „Maschine“ zum Dechiffrieren.

2 @(a) Schreiben Sie ein Programm, das die in Abschn. 2.1 angegebenen homophone Chiffrierung realisiert.
   (b) Chiffrieren Sie damit einen relativ langen Text, und überprüfen Sie, ob (i) alle Buchstaben ungefähr mit der gleichen Häufigkeit auftreten, (ii) ob die Paare von Buchstaben im Geheimtext gleichmäßig verteilt sind.

3 Die Chiffriermethode aus Abschn. 2.1 ist nur ein Beispiel aus einer ganzen Klasse von Verfahren. Beschreiben Sie diese Klasse – und zwar so, dass klar wird, was der „Algorithmus“ ist und welches die „Schlüssel“ sind.

4 Angenommen, für jeden Buchstaben ist die Anzahl der Geheimtextsymbole die von Tab. 2.1. Das heißt, dass wir die Verteilung (5, 2, 3, 5, 17, 2, 3, 4, 8, 1, 1, 3, 3, 10, 3, 1, 1, 7, 7, 6, 3, 1, 1, 1, 1, 1) vorliegen haben.
   (a) Wie können die Symbole 00, …, 99 den Buchstaben zugeordnet werden, so dass
   - jeder Buchstabe die vorgeschriebene Anzahl von Geheimtextsymbolen erhält, und
   - diese Zuordnung „zufällig“ ist?

@(b) Schreiben Sie ein Programm, das jedem Buchstaben die geforderte Anzahl von Geheimtextsymbolen in zufälliger Weise zuordnet.

(c) Wie viele Zuordnungen der Symbole 00, …, 99 zu den Buchstaben gibt es, wenn jeder Buchstabe die geforderte Zahl von Geheimtextsymbolen erhalten soll?

5 Bestimmen Sie weitere Folgen aus gleichen Buchstaben im Beispiel aus Abschn. 2.3.

6 Bestimmen Sie alle Schlüsselwortbuchstaben des Beispiels aus Abschn. 2.3 und entziffern Sie den Text.

7 *Der folgende Text wurde mit dem Vigenère-Algorithmus verschlüsselt. Bestimmen Sie mit dem Kasiski-Test die Schlüsselwortlänge, bestimmen sie das Schlüsselwort und entschlüsseln Sie den Text:

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

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

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

Achtung: Bei so kurzen Texten ist der häufigste Buchstabe nicht immer e. Arbeiten Sie sorgfältig und mit Überlegung, sonst ergibt sich nur Murks.

8 Wenden Sie den Friedman-Test auf den Text der vorigen Übungsaufgabe an und vergleichen Sie das Ergebnis mit dem Kasiski-Test.

9 Ist die folgende Aussage richtig: „Es ist entscheidend, dass an jeder Stelle des Kryptogramms der Schlüssel eindeutig den Klartextbuchstaben zu jedem Geheimtextbuchstaben festlegt"?

10 Kann man durch statistische Untersuchungen eine Transpositions- von einer Substitutionschiffre unterscheiden?

11 Man kann das Verfahren von Vigenère auch dann anwenden, wenn man die Buchstaben durch die Zahlen 0, 1, 2, …, 25 darstellt. Beschreiben

Sie diese „modulo 26-Variante“ des Vigenère-Algorithmus (vergleichen Sie dazu Algorithmus 1.3).

12 @(a) Schreiben Sie ein Programm, das gleiche Folgen von Buchstaben in einem Text findet und den Abstand zwischen diesen Folgen bestimmt.

(b) Schreiben Sie ein Programm, das den Kasiski-Test benutzt, um die Schlüsselwortlänge eines Vigenère-verschlüsselten Textes zu bestimmen.

13 Chiffrieren Sie einen deutschen Text (der mindestens 40mal so viele Buchstaben enthält wie das Schlüsselwort) nach Vigenère, geben Sie diesen einer Freundin mit der Aufforderung, den Text zu entziffern.

14 Berechnen Sie den Koinzidenzindex des Geheimtextes aus Übungsaufgabe 20 von Kap. 1.

15 @Schreiben Sie ein Programm, das

(i) die Länge n eines Textes bestimmt,

(ii) die Häufigkeiten $n_1, \ldots, n_{26}$ der Buchstaben dieses Textes bestimmt, und

(iii) den Koinzidenzindex des Textes berechnet.

16 Seien $p_1, \ldots, p_{26}$ die Häufigkeiten der Buchstaben **a**, …, **z** eines Textes.

(a) Zeigen Sie

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

(b) Erklären Sie, weshalb

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

nicht kleiner als 1/26 (= 0,038) werden kann und diskutieren sie den Fall der Gleichheit.

17 @Entwerfen Sie ein Programm zur Vigenère-Chiffrierung, sowie eines, das Vigenère-chiffrierte Texte knackt (*).

18 ¿ M R U W N I E H C I L K R I W E G L O F E S E I D T S I

19 Der erste polyalphabetische (besser: nicht-monoalphabetische) Substitutionsalgorithmus wurde im Jahre 1470 von Leon Battista Alberti veröffentlicht. Er benutzt eine Maschine aus zwei konzentrischen Scheiben mit je 26 Zeichen (Alberti selbst verwendete 24 Zeichen). Auf der inneren Scheibe steht das Geheimtextalphabet in beliebiger, aber fester Anordnung; auf der äußeren Scheibe finden sich die Zahlen 1, 2, 3, und 4, sowie alle Buchstaben des Alphabets – mit Ausnahme der vier Buchstaben j, y, x, q, die die

geringsten Häufigkeiten aufweisen. Sender und Empfänger müssen sich auf einen Buchstaben, den sogenannten *Indikator* einigen.
Zum Verschlüsseln stellt der Sender die Scheibe beliebig ein. Er sucht den Indikator auf dem äußeren Ring und übermittelt als ersten Buchstaben denjenigen Geheimtextbuchstaben, der dem Indikator entspricht. Nun wird der Klartext verschlüsselt, als würde es sich um eine monoalphabetische Chiffrierung handeln. Falls zufällig einer der Buchstaben j, y, x oder q vorkommt, wird dieser einfach weggelassen.
Wann immer der Sender die Beziehung Klartext-Geheimtext verändern möchte, wählt er eine der Zahlen 1, 2, 3, oder 4, bestimmt den zugehörigen Geheimtextbuchstaben und übermittelt diesen als den nächsten Buchstaben. Dann dreht er die Scheibe so, dass dieser Geheimtextbuchstabe gegenüber dem Indikator liegt.
Aufgaben:
(a) Basteln Sie ein Modell für den *Alberti-Algorithmus* und verschlüsseln Sie eine Nachricht.
(b) Wie muss der Empfänger beim Entschlüsseln vorgehen?
(c) Welche Vorteile hat der Alberti-Algorithmus gegenüber dem Vigenère-Algorithmus?
(d) Welches sind die Schlüssel beim Alberti-Algorithmus?
(e) Wie sicher ist dieser Algorithmus?

Zusatzinformation: Eine „Modifikation" des Alberti-Algorithmus wurde in den italienischen Streitkräften noch nach dem zweiten Weltkrieg benutzt! Keine Angst: Der Alberti-Algorithmus wurde nur dazu verwendet, bereits verschlüsselte Nachrichten nochmals zu „überschlüsseln".

20 Machen Sie sich einen schönen Tag und lesen Sie Fontanes Roman *Irrungen, Wirrungen*.

3

Sicher ist sicher oder Ein bisschen Theorie

Manche Menschen benützen ihre Intelligenz zum Vereinfachen, manche zum Komplizieren (Erich Kästner).

In diesem Kapitel werden wir die Dinge, über die wir bislang schon viel geredet haben, auf eine solide Grundlage stellen. Insbesondere werden wir sagen, was wir unter perfekter Sicherheit eines Chiffriersystems verstehen wollen. Dies wird uns zwanglos zu in der Praxis verwendeten Verfahren führen. Bei der Diskussion perfekter Systeme werden wir nicht umhin können, einige wahrscheinlichkeitstheoretische Überlegungen anzustellen. Allerdings werden wir nur Begriffe elementarster Art brauchen: Mit dem Wissen, dass eine Wahrscheinlichkeit eine Zahl zwischen 0 und 1 ist, kommt man schon ziemlich weit. Diejenigen, die es noch genauer wissen wollen, seien auf die Lehrbuchliteratur, z. B. [Hen13] verwiesen.

3.1 Chiffriersysteme

Nach unseren bisherigen Vorstellungen vereinbaren Sender und Empfänger *einen* Schlüssel und chiffrieren damit *einen* Klartext. Der richtige Gesichtspunkt für die folgenden Überlegungen unterscheidet sich ein bisschen von diesem Ansatz. Wir betrachten jetzt *Systeme*, die aus einer *Menge* von Schlüsseln, einer *Menge* von Klartexten und den zugehörigen Geheimtexten bestehen. Wenn man die folgenden Überlegungen ganz präzise durchführen wollte, so müsste man alles axiomatisch behandeln; für unsere Zwecke ist es aber besser, die Konzepte durch typische Beispiele zu erklären.

Ein typisches Beispiel für eine Menge von Klartexten ist die Sammlung der deutschen Romane von 1800 bis 1900, die sich im Deutschen Literaturarchiv in Marbach befinden. Wir erhalten ein Chiffriersystem, wenn wir zusätzlich alle 312 affinen Chiffrierungen (siehe Abschn. 1.4) und die zugehörigen Geheimtexte betrachten. Für ein anderes Beispiel nehmen wir als Klartexte alle lateinischen Wörter, die in einem der bislang erschienenen *As-*

A. Beutelspacher, *Kryptologie*, DOI 10.1007/978-3-658-05976-7_3,

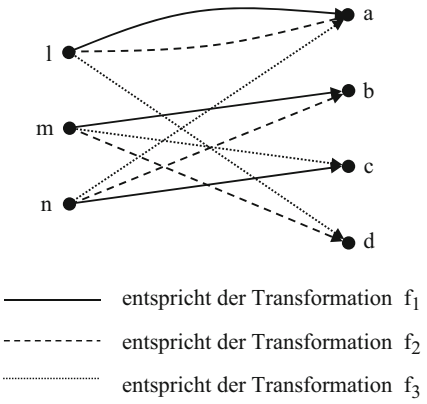


Abb. 3.1 Ein Chiffriersystem

terix-Bände vorkommen, alle 26 additiven Verschlüsselungsalgorithmen und die resultierenden Geheimtexte.

Wir benötigen einige Bezeichnungen. Mit **M** (für messages) bezeichnen wir die Menge aller Klartexte (in unseren Beispielen die Menge aller Romane beziehungsweise die lateinischen Wörter aus *Asterix*). Die Menge aller Schlüssel – wir werden dafür keinen speziellen Namen brauchen – ist in unserem ersten Beispiel die Menge aller Paare (s, t), wobei s eine Verschiebung und t eine der 12 zulässigen Multiplikationen bezeichnet. Schließlich sei **C** die Menge aller Geheimtexte (im ersten Beispiel erhalten wir 312 Geheimtexte für jeden Klartext).

Ein einfaches, aber nicht sehr realistisches Chiffriersystem ist in Abb. 3.1 zu sehen.

M besteht aus drei Klartexten l, m, n. Wenn wir die Mächtigkeit einer Menge durch senkrechte Striche bezeichnen, so ist also $|\mathbf{M}| = 3$. Die Menge **C** der Geheimtexte besteht aus den Elementen a, b, c, d; also ist $|\mathbf{C}| = 4$. Der Algorithmus wird in diesem Bild durch Pfeile dargestellt; er wird mit dem Buchstaben f bezeichnet, wobei als Index 1, 2 oder 3 hinzugefügt wird; diese Zahlen repräsentieren die Schlüssel. Beispielsweise ist $f_1(1) = a$, $f_1(m) = b$, $f_1(n) = c$.

Allgemein bezeichnen wir mit f den Verschlüsselungsalgorithmus. Wenn Sender und Empfänger einen Schlüssel k vereinbart haben, benutzen sie eine Spezialisierung f_k des Algorithmus f. Für jeden Schlüssel k muss es also eine solche Spezialisierung f_k von f geben. Diese Abbildungen f_k von **M** in **C** nennen wir *Transformationen*. Die Menge aller Transformationen sei **F;** im Beispiel von Abb. 3.1 ist also $|\mathbf{F}| = 3$.

Was sind die Eigenschaften der Abbildungen $f_k \in \mathbf{F}$? Eine Forderung, die keiner Begründung bedarf ist, dass der Empfänger in der Lage sein muss, den Geheimtext $f_k(m)$ zu entschlüsseln. Daher muss es eine Abbildung f_k^{-1} geben

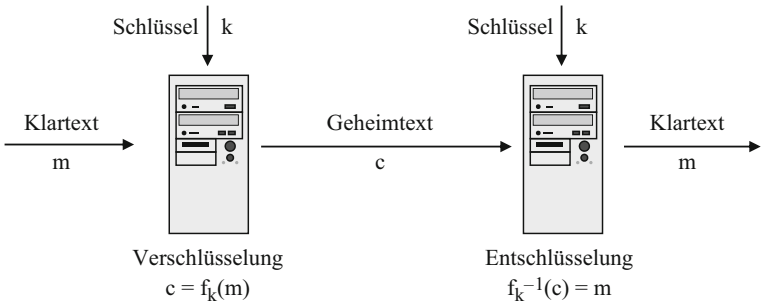


Abb. 3.2 Ein (symmetrisches) Chiffriersystem

mit folgender Eigenschaft:

$$f_k^{-1}(f_k(m)) = m \quad \text{für alle } m \in \mathbf{M}.$$

Mit anderen Worten: Jeder Geheimtext $c = f_k(m)$ kann durch Anwenden von f_k^{-1} entschlüsselt werden. Eine Transformation mit dieser Eigenschaft heißt *umkehrbar*.

Nun sind wir in der Lage, ein Chiffriersystem zu definieren. Diese Definition geht im Wesentlichen auf den Vater der modernen Kryptographie Claude E. Shannon (1916–2001) [Sha49] zurück.

Ein *symmetrisches Chiffriersystem* besteht aus einer Menge **M** von Klartexten, einer endlichen Menge **C** von Geheimtexten und einer Menge **F** von umkehrbaren Transformationen von **M** in **C** (siehe Abb. 3.2). Das System wird symmetrisch genannt, weil Sender und Empfänger bezüglich ihrer Schlüssel symmetrische Rollen haben.

Bevor wir weitere Buchstaben einführen, sollten wir innehalten und nochmals das Beispiel aus Abb. 3.1 betrachten. Dieses Beispiel macht uns zwei Dinge klar:

Es ist möglich, dass zwei verschiedene Transformationen (in unserem Beispiel f_1 und f_2) denselben Klartext in den denselben Geheimtext überführen.

Die Tatsache, dass die Transformation f_k umkehrbar ist, bedeutet, dass an keinem Punkt der rechten Seite (d. h. an keinem Geheimtext) zwei Pfeile, die beide mit f_k bezeichnet sind, ankommen. Sonst könnte dieser Geheimtext nicht eindeutig dechiffriert werden. Daraus ergibt sich zwingend, dass die Anzahl der Klartexte höchstens so groß sein kann wie die Anzahl der Geheimtexte: $|\mathbf{M}| \leq |\mathbf{C}|$.

Der Leser ist aufgefordert, sich einige (möglichst exotische) Chiffriersysteme zu konstruieren (wobei **M, C** und **F** höchstens sechs Elemente zu haben brauchen).

3.2 Perfekte Sicherheit

Jetzt wissen wir, was ein Chiffriersystem ist. Wir können aber noch nicht sagen, was es bedeuten soll, dass ein solches System „sicher“ ist. Das Hauptziel dieses Abschnittes ist es, zu einer präzisen Beschreibung eines sicheren Chiffriersystems zu kommen. Da wir nicht nur an „sicheren“, sondern sogar an „perfekten“ Systemen interessiert sind, müssen wir auch sagen, was das sein soll.

Intuitiv gesagt bedeutet „perfekte Sicherheit“, dass der Kryptoanalytiker Mr. X keine Chance hat, seine Kenntnisse über das System zu vergrößern, auch wenn ihm ein langer Geheimtext, alles Wissen und alle Rechenkapazität der Welt zur Verfügung stehen.

In den beiden ersten Kapiteln haben wir erfahren, dass Kryptoanalyse viel mit Wahrscheinlichkeiten zu tun hat. Daher ist es kein Wunder, dass dieser Begriff auch jetzt wieder auftaucht. Für jeden Klartext μ sei $\mathbf{p}(\mu)$ die Wahrscheinlichkeit für sein Auftreten.

Beispiele: (a) Angenommen, die „Klartexte“ in **M** sind die *Buchstaben* der Wörter eines deutschen Buches, sagen wir, der Märchen der Gebrüder Grimm; im Sinne von Abb. 3.1 wären dann beispielsweise die Buchstaben e, s, w, a, r, e, i, n, m, a, l, … auf der linken Seite des Bilder untereinander geschrieben. Falls die uns interessierende „Nachricht“ μ der Buchstabe **e** ist, so ist $p(\mu)$ nichts anderes als die relative Häufigkeit des Buchstabens **e**; gemäß Tab. 1.2 wäre also $p(\mu) = 0{,}174$.

(b) Nun stellen wir uns vor, dass die Klartextmenge **M** aus den aufeinander folgenden *Paaren* der Buchstaben von Grimms Märchen besteht. In unserem Standardformat wären all die Paare es, sw, wa, ar, re, ei, in, nm, ma, al, … in einer Spalte aufgelistet. Ist μ ein spezielles solches Bigramm, so ist $p(\mu)$ die Häufigkeit dieses Bigramms in Grimms Märchen. Wie wir in Abschn. 1.6 erwähnt haben, sind en und er die häufigsten Bigramme der deutschen Sprache.

Die Zahlen $p(\mu)$ heißen *a priori*- (oder *theoretische*) Wahrscheinlichkeiten. Man kann diese unabhängig vom Verschlüsseln bestimmen, und sie sind natürlich jedem guten Kryptoanalytiker bekannt.

Wir stellen uns nun vor, dass Mr. X einen Geheimtext γ abgefangen hat. Um γ zu analysieren, kann er – mindestens prinzipiell – alle Klartexte μ durchgehen und jeweils die Wahrscheinlichkeit dafür bestimmen, dass der zu analysierende Geheimtext γ von dem Klartext μ herkommt. Wir bezeichnen diese *a posteriori*- (oder *beobachteten*) Wahrscheinlichkeiten mit $p_\gamma(\mu)$.

Beispiele: (c) Sei **M** wie in obigem Beispiel (a) die Liste von Buchstaben aus Grimms Märchen; als Algorithmus betrachten wir die Verschiebechiffren mit allen 26 möglichen Schlüsseln. Wenn wir die Information aus Tab. 1.1

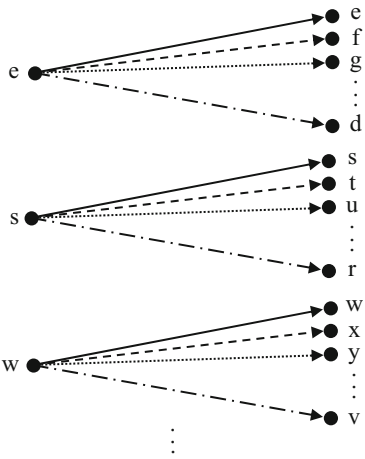


Abb. 3.3 Das Kryptosystem für Beispiel (c)

dazu verwenden, ein Abb. 3.1 entsprechendes Bild anzufertigen, so würden wir zunächst die Klartextbuchstaben links auflisten; von jedem solchen Buchstaben würden dann 26 Geheimtextbuchstaben ausgehen (für jeden Schlüssel einer), und zwar zu 26 Geheimtextbuchstaben – genauer gesagt zu denen, die in Tab. 1.1 unter dem entsprechenden Klartextbuchstaben zu finden sind. Es gibt also 26-mal so viele Buchstaben in der rechten Spalte wie in der linken (siehe Abb. 3.3).

Man beachte, dass jeder Geheimtextbuchstabe γ die gleiche Chance hat, von einem bestimmten Klartextbuchstaben herzurühren. Jeder Klartext ist möglich, und zwar ist er genau so wahrscheinlich, wie seine a priori-Wahrscheinlichkeit angibt: In etwa 17,4 % der Fälle kommt γ von **e** her, in etwa 9,8 % der Fälle von **n** usw. Mit anderen Worten

$$\text{Für jeden Geheimtext } \gamma \text{ ist } p_\gamma(\mu) = p(\mu) \text{ für jeden Klartext } \mu.$$

(d) Nun möge jeder Klartext aus **M** bestehen aus den ersten 100 Buchstaben einer jeden Seite des ersten Bandes des *Großen Brockhaus*. Somit ist **M** gleich der Anzahl der Seiten dieses Bandes. Der Algorithmus möge wieder aus den Verschiebechiffren bestehen. Das Bild bestünde jetzt aus Blöcken von je 100 Buchstaben, die in einer riesigen linken Spalte zusammengefasst wären; von jedem solchen Block gingen 26 Pfeile aus, jeder zu dem entsprechenden Geheimtext. Für jeden Klartext μ ist seine Wahrscheinlichkeit $p(\mu) = 1 / |\mathbf{M}|$, eine zwar sehr kleine, aber immer noch knapp positive Zahl. Da es ganz einfach ist zu überprüfen, ob ein spezieller Geheimtext γ von einem bestimmten Klartext μ herkommt oder nicht (die Verteilung der Buchstaben in γ muss genau der Verteilung der Buchstaben in μ entsprechen), ist $p_\gamma(\mu)$ entweder 0

oder 1. Das bedeutet insbesondere:

$$\text{Für jeden Geheimtext } \gamma \text{ ist } p_\gamma(\mu) \neq p(\mu) \text{ für jeden Klartext } \mu.$$

Diesen Sachverhalt müssen wir uns noch etwas genauer zu Gemüte führen: Angenommen, der Kryptoanalytiker Mr. X würde feststellen, dass für ein gewisses μ

$$p_\gamma(\mu) > p(\mu)$$

gilt. Dann wüsste er, dass der Geheimtext γ mit hoher Wahrscheinlichkeit von dem Klartext μ herkommt. Er hätte also durch die Analyse etwas gelernt. Das soll aber bei einem perfekten System natürlich nicht der Fall sein. Wäre andererseits

$$p_\gamma(\mu) < p(\mu),$$

so wüsste Mr. X, dass γ nur mit sehr kleiner Wahrscheinlichkeit von μ herkommt. Auch in diesem Fall hätte er seine Kenntnisse erweitert.

Wir definieren nun: Ein Chiffriersystem **S** bietet *perfekte Sicherheit*, falls für jeden Geheimtext γ gilt

$$p_\gamma(\mu) = p(\mu) \text{ für alle Klartexte } \mu.$$

Mit anderen Worten: **S** ist perfekt, falls die a priori-Wahrscheinlichkeiten gleich den a posteriori-Wahrscheinlichkeiten sind. Mr. X kann dann so hart arbeiten, wie er will – er wird nach der Analyse keinen Deut schlauer sein als zuvor!

Das ist genau das Phänomen, das in Beispiel (c) auftritt. *Verschiebechiffren sind perfekt, wenn sie auf einzelnen Buchstaben operieren!*

So weit, so perfekt. Aber: Gibt es überhaupt perfekte Systeme? Noch genauer gefragt: Wie kann man erkennen, ob ein System perfekt ist oder nicht? Was wir zur Beantwortung dieser Fragen brauchen, sind einige einfach zu überprüfende Kriterien, die uns sagen, ob ein Chiffriersystem perfekt ist oder nicht.

1. Kriterium *Wenn das Chiffriersystem* **S** *perfekt ist, so kann jeder Klartext mit einem zu* **S** *gehörigen Schlüssel auf jeden beliebigen Geheimtext abgebildet werden.*

Mit anderen Worten: Zeichnen wir ein perfektes System **S** *entsprechend Abb. 3.1, so führt von jedem Punkt links zu jedem Punkt rechts (mindestens) ein Pfeil.*

Warum gilt dieses Kriterium? Betrachten wir dazu einen Klartext μ und einen Geheimtext γ.

Erste Beobachtung: Da **S** perfekt ist, gilt $p_\gamma(\mu) = p(\mu)$.

Zweite Beobachtung: In jedem System ist $p(\mu) > 0$, da jeder Klartext ja mit einer (vielleicht sehr kleinen, aber doch noch positiven) Wahrscheinlichkeit auftritt. Zusammen ergibt sich $p_\gamma(\mu) > 0$.

Was heißt das? Dies bedeutet, dass es einen Schlüssel gibt, mit dem μ in γ chiffriert wird. (Gäbe es keinen solchen Schüssel, so wäre nämlich $p_\gamma(\mu) = 0$). Damit haben wir das erste Kriterium bereits bewiesen.

Dieses Kriterium ist sehr nützlich – in einem „negativen Sinn“: Es ermöglicht uns zu entscheiden, dass gewisse Systeme *nicht perfekt* sind. Beispielsweise ist das Chiffriersystem aus Abb. 3.1 nicht perfekt (zum Beispiel ist l nicht b verbunden). Ebenso kann die Tatsache, dass das Beispiel (d) nicht perfekt ist, auf Kriterium 1 zurückgeführt werden.

2. Kriterium *Wenn* **S** *perfekt ist, dann gilt*:

$$|\mathbf{F}| \geq |\mathbf{C}| \geq |\mathbf{M}|.$$

Wir haben uns bereits am Ende von Abschn. 3.1 klargemacht, dass in jedem Chiffriersystem $|\mathbf{C}| \geq |\mathbf{M}|$ gilt – dies hat nichts mit Perfektheit zu tun.

Warum ist $|\mathbf{F}| \geq |\mathbf{C}|$? Um das einzusehen, betrachten wir einen gegebenen Klartext μ und verschlüsseln diesen mit allen möglichen Transformationen von **S**. Nach dem 1. Kriterium kann μ in jeden möglichen Geheimtext überführt werden. Für jeden Geheimtext braucht man aber mindestens eine Transformation. Denn ein- und dieselbe Transformation kann unmöglich μ in γ und gleichzeitig in $\gamma' \neq \gamma$ abbilden. Also braucht man mindestens so viele Transformationen wie Geheimtexte. Somit gilt $|\mathbf{F}| \geq |\mathbf{C}|$.

Nun drehen wir den Spieß um. Das folgende Kriterium ermöglicht es, perfekte Systeme in Hülle und Fülle zu produzieren. Der Beweis dieser Tatsache ist prinzipiell nicht schwierig. Trotzdem verlangt er etwas Routine im Umgang mit Wahrscheinlichkeiten. Interessenten finden in Übungsaufgabe **9** Trost und Hilfe.

3. Kriterium *Sei* **S** *ein Chiffriersystem mit*

$$|\mathbf{F}| = |\mathbf{C}| = |\mathbf{M}|,$$

in dem alle Schlüssel mit der gleichen Wahrscheinlichkeit vorkommen. Ferner setzen wir voraus, dass es zu jedem Klartext μ und zu jedem Geheimtext γ genau eine Transformation aus **S** *gibt, die μ in γ überführt. Dann ist* **S** *perfekt*!

Wie gesagt, damit kann man perfekte Systeme auf einfachste Weise konstruieren. Abbildung 3.4 zeigt ein Beispiel. (Die vier verschiedenen Transformationen sind durch unterschiedliche Pfeilarten dargestellt).

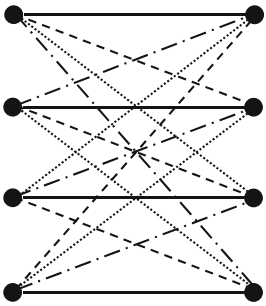


Abb. 3.4 Ein perfektes Chiffriersystem

3.3 Das One-Time-Pad

Jetzt wollen wir ein perfektes System besprechen, das sowohl für die Theorie als auch als Vorbild für die Praxis eine nicht zu überschätzende Bedeutung hat.

Die Klartexte bestehen aus allen Buchstabenkombinationen einer gewissen Länge, sagen wir der Länge n. Als Schlüssel wählen wir ebenfalls alle 26^n Folgen aus n Buchstaben; jede dieser Schlüsselfolgen werde mit der gleichen Wahrscheinlichkeit gewählt. Die Verschlüsselung eines Klartextes $\mathbf{a_1 a_2 \ldots a_n}$ erfolgt gemäß dem Schema aus Abb. 3.5.

Jeder Klartextbuchstabe $\mathbf{a}_i$ wird sozusagen zum Schlüsselbuchstaben $\mathbf{k}_i$ addiert; anders gesagt: $\mathbf{a}_i$ wird mit demjenigen Verschiebealphabet chiffriert, das mit dem Buchstaben $\mathbf{k}_i$ beginnt. In der Sprache von Kap. 2 heißt dies: Man benutzt den Vigenère-Algorithmus, mit dem man den Klartext $\mathbf{a_1 a_2 \ldots a_n}$ mit Hilfe des Schlüsselworts $\mathbf{k_1 k_2 \ldots k_n}$ verschlüsselt.

Mit dem obigen 3. Kriterium können wir uns überzeugen, dass dieses Chiffriersystem perfekte Sicherheit bietet: Die Menge der Klartexte ist gleich der Menge der Geheimtexte, und diese wiederum ist gleich der Menge der Schlüssel. (Denn alle drei Mengen besehen aus allen Buchstabenfolgen der Länge n). Erst recht gilt dann

$$|\mathbf{F}| = |\mathbf{C}| = |\mathbf{M}|.$$

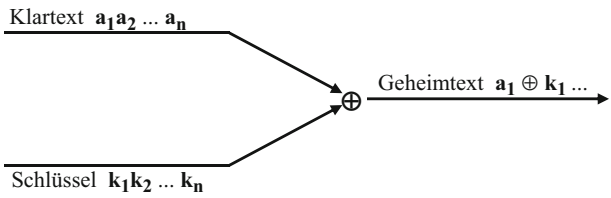


Abb. 3.5 Das One-Time-Pad

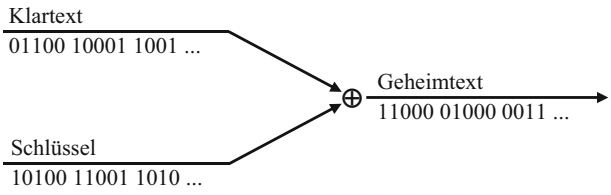


Abb. 3.6 Das One-Time-Pad

Damit ist die erste Voraussetzung bereits nachgeprüft. Um auch die zweite zu verifizieren, überlegen wir uns, dass es zu jedem Klartext $\mu = \mathbf{a_1 a_2 \ldots a_n}$ und zu jedem Geheimtext $\gamma = \mathbf{c_1 c_2 \ldots c_n}$ genau einen Schlüssel $\mathbf{k_1 k_2 \ldots k_n}$ gibt, der μ in γ überführt: Da die Verschlüsselung buchstabenweise erfolgt, genügt es, sich diese Eigenschaft für einen Klartextbuchstaben $\mathbf{a_i}$ und einen Geheimtextbuchstaben $\mathbf{c_i}$ klarzumachen – aber in diesem Fall ergibt sich die Behauptung direkt aus der Definition einer Vigenère-Chiffre.

Damit sind alle Voraussetzungen des 3. Kriteriums erfüllt. Daher wissen wir, dass das betrachtete System perfekt ist. Es ist unknackbar!

Dieses System wurde 1917 von dem amerikanischen AT&T-Ingenieur Gilbert S. Vernam (1890–1960) erfunden; üblicherweise wird es *One-Time-Pad* genannt. Denn in früheren Zeiten waren die Schlüsselbuchstaben auf den Blättern eines Abreißblocks geschrieben; sobald ein Schlüsselbuchstabe verwendet worden war, wurde das entsprechende Blatt abgerissen und vernichtet.

Heute wird das One-Time-Pad nicht mit Buchstaben, sondern mit Bits betrieben (siehe Abb. 3.6). Dann sind die $\mathbf{a_i}$ und $\mathbf{k_i}$ Bits, also Elemente der Menge $\{0, 1\}$. Der Geheimtext ergibt sich als $\mathbf{a_1} \oplus \mathbf{k_1}$, $\mathbf{a_2} \oplus \mathbf{k_2}$, ...; er wird durch bitweise binäre Addition erhalten, also durch Anwenden der Regeln

$$0 \oplus 0 = 0,\ 0 \oplus 1 = 1,\ 1 \oplus 0 = 1,\ 1 \oplus 1 = 0.$$

Wir werden dieser Operation im nächsten Abschnitt wieder begegnen.

Für die Sicherheit dieses Systems ist entscheidend, dass alle Folgen der Länge n mit derselben Wahrscheinlichkeit vorkommen. Mit anderen Worten: Die Schlüsselbits müssen zufällig gewählt werden. Am besten stellt man sich das so vor, dass eine faire Münze geworden wird (Kopf = 1, Zahl = 0). Es gibt natürlich schnellere Methoden. In der Praxis wird man eine physikalische Zufallsquelle benutzen und damit die Bits automatisch erzeugen.

Für diese Form von perfekter Sicherheit muss man – nicht unerwartet – einen hohen Preis bezahlen. Für ein traditionelles One-Time-Pad braucht man eine große Menge an Papier, das vor Angreifern absolut sicher aufbewahrt werden muss. Deshalb werden solche Systeme nur selten benutzt. Im zweiten Weltkrieg wurde es von der englischen Entschlüsselungstruppe von Bletchley

Park benutzt, um dem Premierminister die Nachrichten zu übermitteln, die von den Deutschen mit der Enigma verschlüsselt worden waren und die die Engländer geknackt hatten. So erreichten die Alliierten, dass die Deutschen bis Kriegsende nicht wussten, dass die Enigma geknackt worden war.

In den Zeiten des „kalten Krieges" sollen die Gespräche über den „heißen Draht" zwischen dem Weißen Haus und dem Kreml mit Hilfe eines (elektronischen) One-Time-Pads verschlüsselt worden sein. Allerdings wurde dieses System – denselben unbestätigten Gerüchten zufolge – nur von Wartungstechnikern zu Testzwecken benutzt.

Warum wird dieses unbezweifelbar perfekte System offenbar nur ganz selten eingesetzt? Um diese Frage zu beantworten, versetzen wir uns in die Lage des Empfängers. Dieser kann natürlich den Geheimtext bequem entschlüsseln: Dechiffrieren ist im Wesentlichen derselbe Vorgang wie Verschlüsseln. Wenn man Bits verwendet, dann ist das Entschlüsseln sogar ganz genau dasselbe wie das Verschlüsseln. Gut, er kann also bequem entschlüsseln – aber nur, wenn er den Schlüssel hat!

Wo soll da ein Problem liegen?

Das Problem besteht in der Tat darin, einen langen geheimen Schlüssel zu übermitteln. Wenn man ihn auf demselben Weg übermittelt wie den Klartext, so ist wegen der Länge des Schlüssels die Chance, gelesen zu werden, so groß wie bei einer (unverschlüsselten) Übermittlung des Klartexts. Man könnte die Meinung haben, bei einem solchen System könnte der Sender seinem Gegner den Klartext genauso gut frei Haus schicken. Dies ist nicht ganz richtig; denn für die Schlüsselübermittlung kann der Sender in der Regel sowohl die Art und Weise als auch den Zeitpunkt der Übermittelung selbst bestimmen, während dies bei der Nachricht meist nicht zutrifft.

Eine andere Art und Weise der Schlüsselübermittlung ist es, einen Kurier zu benutzen. Natürlich hat man dadurch das Risiko auf den anderen Übermittlungsweg verlagert: Der Bote lebt ziemlich gefährlich und vielleicht nicht sehr lange. Wichtiger ist, dass die Kommunikationspartner den Zeitpunkt des Schlüsselaustauschs selbst festlegen können, während der Zeitpunkt der Nachrichtenübermittlung weitgehend von äußeren Faktoren bestimmt wird. Am Beispiel des „heißen Drahtes" kann man das gut erkennen: Die Magnetbänder oder CDs, auf denen der Schlüssel gespeichert ist, können von Diplomaten in aller Seelenruhe auf den üblichen Wegen an Ort und Stelle geschafft werden. Wenn dann wirklich einmal eine Krisensituation auftritt, können die beiden Superchefs sofort miteinander telefonieren – jedenfalls werden sie nicht durch komplizierte Schlüsselverteilungsmechanismen daran gehindert.

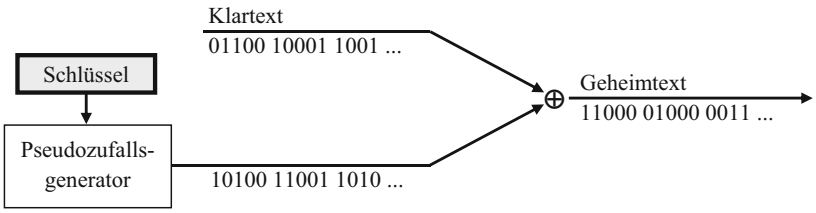


Abb. 3.7 Pseudozufallsgenerator

3.4 Schieberegister

Man kann das Problem der Schlüsselübermittlung beliebig lange diskutieren – Tatsache ist, dass es sich nicht nur um ein theoretisches Problem handelt, sondern dass die Schwierigkeit des Schlüsselaustauschs den praktischen Einsatz von Kryptosystemen stark beeinflusst. In Kap. 5 werden wir neue und sehr erfolgreiche Schlüsselaustauschprotokolle vorstellen. Es bleibt aber äußerst schwierig, im Fall des One-Time-Pads mit diesem Problem fertig zu werden.

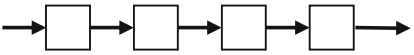
Ein wichtiger Ansatz, dieses Problem zu lösen, besteht darin, anstelle wirklich zufälliger Schlüsselfolgen nur *pseudozufällige* Folgen zu verwenden. Eine solche Folge sieht auf den ersten (und, wenn sie gut ist, auch auf den zweiten) Blick genauso aus wie eine echte Zufallsfolge, aber sie ist keine. Noch wichtiger: Eine Pseudozufallsfolge kann durch ganz wenige Daten bestimmt sein. Der Sender muss also nur diese wenigen Daten übermitteln; diese stellen den eigentlichen Schlüssel dar. Dann können beide Kommunikationspartner aus diesen Daten die Pseudozufallsfolgen berechnen und damit ver- und entschlüsseln (siehe Abb. 3.7). Das Problem der Schlüsselübermittlung ist dadurch natürlich nicht prinzipiell gelöst, aber doch erheblich entschärft.

Selbstverständlich muss man für diesen Vorteil bezahlen: Solche Systeme bieten keine perfekte Sicherheit. Genauer gesagt: Man hat die Sicherheit der Verschlüsselung auf die kryptologische Qualität des Pseudozufallsgenerators zurückgespielt. Man wird daher nach einem Kompromiss zwischen der erzielten Sicherheit und der Menge der geheim zu übermittelnden Daten suchen. In der zweiten Hälfte dieses Kapitels werden wir uns mit der Erzeugung von Pseudozufallsfolgen und einigen dabei auftretenden Problemen beschäftigen.

In der Praxis werden fast ausschließlich (pseudo-)zufällige 0,1-Folgen verwendet, und darauf wollen wir uns im Folgenden auch beschränken. Für kryptographische Zwecke hat sich die Erzeugung von Pseudozufallsfolgen mittels „Schieberegistern“ durchgesetzt. Dafür gibt es zwei wesentliche Gründe.

- Schieberegister können in Hardware sehr effizient realisiert werden.
- Die mathematische Theorie der Schieberegister hat sich in den letzten Jahrzehnten sehr entwickelt, und man versteht die Schieberegister heute ziemlich gut. Insbesondere hat man Methoden entwickelt, die in einem gewissen Sinne die Sicherheit von Schieberegisterfolgen quantifizierbar machen.

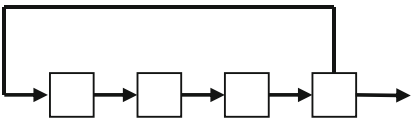
Ein Schieberegister sieht auf den ersten Blick ziemlich technisch aus (wie jedes mechanische Gerät); aber keine Angst: Alles ist gut zu verstehen.



Dies ist ein *Schieberegister* der *Länge* 4. Es besteht aus 4 hintereinander angeordneten Zellen, die jeweils ein Bit speichern können. Jede Zelle hat also den *Zustand* 0 oder 1. In einem gewissen Takt werden die Bits in Pfeilrichtung durch das Register hindurch geschoben (daher der Name). Das heißt: Die erste Zelle (das ist die Zelle ganz rechts) gibt ihr Bit ab und nimmt dafür das der zweiten Zelle auf; diese übernimmt das der dritten Zelle usw. Schließlich gibt die letzte Zelle (also die ganz links) ihr Bit an die vorletzte ab.

Würde das Schieberegister nur das tun, wäre die Sache ungerecht (denn die letzte Zelle müsste ihr Bit ohne Entgelt hergeben) und langweilig (denn nach 4 Takten wäre das Register leer).

Kurzum: Die letzte Zelle muss am Leben erhalten werden und irgendwoher ein Bit kriegen. Aber woher? Und welches Bit? Zunächst ist man versucht, wie folgt zu argumentieren: Die erste Zelle könnte ihr Bit nicht nur nach außen abgeben, sondern auch wieder in die letzte Zelle einspeisen:



Dies ist eine gute Idee; man muss *zurückkoppeln*! Allerdings ist der erste Vorschlag noch zu naiv. Denn das Schieberegister würde dann nur die ursprünglichen Zustände der Zellen periodisch ausstoßen. Eine solche Folge ist für kryptographische Zwecke zu durchsichtig.

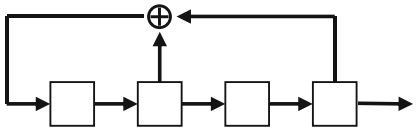
Die letzte Zelle könnte daher nicht nur von der ersten etwas bekommen, sondern auch von anderen. Fragt sich nur, von welchen Zellen und was? Das ist in der Tat die Frage.

Bei einem *linear rückgekoppelten Schieberegister* hat man diese Frage wie folgt beantwortet: Es wird eine gewisse Menge von Zellen ausgewählt. Deren Zustände werden addiert und als Rückkopplung in die letzte Zelle eingeführt. Die *Addition* der entsprechenden Bits wird dabei nach folgender Regel durchgeführt:

$$0 \oplus 0 = 0,\ 1 \oplus 0 = 1,\ 0 \oplus 1 = 1,\ 1 \oplus 1 = 0.$$

Der Mathematiker sagt zu dieser allereinfachsten Addition auch hochnäsig, sie erfolge *modulo* 2, während ein Techniker dieses $\oplus$ als *XOR*-Operation (*exclusive or*) kennt. Für jedes Bit b gilt: $b \oplus b = 0$ (denn es gilt $0 \oplus 0 = 0$ und $1 \oplus 1 = 0$).

Ein typisches linear rückgekoppeltes Schieberegister der Länge 4 sehen Sie auf dem folgenden Bild.

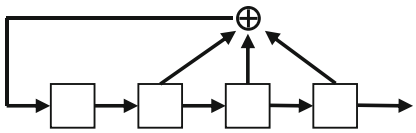


Dieses Schieberegister wird so zurückgekoppelt, dass die Inhalte der ersten und dritten Zelle addiert und das Ergebnis in die letzte Zelle eingespeist wird.

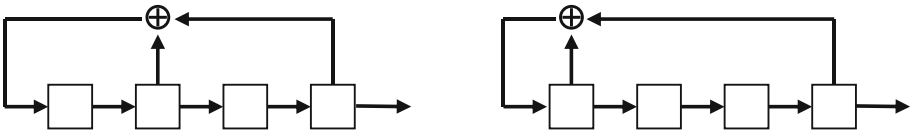
Die Zellen, von denen aus ein Pfeil nach oben geht, sind die ausgewählten Zellen, die die Rückkopplung der letzten Zelle liefern. Die anderen Zellen sind reine Arbeitszellen, denen das Wenige, was sie zum Zeitpunkt i erhalten haben, zum Zeitpunkt i + 1 wieder genommen wird.

Der Wert, der in die letzte Zelle eingespeist wird, heißt *Rückkopplung* (genauer gesagt: *Rückkopplungswert*). Das Schieberegister wird *linear* genannt, weil die Rückkopplungsfunktion so einfach wie möglich, nämlich pure Addition ist.

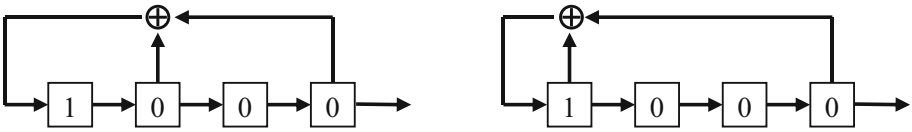
Selbstverständlich kann man für die Rückkopplung auch mehr als zwei Zellen in Betracht ziehen; bei dem folgenden Schieberegister ist die Rückkopplung die Summe der Inhalte der ersten drei Zellen:



Nun ist es an der Zeit, ein *Beispiel* zu studieren; wir betrachten gleich zwei Schieberegister der Länge 4:



Jetzt kann's losgehen! – Halt, wie denn? – Aha, jedes Schieberegister braucht noch eine *Initialisierung*. Das heißt: Jede Zelle muss in einen definierten Zustand versetzt werden. Wir wählen in beiden Fällen die gleiche Initialisierung, nämlich



Wir beobachten jetzt die Zustände der Reihe nach – bis sich ein Zustand wiederholt:

1	0	0	**0**
0	1	0	**0**
1	0	1	**0**
0	1	0	**1**
0	0	1	**0**
0	0	0	**1**
1	0	0	**0**

1	0	0	**0**
1	1	0	**0**
1	1	1	**0**
1	1	1	**1**
0	1	1	**1**
1	0	1	**1**
0	1	0	**1**
1	0	1	**0**
1	1	0	**1**
0	1	1	**0**
0	0	1	**1**
1	0	0	**1**
0	1	0	**0**
0	0	1	**0**
0	0	0	**1**
1	0	0	**0**

Die zugehörigen Pseudozufallsfolgen sind jeweils die Bits, die aus der ersten Zelle heraus geschoben werden. Beim ersten Schieberegister erhalten wir also die Folge

000101 000101 . . .

beim zweiten aber

000111101011001 000111101011001 . . .

Man beobachtet, dass sich die Folgen nach einer gewissen Zeit wiederholen: Sie sind *periodisch*; ihre *Periode* ist die Anzahl der Schritte bis zur Wiederholung. Zum Beispiel hat die erste Folge die Periode 6, während die zweite die Periode 15 hat. Es ist klar, dass man für ein Chiffriersystem eine möglichst lange Periode haben möchte – dann ist das Muster schwerer zu entdecken. Wir werden jedoch bald sehen, dass eine lange Periode einem Sicherheit eventuell auch nur vorgaukelt (siehe Abschn. 3.5).

Wie groß kann die Periode eines linearen Schieberegisters höchstens sein? Anders gefragt: Wie viele verschiedene Zustände kann ein Schieberegister annehmen? Da in unserem Beispiel jeder Zustand ein 4-Tupel aus Nullen und Einsen ist, kann man also auch fragen: Wie viele verschiedene 4-Tupel aus 0 und 1 gibt es? Diese Frage ist einfach zu beantworten: Die Anzahl aller 4-Tupel aus 2 Elementen ist $2^4 = 16$.

Aber ein Zustand spielt bei linearen Schieberegistern eine besondere, genauer gesagt: keine Rolle, und das ist der Zustand, bei dem alle Zellen den Wert 0 haben. Wenn das Schieberegister einmal diesen Zustand erreicht hat, so kommt es davon nie wieder los, und alles bleibt auf ewig so, wie es ist. Um also eine maximale Periode zu erreichen, darf der Nullzustand nicht auftreten. Deshalb hat ein lineares Schieberegister der Länge 4 höchstens die Periode $2^4 - 1 = 15$. Dass es Schieberegister einer solchen maximalen Periode gibt, zeigt das obige Beispiel.

Die Mathematiker haben Kriterien entwickelt, die einem sagen, wann ein lineares Schieberegister der Länge n maximale Periode hat. Insbesondere wurde herausgefunden, dass es für jede gewünschte natürliche Zahl n ein lineares Schieberegister mit maximaler Periode $2^n - 1$ gibt. Der interessierte Leser sei auf die Literatur [BP82, Rue86] verwiesen. Es sei ihm aber nachdrücklich empfohlen, selbst (ohne jede Theorie) sein eigenes lineares Schieberegister mit maximaler Periode zu konstruieren (siehe Übungsaufgabe 14).

3.5 Kryptoanalyse von linearen Schieberegistern

Wunderbar! Wir können also lineare Schieberegister zur Erzeugung von Pseudeozufallsfolgen für kryptographische Zwecke einsetzen. Das ist billig, lineare Schieberegister laufen sehr schnell – was wollen wir mehr?

Nun, es ist zwar unbestreitbar, dass die mit linearen Schieberegistern erzeugten Folgen hervorragende statistische Eigenschaften aufweisen; das gilt sogar für Folgen, die von vergleichsweise kurzen linearen Schieberegistern stammen. Aber kryptologisch betrachtet haben diese Folgen doch einen außerordentlich zwielichtigen Charakter. Das liegt daran, dass sie einem Known-plaintext-Angriff keinen nennenswerten Widerstand entgegenzusetzen haben.

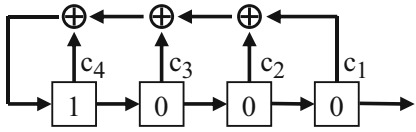
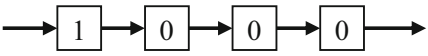


Abb. 3.8 Ein allgemeines Schieberegister der Länge 4

Stellen wir uns vor, der Kryptoanalytiker Mr. X weiß, dass zur Verschlüsselung ein lineares Schieberegister der Länge 4 benutzt wurde. Dann ist es seine Aufgabe, die Initialisierung und die genaue Gestalt der Rückkopplung zu finden. Stellen wir uns weiter vor, dass Mr. X eine Folge von 8 Klartextbits und die zugehörigen Geheimtextbits gefunden hat. Durch Addition dieser Bits erhält er daraus eine 8 Bit lange Teilfolge der Outputfolge des (unbekannten!) Schieberegisters. Nehmen wir an, diese Folge sei

$$00011110.$$

Machen wir uns nochmals die Funktionsweise eines Schieberegisters klar: Die ersten Outputbits sind die Bits, die im Schieberegister gespeichert waren, also die Initialisierungswerte. Weil unser Schieberegister *vier*stellig ist, liefern die ersten vier Bits (also die, die am weitesten links stehen) die Initialisierungswerte. Das heißt: Bevor unsere Folge ausgelesen wurde, hatte das Register folgenden Zustand:



Dies war die einfache Hälfte der Aufgabe von Mr. X. Nun muss er noch die Rückkopplungskoeffizienten ermitteln. Zu diesem Zweck malt er an jede Zelle prophylaktisch einen Draht zur Rückkopplung, versieht aber jeden Draht mit einem Koeffizienten $c_i \in \{0, 1\}$. Das soll bedeuten: Wenn $c_i = 1$ ist, dann ist der entsprechende Draht vorhanden (d. h. die entsprechende Zelle trägt zur Rückkopplung bei); im anderen Fall nicht. Schematisch sieht das Schieberegister also wie auf Abb. 3.8 aus.

Zum Beispiel gilt für die in Abschn. 3.4 betrachteten Schieberegister der Länge 4:$c_1 = 1$, $c_2 = 0$, $c_3 = 1$, $c_4 = 0$ bzw. $c_1 = 1$, $c_2 = 0$, $c_3 = 0$, $c_4 = 1$.

Jetzt kann Mr. X sein Ziel klar formulieren: Bestimme die Werte c_i! Dazu muss er sich einige Takte lang ansehen, was in den Zellen des Registers passiert. Nach einem Takt sieht es folgendermaßen aus:

Zustand der Zelle 4	Zustand der Zelle 3	Zustand der Zelle 2	Zustand der Zelle 1
c_4	1	0	0

Im nächsten Takt steht in der linke Zelle $c_4 \bullet c_4 \oplus c_3$. Dabei bedeutet „•“ die Multiplikation von Bits. Das heißt:

$$1 \bullet 1 = 1,\ 1 \bullet 0 = 0 \bullet 1 = 0 \bullet 0 = 0.$$

Für jedes Bit b gilt insbesondere $b \bullet b = b$ (denn es ist $0 \bullet 0 = 0$ und $1 \bullet 1 = 1$). Da $c_4 \bullet c_4 = c_4$ ist, gilt $c_4 \bullet c_4 \oplus c_3 = c_4 \oplus c_3$; also sieht die Sache nach einem weiteren Takt so aus:

Zustand der Zelle 4	Zustand der Zelle 3	Zustand der Zelle 2	Zustand der Zelle 1
$c_4 \oplus c_3$	c_4	1	0

Entsprechend geht es weiter. Nach dem dritten Takt ist es schon komplizierter. In Zelle 4 steht dann folgender Ausdruck, der glücklicherweise unter Zuhilfenahme der Regel „Punkt vor Strich“ vereinfacht werden kann:

$$\begin{aligned} &c_2 \oplus c_3 \bullet c_4 \oplus c_4 \bullet (c_4 \oplus c_3) = c_2 \oplus c_3 \bullet c_4 \oplus c_4 \bullet c_4 \oplus c_4 \bullet c_3 \\ &= c_2 \oplus c_3 \bullet c_4 \oplus c_3 \bullet c_4 \oplus c_4 \bullet c_4 = c_2 \oplus c_4. \end{aligned}$$

Zustand der Zelle 4	Zustand der Zelle 3	Zustand der Zelle 2	Zustand der Zelle 1
$c_2 \oplus c_4$	$c_4 \oplus c_3$	c_4	1

Und nach dem vierten Takt meint man schon, den Überblick verloren zu haben:

Zustand der Zelle 4	Zustand der Zelle 3	Zustand der Zelle 2	Zustand der Zelle 1
$c_1 \oplus c_3 \oplus c_4 \oplus c_3 \bullet c_4$	$c_2 \oplus c_4$	$c_4 \oplus c_3$	c_4

Nun brauchen wir aber nicht mehr zu rechnen! In den nächsten vier Takten werden nämlich die jetzt vorhandenen Zustände der Reihe nach (von rechts nach links) ausgelesen.

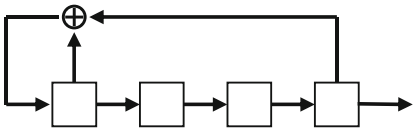
Was herauskommt, weiß Mr. X aber bereits: Es sind die restlichen vier Bits seiner abgehörten Folge! Daher weiß Mr. X, dass die folgenden Gleichungen gelten:

$$\begin{aligned} c_4 &= 1, \\ c_3 \oplus c_4 &= 1, \\ c_2 \oplus c_4 &= 1, \\ c_1 \oplus c_3 \oplus c_4 \oplus c_3 \bullet c_4 &= 0. \end{aligned}$$

Daraus erhält man ohne Schwierigkeit der Reihe nach

$$c_4 = 1, c_3 = 0, c_2 = 0 \quad \text{und} \quad c_1 = 1.$$

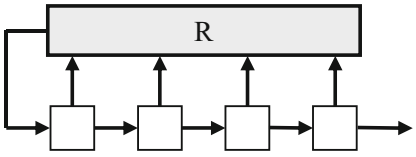
Das gesuchte Schieberegister ist also unser wohlbekanntes Register der Periode 15:



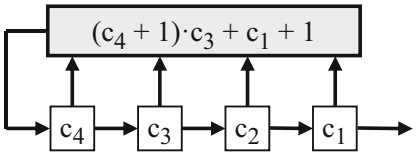
Allgemein kann man beweisen (siehe etwa [BP82, chapter 5]), dass Mr. X ein lineares Schieberegister der Länge n (also der maximalen Periode $2^n - 1$) knacken kann, wenn er auch nur 2n aufeinander folgende zusammengehörige Klartext/Geheimtext-Bits kennt. Das ist ein spektakuläres Ergebnis: Beispielsweise kann er eine Folge, die sich erst nach einer Million (genauer gesagt nach $1.048.575 = 2^{20} - 1$) Bits wiederholt, durch Kenntnis von schlappen 40 Outputbits rekonstruieren.

Was tun? Hat sich die ganze schöne Schieberregisteridee in Nichts aufgelöst? Vielleicht war es doch zuuu schön: Schieberegister lassen sich leicht realisieren und produzieren Pseudozufallsfolgen mit atemberaubender Geschwindigkeit (100.000.000 Bits pro Sekunde sind kein Problem). Diese schönen Eigenschaften möchte man sich erhalten. Aber man muss natürlich die eklatanten kryptographischen Schwächen vermeiden.

Das tut man, indem man *nichtlinear rückgekoppelte* (kurz: *nichtlineare*) Schieberegister verwendet. Bei diesen setzt man eine komplizierte (eben eine „nicht-lineare“) Rückkopplungsfunktion **R** ein. Beispielsweise kann man neben Additionen auch Multiplikationen betrachten.



Als einfaches Beispiel betrachten wir folgendes Schieberegister der Länge 4.



Man sieht relativ einfach, dass dieses Schieberegister nicht linear sein kann, da der Nullzustand in einen anderen Zustand überführt wird. (Man konsultiere auch die Übungsaufgaben 19 und 20). Natürlich bieten auch nichtlineare Schieberegister keine perfekte Sicherheit im Sinne von Abschn. 3.2, aber sie sind in der Regel besser als lineare Folgen.

Eine besonders raffinierte Methode ist der so genannte *Shrinking generator*. Dazu benutzt man zwei lineare Schieberegister, die im gleichen Takt laufen. Die Vorschrift lautet: Man nimmt nur Outputbits des zweiten Registers, aber nicht alle. Ein Outputbit des zweiten Registers wird nur dann genommen, wenn das erste Register gleichzeitig eine 1 liefert; wenn das erste Register eine 0 ausspuckt, wird das entsprechende Bit des zweiten Registers einfach weggeworfen.

Tatsächlich ist es so, dass die Mehrzahl der heute verwendeten Verschlüsselungsalgorithmen auf nichtlinearen Schieberegisterfolgen basieren; man spricht allgemein von *Stromchiffren*. Zum Beispiel ist der Algorithmus A5, der beim Mobilfunk zur Verschlüsselung benutzt wird, eine Stromchiffre (siehe S. 91). Man sollte aber nicht vergessen, dass die Algorithmen mit den höchsten Einschaltquoten Blockchiffren wie der Triple-DES und der AES sind.

Frage: *Kann man die kryptologische Qualität einer 0,1-Folge vernünftig messen?* Wir haben gesehen, dass ihre Periode sich dazu kaum eignet. In gewisser Weise bietet sich die so genannte „lineare Komplexität" als ein solches Maß an: Es ist klar, dass sich jede periodische 0,1-Folge (der Periode p) durch ein *lineares* Schieberegister erzeugen lässt: Wenn einem nichts Intelligenteres einfällt, nimmt man einfach ein beliebiges Schieberegister der Länge p und schreibt die fragliche Folge als Startzustand in die Zellen des Registers. Dann wird in den ersten p Takten die gewünschte Folge ausgelesen. In der Regel wird man ein Schieberegister kleinerer Länge finden, das ebenfalls als ein Teil seines Outputs die fragliche Folge hat: Jede Folge maximaler Periode $p = 2^n - 1$, die mit einem linearen Schieberegister der Länge n erzeugt wurde, dient als Beispiel mit $n \sim \log p$.

Die *lineare Komplexität* einer Folge ist definiert als die kürzeste Länge eines linearen Schieberregisters, das die vorgegebene Folge als (Teil ihrer) Outputfolge produziert. Je größer die lineare Komplexität einer Folge ist, desto besser ist diese Folge für Verschlüsselungszwecke geeignet. Der oben vorgestellte shrinking generator hat eine besonders hohe lineare Komplexität. Aber Vorsicht: Eine große lineare Komplexität ist nur eine notwendige Bedingung für die kryptologische Stärke einer Folge!

Es gibt Verfahren, mit denen man die lineare Komplexität einer vorgegebenen Folge abschätzen kann, nämlich der *Berlekamp-Massey-Algorithmus* ([Mas69], siehe auch etwa [FR88], [Rue86]). Darauf soll aber hier wirklich nicht mehr eingegangen werden.

3.6 Wie sicher ist Kryptographie?

Aus mathematischer Sicht macht alles einen guten Eindruck. Wir können perfekte Verschlüsselungssysteme konstruieren, wir können Schieberegisterfolgen analysieren, wir können die lineare Komplexität von Pseudozufallsfolgen berechnen und so weiter. Das ist die schöne Welt der Mathematik.

Aber sobald man ein Verfahren realisiert, sobald man es praktisch umsetzt, sobald man es in Software oder Hardware gießt, verlässt man das Reich der Mathematik. Man begibt sich in Gebiete, die längst nicht so gut zu beherrschen sind wie mathematische Algorithmen. Die meisten Angriffe auf Sicherheitssysteme zielen daher nicht auf die mathematischen Verfahren, sondern auf deren praktische Umsetzung. Meist (nicht immer!) ist nicht der Algorithmus die Schwachstelle, sondern zufällige oder unvermeidliche Schwachstellen bei der praktischen Umsetzung. Hier sind der Fantasie der Angreifer keine Grenzen gesetzt. Oft führen banalste Ideen zum Ziel. Einige Beispiele mögen das belegen:

Kann der geheime Schlüssel ausgelesen werden? Manchmal reicht ein einfacher Lesebefehl, und der Angreifer kennt den Schlüssel.

Wird überhaupt verschlüsselt? Die Software könnte so manipuliert sein, dass sie mir gegenüber zwar behauptet, die Nachrichten zu verschlüsseln, in Wirklichkeit aber alles im Klartext schickt.

Diese beiden Angriffe kann man nur verhindern, wenn man die Software vertrauenswürdig ist, wenn sie also garantiert genau die Funktionalitäten ausführt, die von ihr verlangt werden – nicht weniger, aber vor allem auch nicht mehr! Das korrekte Funktionieren einer Software nachzuweisen ist allerdings enorm schwierig.

Ist der Schlüssel beim One-Time-Pad wirklich eine Zufallsfolge? Selbst wenn man die Folge der Schlüsselbits durch einen grundsätzlich sehr guten physikalischen Prozess erzeugt, at der Angreifer eventuell die Möglichkeit, auf die Zufallsquelle einzuwirken und so beispielsweise dafür zu sorgen, dass die Mehrzahl der Bits gleich Null ist.

Sie sehen: Alles ist möglich! Und alles, was möglich ist, wird auch irgendwann einmal angewendet werden. Die bekannt gewordenen Angriffe der NSA und anderer Geheimdienste basieren zu einem großen Teil, wenn auch nicht ausschließlich auf solchen „banalen“ Angriffen.

Während auf der mathematischen Ebene die Entwickler von Algorithmen (die „Guten“) einen prinzipiellen Vorsprung vor den Angreifern (den „Bösen“) haben, herrscht auf der Ebene der Realisierung von Sicherheitsverfahren eher Waffengleichheit. Das bedeutet, dass es keine Sicherheit „für immer“ gibt, sondern dass man stets auf der Hut sein muss.

3.7 Übungsaufgaben

1 Konstruieren Sie einige Chiffriersysteme, indem Sie Klartexte, Geheimtexte und Transformationen geeignet definieren.

2 Zählen die Häufigkeiten der Buchstaben einer fremden Sprache (englisch, italienisch, ungarisch). Stellen Sie die Hauptunterschiede zu den entsprechenden Häufigkeiten der deutschen Sprache fest.

3 Wie viele Buchstabenkombinationen der Länge 3 gibt es:
   (a) wenn man alle möglichen Kombinationen zulässt,
   (b) wenn man nur sinnvolle deutsche Wörter betrachtet? [Hinweis: Über 50.]

4 Ein Betriebssystem akzeptiert Authentifikation der Benutzer (vergleiche Kap. 4) Passwörter, die aus genau 4 alphanumerischen Zeichen bestehen.
   (a) Wie viele Passwörter gibt es insgesamt?
   (b) Wie viele Passwörter sind möglich, wenn jedes mindestens eine Ziffer und mindestens einen Buchstaben enthalten muss?
   (c) Vergleichen Sie die in (a) und (b) erhaltenen Zahlen mit der Anzahl der Ihnen bekannten weiblichen Vornamen.
   (d) Welches der Systeme (a) oder (b) würden Sie wählen, wenn Sie eine große Anzahl tatsächlich benutzter Passwörter anstreben?

5 Das Beispiel aus Abb. 3.1 kann durch weitere Transformationen ergänzt werden. Geben Sie mindestens vier solche Transformationen an.

6 (a) Wie viele umkehrbare Transformationen von **M** nach **C** gibt es, wenn $|\mathbf{M}| = 3$ und $|\mathbf{C}| = 5$ ist?
   (b) Lösen Sie das entsprechende Problem für $|\mathbf{M}| = 1066$ und $|\mathbf{C}| = 2001$.

7 Beweisen Sie: Ein Chiffriersystem **S** ist perfekt, falls für alle Klartexte μ gilt

$$p_\mu(\gamma) = p(\gamma) \quad \text{für alle Geheimtexte } \gamma.$$

8 Geben Sie ein Chiffriersystem **S** mit

$$|\mathbf{F}| \geq |\mathbf{C}| \geq |\mathbf{M}|$$

an, das nicht perfekt ist.

9 ▶Schauen Sie in einem Statistik-Buch (etwa [Hen]) die Bayes'sche Formel nach und beweisen Sie damit das 3. Kriterium. [Hinweis: Verwenden Sie Aufgabe 7.]

10 Überzeugen Sie sich, dass die folgenden Verfahren Zufallsfolgen liefern:
   (a) Werden eines fairen Würfels. Falls man ungerade Zahlen in 1 und gerade Zahlen in 0 übersetzt, erhält man eine binäre Zufallsfolge.
   (b) Drehen eines Roulette-Rads. Dies liefert eine Zufallsfolge, bei der jedes Element 37 Werte annehmen kann (die Zahlen 0,1, ..., 36).

11 Haben Sie schon mal einen 4-seitigen, 8-seitigen, 12-seitigen oder 20-seitigen fairen „Würfel" gesehen. [Hinweis: Schauen Sie in einem Geometriebuch (etwa [Cox63] oder [Beu86]) unter „platonische Körper" nach.]

12 (a) Konstruieren Sie ein lineares Schieberegister der Länge 4, das einen Nicht-Null-Zustand in den Null-Zustand überführt.
   (b) Konstruieren Sie ein nichtlineares Schieberegister der Länge 4, das den Null-Zustand in einen Nicht-Null-Zustand überführt.

13 Überzeugen Sie sich von folgender Tatsache: Wenn ein lineares Schieberegister maximale Periode hat, dann geht aus der ersten Zelle (ganz rechts) ein Pfeil nach oben heraus (das heißt, diese Zelle trägt aktiv zur Rückkopplung bei).

14 Konstruieren Sie lineare Schieberegister der Längen 3, 5, 6 mit maximalen Perioden.

15 Die folgende Outputfolge wurde von einem linearen Schieberegister der Länge 5 erzeugt:

0000100011.

Rekonstruieren Sie das Schieberegister.

16 Jemand behauptet, die folgende Folge wäre von einem linearen Schieberegister der Länge 4 erzeugt worden:

00001000.

Kommentieren Sie diese Behauptung.

17 Geben Sie sich eine beliebige binäre Folge der Länge 10 vor und versuchen Sie, ein lineares Schieberegister der Länge 5 zu berechnen, das diese Folge als (Teil seiner) Outputfolge hat.

18 (a) Konstruieren Sie für ein Schieberegister der Länge 4 eine nichtlineare Rückkopplungsfunktion.
   (b) Überprüfen Sie, ob die Outputfolge, die von Ihrem in (a) konstruierten Schieberegister stammt, von einem linearen Schieberegister der Länge 4 erzeugt worden sein könnte.

19 Zeigen Sie, dass das Schieberegister, von dem behauptet wurde, es sei nichtlinear (das letzte Bild von Abschn. 3.5) tatsächlich nicht äquivalent

zu einem linearen Schieberegister der Länge 4 ist. [Zum Beispiel könnten Sie zeigen, dass die Methode von Mr. X zu einander widersprechenden Gleichungen führt.]

20 Welche der folgenden Rückkopplungsfunktionen beschreiben ein nichtlinear rückgekoppeltes Schieberegister?

$$c_4c_1 + c_3 + 1,$$
$$(c_4 + 1)c_3 + (c_3 + 1)c_2 + c_1 + 1,$$
$$(c_4 + 1)c_3 + (c_3 + 1)c_2 + c_1,$$
$$(c_4 + 1)c_3 + (c_4 + 1)(c_3 + 1)c_2 + c_1 + 1.$$

21 Gibt es ein Schieberegister der Länge 4, das eine Outputfolge der Periode 16 hat? [Betrachten Sie die Beispiele der vorigen Aufgabe.]

22 (a) Gibt es ein lineares Schieberegister der Länge 5, dessen Outputfolge 5 aufeinander folgende Nullen besitzt?
(b) Welche lineare Komplexität hat die Folge 100000?
(c) Gibt es eine binäre Folge der Länge 1000, deren lineare Komplexität 1000 ist?

Zusatzinformation: Wenn man über die Länge des Schlüssels eines kryptographischen Algorithmus spricht, hört man immer wieder den Einwand: „Gut, es mag ja sehr viele Schlüssel geben – aber, *wenn ich Glück habe,* kann ich den Schlüssel doch raten!"

Diese Aussage sollte genau analysiert werden. Zunächst muss ich zugeben, dass der Einwand, in einem theoretischen Sinn, richtig ist. Zum Beispiel ist die Wahrscheinlichkeit, einen 64-Bit Schlüssel richtig zu raten, gleich $1/2^{64}$, eine Zahl, die – darüber kann kein Zweifel bestehen – größer als Null ist.

Um ein Gefühl dafür zu entwickeln, um *wie viel* (genauer gesagt: *um wie wenig*) diese Zahl größer als Null ist, vergleichen wir sie mit anderen Größen. Zunächst rechnen wir die Anzahl der Schlüssel in eine Zehnerpotenz um; es ist $2^{64} \approx 1{,}84 \cdot 10^{19}$.

Mit anderen Worten: Den Schlüssel zu raten, ist gleichwertig dazu, aus einer Menge von mehr als 10^{19} Elementen (das sind 10 Trillionen) ein spezielles Element auf Anhieb zu treffen. Können Sie sich diese Zahl vorstellen? Vielleicht helfen Ihnen die folgenden Vergleiche.

- Die Weltbevölkerung beträgt derzeit etwa 6 Milliarden, also $6 \cdot 10^9$ Menschen. Da es ungefähr genau so viele Männer wie Frauen gibt, kann man

etwa $(3 \cdot 10^9) \cdot (3 \cdot 10^9) = 9 \cdot 10^{18}$ Paare bilden. Das heißt: Die Anzahl aller möglichen verschiedengeschlechtlicher Paare (übrigens auch die Anzahl aller gleichgeschlechtlichen Paare) ist kleiner als die Anzahl aller 64 Bit-Schlüssel.

- In einem Tropfen, also etwa einem Kubikmillimeter Wasser gibt es etwa 10^{19} Moleküle. Also ist das Raten des Schlüssels etwa genau so schwierig wie auf Anhieb aus einem Tropfen Wasser ein spezielles Molekül zu fischen. Dieser Vergleich wird noch viel eindrucksvoller, wenn man Schlüssel betrachtet, die aus 256 Bit bestehen. Die Astronomen haben herausgefunden, dass das Universum nur ca. 10^{77} Elementarteilchen enthält. Da $2^{256} \approx 1{,}15 \cdot 10^{77}$ ist, gibt es also mehr 256 Bit-Schlüssel als Elementarteilchen im Universum. Daher ist klar, dass eine systematische Suche ins Leere gehen muss.
- Die Anzahl der Möglichkeiten, beim „6 aus 49"-Lotto einen Volltreffer zu erzielen, ist $13.983.816 \approx 10^7$. Das ist nämlich die Anzahl der Möglichkeiten, 6 Zahlen aus einer Menge von 49 auszuwählen. Das bedeutet: Die Chance, einen 64 Bit-Schlüssel zu raten, ist genau so gut, wie im Lotto 6 Richtige zu haben – und zwar in dieser Woche, in der kommenden und in der übernächsten!

Moral: Wenn Sie durch das Raten eines kryptographischen Schlüssels einen Gewinn von nur ein paar Millionen Euro erwarten, dann ist es sinnvoller, Ihr Geld beim Lotto zu verpulvern!

4

Daten mit Denkzettel oder Ein Wachhund namens Authentifikation

Der König, als er den Haspel auf dem Grund fand, ließ Allerleirauh rufen.
Da erblickte er den weißen Finger und sah den Ring,
den er im Tanze ihr angesteckt hatte.
Da ergriff er sie an der Hand und hielt sie fest,
und als sie sich losmachen und fortspringen wollte,
tat sich der Pelzmantel ein wenig auf,
und das Sternenkleid schimmerte hervor.
Der König fasste den Mantel und riss ihn ab.
Da kamen die goldenen Haare hervor,
und sie stand da in voller Pracht
und konnte sich nicht länger verbergen
(Brüder Grimm, Allerleirauh).

4.1 Motivation

In der ersten Hälfte dieses Buches haben wir Methoden vorgestellt, die gegen einen *passiven Angriff* helfen können: durch Verschlüsselung können Daten für Außenstehende unlesbar gemacht werden. Das Thema dieses Kapitels sind Methoden gegen einen *aktiven Angriff* (siehe Abb. 4.1).

Es ist klar, dass der Angreifer Mr. X wesentlich größeren Schaden anrichten kann, wenn er Daten nicht nur passiv lesen, sondern sogar aktiv verändern kann. Tatsächlich wird bei den meisten heutigen Anwendungen der Kryptolo-

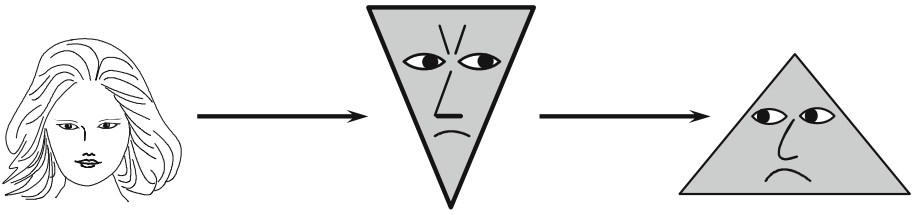


Abb. 4.1 Der aktive Angreifer

A. Beutelspacher, *Kryptologie*, DOI 10.1007/978-3-658-05976-7_4,

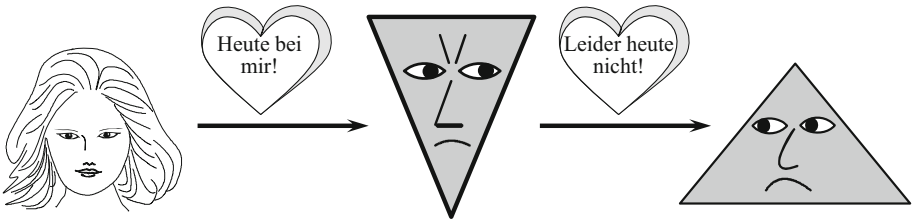


Abb. 4.2 Nachrichtenintegrität

gie die Authentizität der Daten gefordert und nicht ihre Geheimhaltung. Es ist üblich, drei Typen von Problemen zu unterscheiden, die sich aus verschiedenen Varianten des aktiven Angriffs ergeben. Die entsprechende „Sicherheitsarchitektur" wurde von der *International Standards Organisation* (ISO) in dem „Security Addendum zum OSI-Referenz Modell" [ISO] niedergelegt.

Zum einen fragt man danach, ob die Nachricht ohne Veränderung oder Verfälschung übermittelt wurde; es handelt sich um die Forderung nach *Nachrichtenintegrität* (Abb. 4.2).

Wenn Mr. X in der Lage ist, Nachrichten zu verändern, kann man bestenfalls darauf hoffen, dass der Empfänger eine eventuelle Veränderung *bemerkt*. Er muss entscheiden können, ob die Nachricht verändert wurde oder nicht. Der zweite Typ von Angriff ist dem ersten ähnlich; hier liegt der Akzent auf der Frage, ob der Empfänger sicher sein kann, dass die Nachricht wirklich von der angeblichen Senderin stammt. Sie stellt sich die Frage, ob sie später beweisen kann, die Nachricht wirklich geschickt zu haben oder ob sie dies erfolgreich ableugnen kann. Man spricht von *Nachrichtenauthentifikation* (Abb. 4.3).

Hier ist die Aufgabe, Werkzeuge zu entwickeln, die es dem Empfänger ermöglichen, sich zweifelsfrei davon zu überzeugen, dass die Nachricht von dem angegebenen Absender stammt.

Die letzte Variante ist die Benutzerauthentifikation: Kann eine Person ihre Identität beweisen? Der Empfänger braucht ein Mittel, um sich davon zu

Abb. 4.3 Nachrichtenauthentifikation

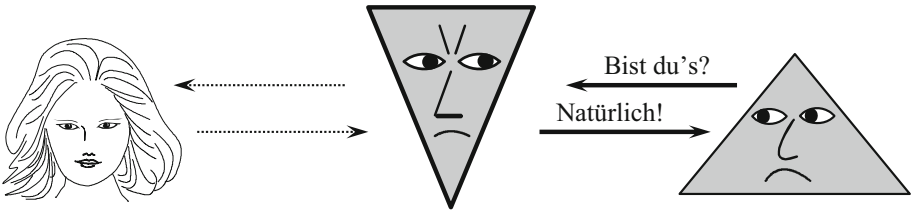


Abb. 4.4 Benutzerauthentifikation

überzeugen, dass er wirklich mit der Person kommuniziert, mit der er verbunden zu sein glaubt (Abb. 4.4).

In diesem Kapitel werden wir verschiedene Methoden zur Lösung dieser Probleme erörtern. In Abschn. 4.3 stellen wir Chipkarten vor, ein ideales Medium zur Realisierung von Integrität und Authentizität.

Zuerst illustrieren wir die eingeführten Begriffe und gewisse Lösungsmöglichkeiten an einem Beispiel aus dem wirklichen Leben.

Es waren einmal eine junge Dame und ein junger Herr, die schon lange befreundet waren. Nachdem sie sich jahrelang, wie damals üblich, nur Briefe geschrieben hatten, kam unaufschiebbar die Zeit für ein Rendezvous. Da die beiden sehr diskret waren, hatten sie sich auch niemals ein Bild geschickt; daher wusste keiner, wie der andere aussah. Wie konnten sie sich ohne Verwechselungsgefahr erkennen? Die beiden Verliebten in spe waren nicht auf den Kopf gefallen und verabredeten Identifikationszeichen: *Er* stellte sich (nicht sehr originell) mit einem Blumenstrauß zum Treffpunkt ein, während *sie* sich entschloss (Sie mögen es glauben oder nicht!), ein Mathematikbuch unterm Arm zu tragen. Ihr Treffen verlief ohne jedes Ereignis, das uns interessieren müsste; aber ich kann Ihnen versichern: Es war Liebe auf den ersten Blick.

Wie man sich vorstellen kann, wurden ihre Briefe danach immer intensiver, und beide waren sehr darauf bedacht, dass kein Brief in die Hand von Konkurrenten oder Neidern fiel. Wie konnte er sicher sein, dass die süßen Sätze, die er las, wirklich diejenigen waren, die sie mit ihrer ganzen Liebe und Hingabe ersonnen hatte? Ganz einfach: Sie versah jeden Brief mit einem Tropfen jenes Parfüms, das er so gut kannte. Wie konnte er sicher sein, dass keiner der Neider den Brief geöffnet und den Inhalt geändert hatte, um ihn aufs Glatteis zu führen? Auch in diesem Punkt war er sicher: Sie schrieb nämlich mit Tinte (und nicht nur mit Bleistift), und vor allem war ihre Handschrift so einzigartig schön, dass ihm jeder Fälschungsversuch sofort aufgefallen wäre.

In dieser kleinen Geschichte finden wir Techniken für Benutzerauthentizität (geheime Erkennungszeichen), Nachrichtenauthentizität (Parfüm) und Nachrichtenintegrität (Tinte, Handschrift).

Nun aber genug der Romantik; zurück zum wirklichen Leben!

4.2 Integrität und Authentizität

Die Mechanismen zur Erreichung der Integrität bzw. Authentizität sind die gleichen, sie unterscheiden sich nur darin, auf welchen Teil der Gesamtnachricht sie angewandt werden. Deshalb sprechen wir im Folgenden nur von Authentifikationsmechanismen (im allgemeinen Sinn).

4.2.1 Mac 'n Data

Wir erinnern uns daran, dass es bei Nachrichtenintegrität und -authentizität darum geht, Methoden zu entwickeln, die es dem Empfänger ermöglichen zu entscheiden, ob die Nachricht unversehrt und authentisch bei ihm angekommen ist. Dazu braucht der Empfänger etwas, mit dem er die Nachricht prüfen kann: Er benötigt zusätzliche Information vom Sender.

Ein solcher Informationsblock wird *kryptographische Prüfsumme*, *kryptographischer Fingerabdruck* oder *Message Authentication Code*, kurz *MAC* genannt. Wir werden den letzteren Ausdruck verwenden. Das Protokoll zur Erzeugung und Verifizierung eines MAC ist in Abb. 4.5 zu sehen.

Das Verfahren beruht auf einem geheimen Schlüssel k, der sowohl dem Sender als auch dem Empfänger bekannt ist, und einem kryptographischen Algorithmus F, den wir noch diskutieren werden.

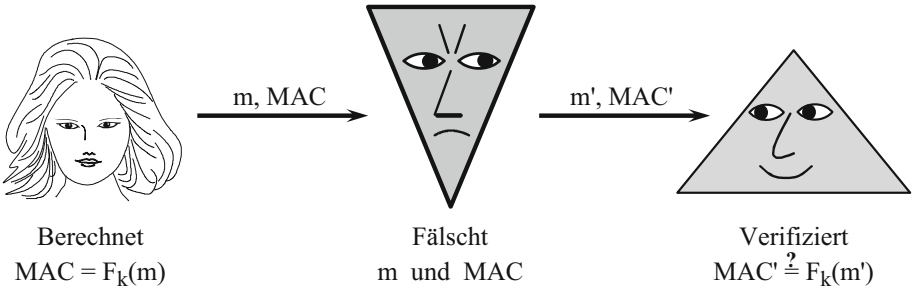


Abb. 4.5 MAC-Erzeugung und -Verifizierung

Der Sender schickt nicht nur die nackte Nachricht m, sondern zusätzlich den zugehörigen MAC; dieser wird mit Hilfe des geheimen Schlüssels k durch den Algorithmus F aus der Nachricht m wie folgt berechnet:

$$\mathrm{MAC} = \mathrm{F_k(m)}.$$

Beachten Sie, dass m unverschlüsselt gesendet wird, da das Ziel des Senders nicht die Geheimhaltung der Nachricht, sondern deren Authentizität ist. Wenn zusätzlich Vertraulichkeit erreicht werden soll, müssen m und MAC zusätzlich verschlüsselt werden. Systemseitig werden keine physikalischen Vorkehrungen getroffen, Mr. X davon abzuhalten, m und MAC zu m′ und MAC′ zu verändern.

Nun kommt der Empfänger an die Reihe. Sein Interesse ist es zu erfahren, ob die empfangene Nachricht mit der gesandten übereinstimmt und ob sie wirklich von dem behaupteten Absender stammt. Um das zu überprüfen, imitiert er die Prozedur des Senders: Er wendet den Algorithmus F mit dem Schlüssel k auf die empfangene Nachricht m′ an und überprüft, ob das Ergebnis mit dem erhaltenen Message Authentication Code MAC′ übereinstimmt.

Falls $\mathrm{F_k(m')} \neq \mathrm{MAC'}$ ist, weiß der Empfänger, dass „etwas passiert“ ist; demzufolge wird er die Nachricht nicht als authentisch akzeptieren und also ablehnen. Wenn aber $\mathrm{F_k(m')} = \mathrm{MAC'}$ ist, so kann er ziemlich sicher sein, dass die Nachricht nicht verändert wurde. Natürlich hängt diese Gewissheit in starkem Maße von der Qualität des Algorithmus F und dem Umfang der Menge aller möglichen Schlüssel k ab.

Algorithmus 4.1: Message Authentication Code

Erzeugen eines MAC:	Aus der Nachricht m berechnet der Sender mit Hilfe eines Algorithmus F unter einem geheimen Schlüssel k den Message Authentication Code $\mathrm{MAC} = \mathrm{F_k(m)}$.
Verifizieren des MAC:	Der Empfänger erhält eine Nachricht m′ zusammen mit einem potenziellen MAC′. Er überprüft, ob $\mathrm{F_k(m')} = \mathrm{MAC'}$ ist. Er akzeptiert die Nachricht genau dann, wenn diese Gleichung gilt.

Die dem MAC-Mechanismus zugrundeliegenden Vorstellungen sind die folgenden:

- Der Betrug von Mr. X wird verhindert, da er den Schlüssel k nicht kennt: Er müsste nämlich den seiner Nachricht m′ entsprechenden MAC berechnen.

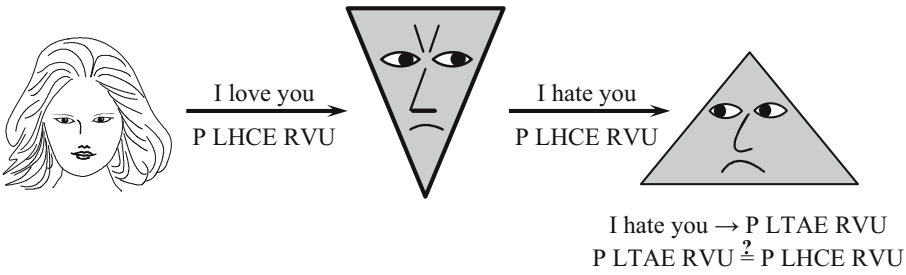


Abb. 4.6 Ein einfacher MAC-Algorithmus

- Der Empfänger kann nur *erkennen*, ob die Nachricht unversehrt und authentisch ist; im negativen Fall hat er keine Möglichkeit, die Originalnachricht zu rekonstruieren. Das bedeutet, dass in einem solchen Fall die Nachricht aufs Neue übertragen werden muss.
- Der MAC-Mechanismus ist eine Methode, um Integrität und Authentizität zu erreichen. Wir haben bereits bemerkt, dass Integrität erkannt werden kann. Wenn die Verifikation positiv verläuft, ist der Empfänger auch von der Authentizität der Nachricht überzeugt, da der Sender die einzige andere Instanz ist, die den geheimen Schlüssel k kennt.

Frage: Welche Algorithmen F kann man zur MAC-Berechnung verwenden? Die erste Antwort ist einfach: Man benutze einfach einen Verschlüsselungsalgorithmus, wobei MAC der Geheimtext ist, der dem Klartext m entspricht. Wir illustrieren dies mit Hilfe eines Vigenère-Algorithmus in Abb. 4.6.

Abgesehen davon, dass ein solch schwacher Algorithmus nicht empfehlenswert ist, hat dieser Vorschlag den Nachteil, dass die übertragenen Daten doppelt so lang sind wie die „eigentliche Nachricht". Natürlich verlängert jeder MAC die Nachricht, aber man möchte die Länge dieses Zusatzblocks doch innerhalb vernünftiger Grenzen halten.

In der Praxis verwendet man zur MAC-Berechnung auch einen Verschlüsselungsalgorithmus, aber nicht direkt, sondern im so genannten *Cipher-Block-Chaining-Modus*.

Stellen wir uns einen Verschlüsselungsalgorithmus f vor, der Klartextblöcke von n Zeichen unter einem Schlüssel k auf Geheimtextblöcke von ebenfalls n Zeichen abbildet, mit anderen Worten: eine Blockchiffre. Typische Zahlen sind $n = 64$ oder $n = 128$. Die in Abschn. 1.8 angesprochenen Algorithmen DES und AES sind die in der Praxis verwendeten Beispiele.

Um den MAC zu berechnen, wird die Nachricht m in Blöcke $m_1, m_2, \ldots, m_s$ der Länge n aufgeteilt. Dann wendet man f auf m_1 an und erhält den ersten Geheimtextblock $c_1 = f_k(m_1)$. Es geht nun aber nicht so weiter, wie man zunächst denken könnte, sondern bevor man f auf m_2 anwendet, addiert man c_1

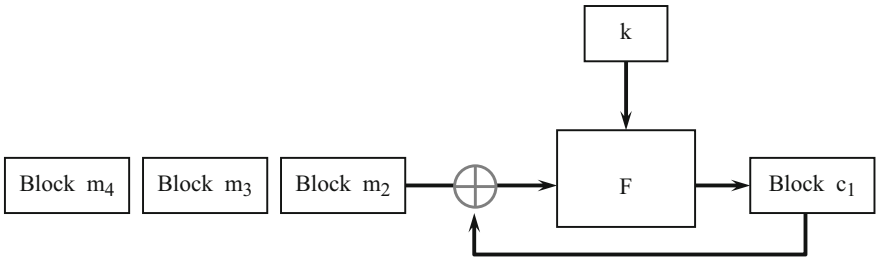


Abb. 4.7 MAC mittels des Cipher-Block-Chaining-Modus

zu m_2, und berechnet dann $c_2 = f_k(c_1 \oplus m_2)$. (Das Symbol $\oplus$ bezeichnet wie in Kap. 3 die Addition modulo 2. Das heißt, das erste Bit von c_1 wird zum ersten Bit von m_2 addiert, wobei die Regel $1 \oplus 1 = 0$ zu beachten ist. Dann werden die zweiten Bits addiert und so weiter.) Im dritten Schritt wird $c_3 = f_k(c_2 \oplus m_3)$ berechnet, und so weiter (siehe Abb. 4.7).

Der letzte Output $c_s = f_k(c_{s-1} \oplus m_s)$ wird dann als MAC genommen. Ein so berechneter MAC hat die folgenden Vorteile:

- Der MAC hat eine feste Länge n unabhängig von der Länge der Nachricht.
- Der MAC hängt von allen Blöcken der Nachricht ab.

Da alle möglichen Nachrichten auf einen MAC fester Länge komprimiert werden, haben viele Nachrichten denselben MAC. Dies stellt kein Problem für den Empfänger dar, da dieser die Originalnachricht nicht aus dem MAC rekonstruieren muss.

Nicht jeder Algorithmus ist geeignet, Mr. X vor unüberwindliche Probleme zu stellen. Ein Algorithmus zur MAC-Berechnung sollte die folgenden Eigenschaften haben.

(1) Es sollte praktisch unmöglich sein, zu einem gegebenen MAC eine passende Nachricht m zu finden. Wenn diese Eigenschaft gilt, nennt man den Algorithmus eine *Einweg-Hashfunktion*. „Praktisch unmöglich" bedeutet, dass mit heutigen Methoden und heutigen Rechnern die Lösung des Problems viel zu lange, etwa einige Jahrhunderte, dauern würde.
(2) Es sollte praktisch unmöglich sein, zwei verschiedene Nachrichten m und m' zu finden, die denselben MAC haben. Eine Einweg-Hashfunktion, die die Bedingung (2) erfüllt, wird *kollisionsresistent* genannt.

Der DES-Algorithmus und der AES-Algorithmus, die wir in Kap. 1 erwähnt haben, erfüllen die Kriterien (1) und (2) in hinreichend gutem Maße; jedenfalls werden sie im Cipher-Block-Chaining-Modus allgemein zur MAC-Berechnung benutzt.

4.2.2 Benutzerauthentifikation

Eine Grundaufgabe der Sicherheitstechnik ist es, Personen verlässlich zu identifizieren, das heißt zu authentifizieren. Früher war dies ein Prozess zwischen zwei Menschen, der heute zu einem Prozess zwischen Mensch und Rechner erweitert ist. Hierbei gibt es natürlich Probleme; aber wir werden sehen, dass auch mit einem Rechner eine sehr gute Benutzerauthentifikation möglich ist. Zunächst erinnern wir daran, wie das Problem im zwischenmenschlichen Bereich gelöst wird. Man kann einen Menschen an folgenden Merkmalen erkennen:

- Eine Person kann durch charakteristische *Eigenschaften* identifiziert werden.
- Eine Person kann durch *Besitz* identifiziert werden.
- Eine Person kann durch *Wissen* identifiziert werden.

Der erste Mechanismus wird im täglichen Leben so häufig verwendet, dass er uns fast nicht bewusst ist: Wir erkennen unsere Bekannten an ihrem Aussehen, ihrer Stimme, ihrem Gang, usw. In gewissen Situationen (in die *unsere* Freunde natürlich nie geraten) werden Fingerabdrücke zur verlässlichen Identifizierung herangezogen. Um ein besonders hohes Maß an Sicherheit zu gewährleisten, werden unsere Fingerabdrücke in unseren Reisepässen gespeichert.

Die beiden anderen Methoden werden nur in speziellen Situationen benutzt. Beim Grenzübergang muss man sich mit einem Ausweis identifizieren; wenn man mit Kreditkarte bezahlt, wird die Identität auch durch den Besitz der Kreditkarte bewiesen.

Identifizierung durch Wissen geschieht nur ganz selten – obwohl der Mechanismus schon aus ältesten Zeiten bekannt ist (vergleiche im Alten Testament *Richter 12, Vers 6*). Soldaten müssen die aktuelle Parole kennen, um sich zu identifizieren. Ein anderes Beispiel ist das Folgende: Wenn die Polizei wissen möchte, ob eine entführte Person noch am Leben ist, stellt sie Fragen, die nur diese Person beantworten kann.

Bei einem Authentifikationsprozess zwischen einem Menschen und einem Rechner ist die Lage ganz anders. Authentifikation durch Wissen ist der am einfachsten zu realisierende Mechanismus; Authentifikation durch Besitz ist ebenfalls möglich. Im Gegensatz dazu ist Authentifikation durch Eigenschaften (durch so genannte *biometrische Verfahren*) ziemlich komplex; solche Verfahren werden bis auf weiteres nur bei Anwendungen, die sehr hohe Sicherheit fordern, eingesetzt werden. Daher werden wir uns hier ausschließlich mit Methoden beschäftigen, die auf Wissen oder auf Besitz der sich authentifizierenden Person aufbauen. Bei diesen Methoden hat ein Benutzer ein Geheimnis,

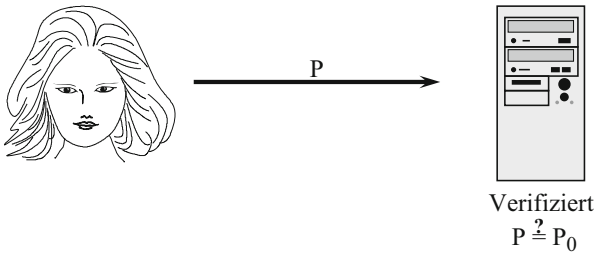


Abb. 4.8 Der naive Passwortmechanismus

und der Rechner will sich davon überzeugen, dass die Person das ihr entsprechende Geheimnis hat.

Bei jeder Authentifikation durch Wissen geht es darum, dem Computer gegenüber nachzuweisen, ein bestimmtes Geheimnis zu kennen. Die verschiedenen Verfahren unterschieden sich nur darin, wie dieser Nachweis erfolgt. Insbesondere ist die Frage, ob man sein Geheimnis direkt übermitteln muss und ob die Gegenstelle das Geheimnis ebenfalls kennen muss. Das Passwortverfahren ist unter dieser Fragestellung das primitivste Authentifikationsverfahren.

Passwörter

Die klassische Methode, mit der sich eine Person gegenüber einer Maschine authentifiziert, beruht auf Passwörtern. Ein *Passwort* ist eine beliebige Folge von Zeichen, üblicherweise Buchstaben, Ziffern und Sonderzeichen, die das exklusive gemeinsame Geheimnis einer Person und des Rechners ist. Das heißt: Im Idealfall (der in der Praxis leider viel zu selten auftritt) kennen nur der Rechner und der jeweilige Benutzer das Passwort.

Die zugrunde liegende Idee ist einfach: Der Rechner hat ein „Referenzpasswort" P_0 gespeichert, das dem Benutzer bekannt ist. Wenn der Benutzer sich authentifizieren muss (zum Beispiel beim Rechnerzugang), tippt er sein Passwort P ein, und der Rechner überprüft, ob P und P_0 übereinstimmen. Der Benutzer wird zugelassen, wenn $P = P_0$ ist (siehe Abb. 4.8).

Algorithmus 4.2: Das Passwortverfahren

Erzeugen eines Passworts:	Zwischen Verifizierer und dem Benutzer wird ein geheimes Passwort vereinbart. (Dieses kann dem Benutzer zugewiesen werden, oder er kann sich dies wählen und dem Verifizierer übermitteln.)
Verifizieren eines Passworts:	Der Benutzer gibt seine Identität an und überträgt das Passwort. Der Verfizierer überprüft, ob dies das dem Benutzer zugeordnete (und ihm bekannte) Passwort ist.

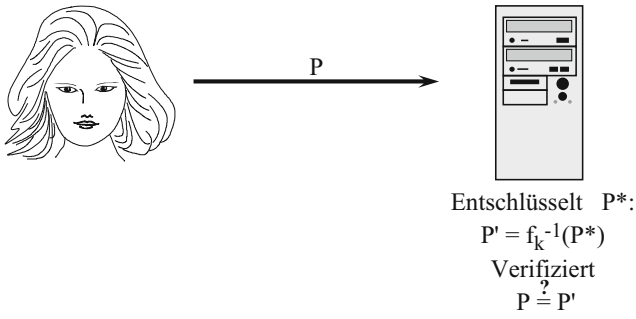


Abb. 4.9 Ein Passwortmechanismus mit Verschlüsselung

Es gibt viele, viele Probleme im Zusammenhang mit Passwörtern, insbesondere solche, die mit dem schlechten Umgang mit Passwörtern zusammenhängen. Wir werden uns hier nur solchen Problemen zuwenden, die mit kryptologischen Mitteln gelöst werden können. Das sind Probleme, die die Speicherung der Passwörter betreffen.

Wenn Mr. X lesenden Zugriff auf die Passwortdatei hat (oder sich diesen verschafft), kann er alle Passwörter lesen und damit in die Rolle eines jeden Benutzers schlüpfen. Die oben beschriebene Prozedur ist also nur dann sicher, wenn die Passwortdatei unauslesbar gespeichert wird – eine Forderung, die jeder Systemprogrammierer als unrealisierbar ablehnen wird. Als Rezept kommt jedem, der das Buch bis hierher gelesen hat, *Verschlüsselung der Passwörter* in den Sinn. Wenn man dies naiv macht, wird man ein Protokoll vorschlagen, wie es in Abb. 4.9 zu sehen ist.

Der Rechner speichert die verschlüsselten Passwörter $P^* = f_k(P_0)$; bei der Authentifikation wird P* entschlüsselt und mit dem vom Benutzer eingetippten Passwort verglichen.

Diese Prozedur hat einen klaren Vorteil. Selbst wenn Mr. X die Passwortdatei lesen kann, kann er nur die verschlüsselten Passwörter lesen. Da er aber nicht in der Lage ist, diese zu entschlüsseln, kann er auch nicht die einzelnen Benutzer durch Eingabe der korrekten Passwörter simulieren.

Ist dieses Argument richtig? Natürlich hat man die Menge der geheimen Daten erheblich reduziert; anstelle einer unbegrenzten Anzahl geheim zu speichernder Passwörter muss man im Verschlüsselungsmodell nur den Schlüssel k geheim speichern. Dies ist zweifellos ein großer Fortschritt, aber keine prinzipielle Lösung des Problems: Wenn sich Mr. X nämlich den Schlüssel k verschafft, kann er alle Passwörter entschlüsseln und hat nach wie vor die Möglichkeit, falsche Identitäten vorzuspiegeln. Fragen Sie einen Computerspezialisten, wie schwierig (?) es ist, Daten auszulesen, die nur in Software gespeichert sind.

Nach einer solchen Analyse wagt sich ein Entwickler eines Passwortsystems wahrscheinlich nicht mehr, folgenden Traum zu äußern: Er träumt von einer Funktion, die Passwörter „verschlüsselt" – aber ohne dazu einen geheimen Schlüssel zu benötigen. Das hört sich utopisch an, aber unserem Manne kann geholfen werden. Dafür benötigt man so genannte *Einwegfunktionen*. Dies sind Funktionen, die Klartexte in Geheimtexte transformieren, aber nicht invertiert werden können. Mit anderen Worten: Eine Einwegfunktion ist eine umkehrbare Funktion mit der (unglaublichen!) Eigenschaft, dass es zu jedem gegebenen Geheimtext nur mit unvorstellbar großem Aufwand möglich ist, sein Urbild zu berechnen.

Sie werden kaum glauben, dass die Welt von Einwegfunktionen nur so wimmelt (siehe Übungsaufgaben 2, 3 und 4). Tatsächlich sind die meisten zeitabhängigen Vorgänge Einwegfunktionen; solche Vorgänge können in einer Richtung äußerst leicht durchgeführt werden, sind aber praktisch nicht umkehrbar: Sie werden zweifellos zum Beispiel schon festgestellt haben, dass die Menschen in Ihrem Bekanntenkreis immer älter werden aber keiner jünger.

Es ist deutlich schwieriger, mathematische Einwegfunktionen zu erfinden. Ein Beispiel ist das folgende: Sei F ein Verschlüsselungsalgorithmus, der einen Klartext m unter dem Schlüssel k auf den Geheimtext $c = F_k(m)$ abbildet. Wir werden diese Funktion nur ein winziges bisschen verändern. Dazu legen wir einen (nicht geheimen) „Klartext" m_0 (etwa $m_0 = 00\ldots0$) fest; dieser geht zwar in den Algorithmus an Stelle des Klartextes ein, spielt aber nicht die Rolle eines variablen Klartexts. Der variable Klartext m wird dem Algorithmus F an Stelle des Schlüssels eingegeben. In kurzen Worten heißt dies: Wir betrachten die Funktion

$$f\colon m \longmapsto c = F_m(m_0).$$

Behauptung: f ist eine Einwegfunktion. Stellen wir uns vor, dass Mr. X sowohl m_0 als auch c kennt und m finden möchte. In der Sprache des Verschlüsselungsalgorithmus F kann man das so ausdrücken: Mr. X kennt das zusammengehörige Klartext-Geheimtextpaar (m_0, c) und möchte den Schlüssel m berechnen. Falls der Algorithmus F kryptologisch sicher ist, ist er resistent gegen diesen known plaintext-Angriff, und daher ist f eine Einwegfunktion.

Frage: Kann man *mathematisch beweisen*, dass die Funktion f eine Einwegfunktion ist? Antwort: Nein! … und die Wahrheit ist noch viel schlimmer! Die Mathematiker konnten nämlich noch von keiner einzigen Funktion stringent nachweisen, dass sie eine Einwegfunktion ist. Das heißt: Man kennt keine Funktion, deren Funktionswerte in polynomieller Zeit berechnet werden können, bei der aber das Invertieren exponentiellem Aufwand benötigt. Im

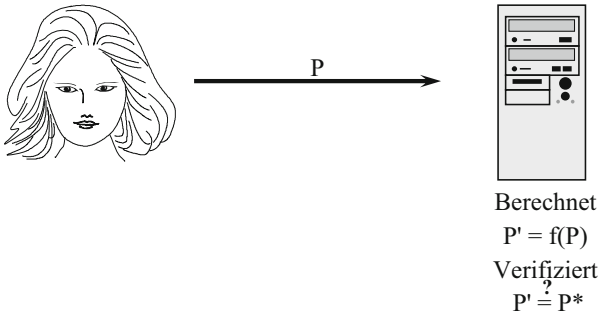


Abb. 4.10 Ein Passwortmechanismus mit einer Einwegfunktion

Klartext: Wir wissen nicht, ob es theoretisch Einwegfunktionen gibt (vergleiche hierzu [BDG95], Kap. 3). Für praktische Zwecke haben aber Funktionen wie die oben beschriebene hinreichend gute Einwegeigenschaften.

Nun setzen wir eine Einwegfunktion für die vorher angesprochene Passwortspeicherung ein. Sei dazu f eine beliebige Einwegfunktion, f muss nicht notwendig wie im obigen Beispiel konstruiert worden sein. Im Speicher des Rechners wird der Wert $P^* = f(P_0)$ gespeichert. Während eines Authentifikationsvorgangs wird f auf die (vielleicht falsche) Zeichenfolge P' angewandt, die der Benutzer eingetippt hat. Anschließend überprüft der Rechner, ob $f(P') = P^*$ ist oder nicht (siehe Abb. 4.10).

Der Vorteil dieser Methode ist offensichtlich: Die Passwortdatei ist unentschlüsselbar, und es gibt kein Geheimnis. Es ist bemerkenswert, dass Einwegfunktionen genau dafür erfunden wurden, um Passwörtern unauslesbar zu speichern; diese Methode wurde in den sechziger Jahren des vorigen Jahrhunderts von Roger Needham von der Universität Cambridge vorgeschlagen und realisiert (vergleiche hierzu [Dif88]).

Bemerkung: Natürlich schützt eine solche Prozedur nicht gegen schlecht gewählte Passwörter. Mr. X kann nämlich einen c*hosen password-Angriff* versuchen: Wenn er eine Liste mit den am häufigsten verwendeten Passwörtern (Mädchennamen, Telefonnummern, Geburtsdaten, …) hat, so kann er f auf diese populären Passwörter anwenden und die Ergebnisse mit den Einträgen in der Passwortdatei vergleichen. Damit kann er die Rechte all derjenigen Benutzer erringen, die eines der populären Passwörter verwenden. Vergleichen Sie dazu Übungsaufgabe 10. Empirische Untersuchungen zeigen, dass Mr. X auf diese Art und Weise einen erheblichen Prozentsatz aller Passwörter knacken kann – aber nicht Ihres; denn Sie haben sich sicherheitsbewusst ein komplexes, schwierig zu erratendes Passwort gewählt!

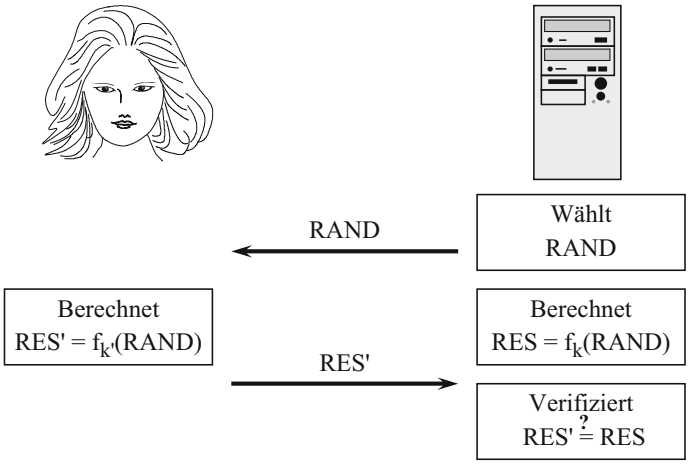


Abb. 4.11 Challenge and response

Authentifikation mit Chipkarten

Kein Passwortsystem genügt extrem hohen Sicherheitsanforderungen. Das liegt ganz einfach daran, dass Menschen nicht für Passwörter geschaffen sind. Damit ein Passwort im Gedächtnis gespeichert werden kann und nicht aufgeschrieben werden muss, muss es kurz und übersichtlich sein. Außerdem sind Passwörter eine statische Authentifikationsmethode. Natürlich kann der Rechner uns Benutzer zwingen, die Passwörter regelmäßig, beispielsweise jeden Monat, zu wechseln. Dies ist aber nur ein kleiner Fortschritt: Das gleiche Passwort wird immer noch mehrere Male verwendet. Warum ist dies ein Problem? Nun, Mr. X hört einmal die Übertragung des Passworts ab, dann kennt er es und kann sich von da an als ein anderer Benutzer maskieren.

Um dieses Problem zu lösen, müssen sich die Daten, die zwischen dem Rechner und dem Benutzer ausgetauscht werden, ständig ändern – jedes Mal eine neue Frage und eine neue Antwort. Es ist klar, dass die Antwort des Benutzers so gestaltet sein muss, dass kein anderer diese Antwort geben kann. Das bedeutet, dass die Reaktion des Benutzers auf die Frage von einem Geheimnis abhängen muss, da sonst auch Mr. X die Antwort geben könnte. Ein entsprechendes Protokoll nach der Methode *challenge and response* läuft so ab, wie in Abb. 4.11 dargestellt ist.

Sowohl der Rechner wie der Benutzer verfügen über eine schlüsselabhängige Einwegfunktion f und einen gemeinsamen geheimen Schlüssel k.

Der Rechner erhält vom Benutzer die Identifikationsdaten; in unserem Fall etwa den Namen der Benutzerin A. Daraufhin verschafft sich der Rechner den zu dem Namen gehörigen Schlüssel k. (Der Schlüssel k kann in einer Liste stehen, von einem systemeinheitlichen „Globalschlüssel“ abgeleitet oder durch eine ähnliche Prozedur bereitgestellt werden.)

Der Rechner muss sich davon überzeugen, dass der sich authentifizieren wollende Benutzer wirklich die Benutzerin A ist – und nicht etwa der hinterlistige Mr. X. Wenn der Benutzer nachweisen kann, dass er den richtigen Schlüssel k hat, so wird er als authentisch anerkannt. Das heißt: Es gilt dann als bewiesen, dass es sich wirklich um die Benutzerin A handelt. Das Ziel des Rechners in der im Folgenden beschriebenen Authentifikationsprozedur ist, sich indirekt davon zu überzeugen, dass der Benutzer im Besitz des Schlüssels k ist. In der Beschreibung des Protokolls bezeichnen wir zur Vorsicht den Schlüssel, der im Besitz des Benutzers (der Mr. X sein könnte!) ist, mit k'. Der Rechner muss sich also überzeugen, ob $k' = k$ ist oder nicht.

Dazu stellt der Rechner dem Benutzer eine Frage, indem er ihm die zufällig ausgewählte Zahl RAND schickt, die dieser kein bisschen besser vorhersagen kann als Mr. X es könnte. Dann berechnet der Benutzer die *Antwort* (*Response*) $RES' = f_{k'}(RAND)$ und sendet diesen an den Rechner. Gleichzeitig bildet der Rechner den Wert $RES = f_k(RAND)$ und überprüft dann, ob $RES' = RES$ ist. Wenn nicht, dann „stimmt etwas nicht"; andernfalls hat der Benutzer mit hoher Wahrscheinlichkeit den gleichen Schlüssel wie der Rechner benutzt. Folglich wird er als authentisch akzeptiert.

Algorithmus 4.3: Challenge and Response

Der Verifizierer schickt dem Benutzer eine Zufallszahl.

Dieser wendet darauf einen Authentifikationsalgorithmus unter einem geheimen Schlüssel an und schickt das Ergebnis RES zurück.

Der Empfänger erhält RES und vergleicht den erhaltenen Wert mit dem von ihm berechneten. Er akzeptiert den Benutzer genau dann, wenn die beiden Werte übereinstimmen.

Diese Prozedur hat zwei Vorteile und zwei Nachteile. Lassen Sie uns mit den guten Nachrichten beginnen: Da die Zahl RAND zufällig gewählt wird, verändert sie sich von Mal zu Mal, und Mr. X hat keinerlei Chance, die jeweils nächste RAND vorherzusagen. Auch RES hat den Charakter einer Zufallszahl, so dass Mr. X nie die richtige Antwort auf eine Frage geben kann. Ebenso wichtig ist die Tatsache, dass durch das Protokoll zwar nachgewiesen wird, dass die beiden Schlüssel gleich sind, die Schlüssel selbst aber innerhalb des Protokolls niemals übermittelt werden.

Die Nachteile des Challenge-and-Response-Protokolls sind zwar „klein", aber dürfen weder theoretisch noch praktisch vernachlässigt werden. Beachten Sie, dass der Rechner die geheimen Schlüssel der Benutzer besitzt. Daher ist es dem Rechner möglich, sich gegenüber anderen Instanzen (oder gegenüber sich selbst) als Benutzer auszugeben. Mit anderen Worten: Eine Voraussetzung für

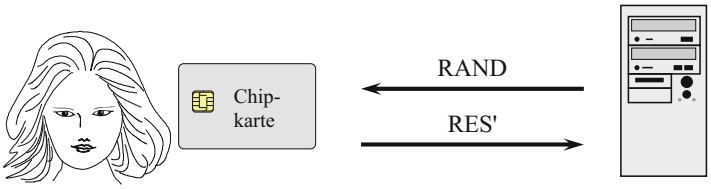


Abb. 4.12 Challenge and response mit einer Chipkarte

die Anwendung von challenge and response ist, dass die Benutzer der Integrität des Authentifikationsrechners vertrauen. Sie glauben wahrscheinlich, dass man ein solches Problem nicht mit kryptographischen Mitteln lösen kann; wir werden aber in Abschn. 4.2.3 und in Kap. 5 höchst elegante Lösungen für dieses Problem kennen lernen.

Der andere Nachteil besteht darin, dass die Senderin A den Algorithmus f ausführen muss. Nun sind Frauen zweifellos zu viel mehr in der Lage, als unsere Schulweisheit sich träumen lässt, aber einen kryptographischen Algorithmus auszuführen, übersteigt doch das Können jedes menschlichen Wesens. Diese Dinge sollte man wirklich einem elektronischen Diener überlassen. Ein solcher Diener kann beispielsweise eine *Chipkarte* sein; diese werden wir in Abschn. 4.3 behandeln. Eine bessere Vorstellung des Protokolls bietet daher Abb. 4.12.

Toll! Hier werden Rechner genau dafür eingesetzt, wofür sie da sind: Um für uns etwas zu berechnen. Gibt's dabei ein Problem? Nun – wie weiß die Chipkarte, dass sie der speziellen Benutzerin A gehört? Genauer gefragt: Wie kann sich die Chipkarte davon überzeugen, dass in einem speziellen Augenblick ihr authentischer Benutzer (und nicht Mr. X, der die Karte gestohlen haben könnte) sie benutzen will? Antwort: Die Chipkarte verlangt vom Benutzer, dass er sich ihr gegenüber authentifiziert. Genaueres werden Sie in Abschn. 4.3 erfahren.

Sicherheit beim Handy

Im Mobilfunk ist Sicherheit unerlässlich. Und wie bei jedem großen System haben die verschiedenen Partner unterschiedliche Sicherheitsinteressen:

- Authentifikation: Der *Netzbetreiber* möchte sicher gehen, dass er seine Rechnung an den richtigen Teilnehmer schickt. Mit anderen Worten: Er muss den anrufenden Teilnehmer zweifelsfrei identifizieren.
- Verschlüsselung: Die *Teilnehmer* möchten sicher sein, dass ihre Gespräche mindestens so geheim bleiben wie im Festnetz. (Wie geheim die Gespräche im Festnetz wirklich sind, ist allerdings eine andere Frage . . .).

Diese Anforderungen wurden beim *GSM*-System (Global System for Mobile Communication) wie folgt gelöst.

Der Netzbetreiber authentifiziert den Teilnehmer durch ein Challenge-and-Response-Protokoll. Ein Problem tritt auf, wenn der Teilnehmer nicht in seinem Heimnetz, sondern in einem Fremdnetz eingebucht ist. Der Betreiber könnte seinen Authentifizierungsschlüssel an den Betreiber des Fremdnetzes ausliefern; dies liegt aber nicht in seinem Interesse, weil er damit die Kontrolle über seinen Kunden abgibt. Daher erzeugt der Betreiber ein oder mehrere Paare der Form (RAND, RES) und schickt diese auf Anforderung an den Betreiber des Fremdnetzes. Dieser kann dann ein oder mehrere Authentifizierungsprotokolle mit dem Teilnehmer durchführen, ist aber nicht in der Lage, eigenständig neue Paare (RAND, RES) zu erzeugen.

Übrigens: Die Authentifizierung ist nur einseitig. Das heißt, dass sich zwar der Netzbetreiber vor betrügerischen Teilnehmern schützen kann, ein Teilnehmer aber einem vorgespielten betrügerischen Netz machtlos gegenüber steht. Dieses Problem wurde bei der Standardisierung von *UMTS* (Universal Mobile Telecommunication Systems) behoben: Dort ist eine zweiseitige Authentifikation vorgesehen.

Die Verschlüsselung der digitalisierten Telefonate erfolgt nur zwischen dem Handy und der ersten Basisstation, auf der so genannten „Luftschnittstelle". Anschließend wird das Gespräch im Festnetz weiter vermittelt und erfährt dort die gleiche (Un-)Sicherheit wie konventionelle Telefonate. Zur Verschlüsselung wird ein prinzipiell geheimer Algorithmus benutzt, der auf den prosaischen Namen *A5* hört.

Aus Kundensicht ist die natürliche Anforderung eine Ende-zu-Ende-Sicherheit, also eine durchgängige Verschlüsselung zwischen den beiden Teilnehmern. Das wird aber auch in UMTS nicht realisiert werden.

Alle Schlüssel, die für die kryptographischen Operationen verwendet werden, sind in der *SIM* (Subscriber Identity Module) gespeichert; das ist die winzige Chipkarte, die Sie bei Vertragsabschluss erhalten und in Ihr Handy eingeschoben haben.

4.2.3 Zero-Knowledge-Protokolle

Ich beginne mit einer persönlichen Erfahrung. Als ich eines Abends von der Arbeit nach Hause kam, fragte ich meine Tochter Maria: „Hast du deine Hausaufgaben gemacht?" Sie antwortete spontan „Natürlich. Aber ich zeige sie dir nicht!" Auf meine offensichtliche Frage erwiderte sie „Keine Angst, es ist alles richtig." „Das ist nicht mein Problem", musste ich zugeben, „ich möchte mich nur davon überzeugen, dass du deine Hausaufgaben gemacht hast." Jetzt wurde Maria verständlicherweise ärgerlich: „Ich werde dir meine Hausaufga-

ben nicht zeigen. Du musst mir eben vertrauen, dass ich sie gemacht habe!“ – Und natürlich musste ich.

Offenbar ist dies ein unauflösliches Dilemma: Maria kann mich unmöglich von ihrem Geheimnis überzeugen, ohne es mir zu zeigen. Oder . . . ?

Das ist die Frage – und die Antwort ist „doch“! Kaum zu glauben, aber wahr: es gibt Prozeduren, die es Maria (oder ihren elektronischen Helfershelfern) erlauben, mich davon zu überzeugen, dass sie ein gewisses Geheimnis hat, ohne mir auch nur das Geringste über das Geheimnis zu verraten. Solche Protokolle heißen völlig zu Recht *Zero-Knowledge-Protokolle*.

Die nächsten beiden Abschnitte haben zum Ziel, Sie zu überreden, an die Existenz von Zero-Knowledge-Protokollen zu glauben. Das erste ist ein historisches Beispiel, das zweite ein Spiel, das dann zu den „richtigen“ Algorithmen überleitet.

Historisches Beispiel: Das Geheimnis des Tartaglia

Das Lösen von Gleichungen war in der gesamten Geschichte der Mathematik ein äußerst wichtiges Problem. Schon früh war bekannt, wie man lineare Gleichungen löst; zum Beispiel hat die Gleichung $3x + 5 = 7$ die Lösung $x = 2/3$. Schon in der Antike wurden auch Verfahren zur Lösung von quadratischen Gleichungen, also Gleichungen der Form $x^2 + px + q = 0$, entwickelt. Bereits in der ersten Hälfte des neunten Jahrhunderts beschrieb der berühmte arabische Mathematiker Muhammad ibn Musa Al-Khwarizmi in seinem Buch *Hisab al-jabr w'al-umqabala* die Lösung, an die Sie sich sicherlich noch, vielleicht mit gemischten Gefühlen, aus dem Mathematikunterricht der Mittelstufe erinnern:

$$x_{1,2} = -\frac{p}{2} \pm \frac{\sqrt{p^2 - 4q}}{2}.$$

Eine kubische Gleichung, also eine Gleichung der Form $ax^3 + bx^2 + cx + d = 0$, zu lösen, ist unglaublich schwierig, und die Entdeckung dauerte entsprechend lange. Wir stellen hier nur ein Kapitel der langen Geschichte dar. Der venezianische Rechenmeister Niccolò Tartaglia (der „Stotterer“), der von ca. 1499 bis 1557 lebte, entdeckte nach seinen eigenen Angaben im Jahre 1535 die Methode zur Lösung der kubischen Gleichung. Er verkündete seine Entdeckung öffentlich, behielt aber die Formel streng geheim. Im Gegensatz zu meinem Problem mit Maria war Tartaglia allerdings in der Lage, seine Zweifler davon zu überzeugen, dass er das Geheimnis zur Lösung kubischer Gleichungen wirklich besaß. Er ließ sich einfach kubische Gleichungen geben; und wenn er nach einer Weile die Lösung präsentierte, konnte sich jeder ohne Schwierigkeiten davon überzeugen, dass die Lösung korrekt war.

Stellen wir uns vor, jemand hätte Tartaglia die Gleichung

$$x^3 + \frac{1}{2}x^2 - \frac{13}{2}x + 3 = 0$$

vorgelegt und Tartaglia hätte nach einiger Zeit geantwortet: Die Lösungen sind 2, −3 und 1/2. Dann hätte sich jeder überzeugen können, dass Tartaglia recht hat, indem er einfach überprüfte, ob

$$(x-2)\,(x+3)\left(x-\frac{1}{2}\right) = x^3 + \frac{1}{2}x^2 - \frac{13}{2}x + 3$$

gilt. Obwohl seine Konkurrenten sich außerordentlich anstrengten, kamen sie nicht hinter Tartaglias Geheimnis. Schließlich überredete der damalige Wissenschaftsstar Geronimo Cardano (1501 bis 1576) den armen Tartaglia, ihm die Formel zu zeigen; er werde sie, so schwur er, absolut geheim halten. Es kam, wie es kommen musste: In seinem Buch *Ars Magna* (1545) veröffentlichte Cardano Tartaglias Formel; Cardano schrieb ehrlicherweise auch, dass diese Formel auf Tartaglia zurückgehe. Dennoch ist es eine Ironie der Geschichte, dass heute die Formel zur Lösung kubischer Gleichungen die *Cardano'sche Formel* heißt.

Natürlich könnte man noch viel darüber erzählen. Ich empfehle jedem, sich das Vergnügen zu gönnen, diese „außergewöhnliche und bizarre" Geschichte in einem Buch über Geschichte der Mathematik nachzulesen (etwa in [Str56], [WA75] oder, in romanhafter Form, in [Jö01].)

Für uns ist vor allem die bemerkenswerte Tatsache wichtig, dass Tartaglia ein Geheimnis hatte (nämlich die Methode zur Lösung kubischer Gleichungen), das er einerseits geheim halten, von dessen Existenz er aber andererseits andere überzeugen konnte.

Das Quadratwurzelspiel

Ich denke mir eine Zahl, deren Quadrat mod 55 gleich 34 ist! Ich möchte dich davon überzeugen, dass ich diese Zahl kenne – und zwar ohne dass ich dir den Spaß an Übungsaufgabe 5 verderbe. Deswegen wähle ich zufällig eine Zahl r, quadriere sie mod 55, erhalte $r^2 \bmod 55 = 26$ und schicke dir diese Zahl.

Jetzt kommst du dran. Du wirfst eine Münze; wenn sie Kopf zeigt, willst du von mir $r \cdot s \bmod 55$, andernfalls einfach r.

Angenommen, die Münze zeigt Kopf. Dann sage ich $r \cdot s \bmod 55 = 53$, und du kannst nun einfach überprüfen, dass tatsächlich

$$53^2 \bmod 55 = 4 = (26 \cdot 34) \bmod 55 = (r^2 \cdot s^2) \bmod 55$$

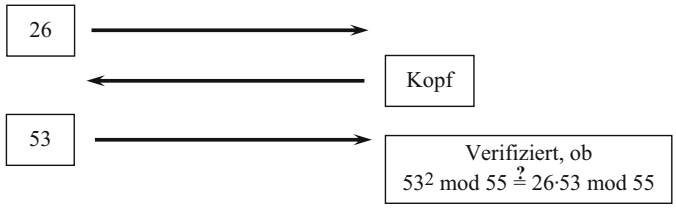


Abb. 4.13 Das Quadratwurzelspiel

gilt (siehe Abb. 4.13). Beachte, dass es genauso schwierig ist, s zu finden, wenn du $r^2 \bmod 55$ und $(r \cdot s) \bmod 55$ kennst, wie ohne diese Kenntnis. Andererseits darf r nicht völlig zufällig gewählt werden; zum Beispiel sollte r größer als $\sqrt{55}$ sein. Denn wenn du r aus r^2 direkt finden könntest, könntest du auch ganz einfach $s = (53/r) \bmod 55$ berechnen. (Dabei bedeutet $53/r \bmod 55$ die Multiplikation von 53 mit der modularen Inversen von r mod 55; siehe Abschn. 5.3.2.2.)

Sollen wir noch eine Runde spielen? Ich sage $r^2 \bmod 55 = 14$. Du wirfst die Münze, erhältst „Zahl", und ich muss zugeben, dass r = 17 ist. Jetzt kannst du leicht überprüfen, dass ich nicht gemogelt habe, denn es ist $17^2 = 289$ und $289 \bmod 55 = 14$.

Würde ich dich betrügen? Ehrlich gesagt ja – wenn du es nicht entdecken würdest. Aber bei diesem Quadratwurzelspiel wird jeder Betrug *irgendwann einmal* offenbar! Stellen wir uns vor, ich hätte gemogelt und würde s gar nicht kennen. Wenn ich wüsste, dass deine Münze „Zahl" zeigt, dann hätte ich keine Probleme, dir r zu geben (das ich natürlich kenne, weil ich es ja gewählt habe). Allerdings wüsste ich nicht, was ich tun sollte, wenn du $(r \cdot s) \bmod 55$ wissen willst!

Wenn ich andererseits wüsste, dass deine Münze „Kopf" zeigt, käme ich auch zurecht. Ich würde eben mein r^2 entsprechend präparieren: Zunächst wähle ich eine Zahl, von der ich weiß, dass sie modulo 55 ein Quadrat ist, etwa $2^2 = 4$, und würde dann

$$r^2 := (4 / s^2) \bmod 55 = (4 / 34) \bmod 55 = 26$$

wählen. Wenn du jetzt $(r \cdot s) \bmod 55$ wissen willst, sage ich einfach 2, und – wer hätte es gedacht! – du kannst dich überzeugen, dass gilt:

$$2^2 = 26 \cdot 34 \bmod 55 = (4 / 34) \cdot 34 \bmod 55.$$

Wenn ich mich aber täusche und deine Münze „Zahl" zeigt und ich dir also r geben müsste, würde mein Betrug sofort auffliegen.

Ich kann also immer betrügen, wenn ich das Ergebnis des Münzwurfs vorhersagen kann. Das heißt, dass ich in jeder Runde eine Chance von 50 % habe, erfolgreich zu betrügen. Um dich also wirklich zu überzeugen, dass ich das richtige s habe, müssen wir das Spiel über viele Runden spielen. Wenn ich s kenne, gebe ich dir jedes Mal die richtige Antwort. Wenn ich aber s nicht kenne, muss ich deine Frage jedes Mal vorhersehen. In jeder Runde kann ich das nur mit Wahrscheinlichkeit 1/2 machen; in zwei Runden mit Wahrscheinlichkeit $1/2 \cdot 1/2 = 1/4$; die Wahrscheinlichkeit, die Münzwürfe für t Runden richtig vorherzusagen, ist $1/2^t$.

Dies ist natürlich nur ein Spiel. Es ist nämlich nicht allzu schwer, Quadratwurzeln modulo 55 zu finden; dies ist sogar so leicht, dass Sie das als Übungsaufgabe lösen können (Übungsaufgabe 5). Dieses Spiel zeigt jedoch die entscheidenden Eigenschaften der echten Zero-Knowledge-Protokolle, die wir anschließend behandeln werden:

- Das Protokoll ist *interaktiv*; beide Partner machen Zufallswahlen. (Ich wähle die Zufallszahl r, du entscheidest dich, ob du $r \cdot s$ oder r haben möchtest.)
- Die Wahrscheinlichkeit für einen erfolgreichen Betrug meinerseits hängt von der Anzahl der gespielten Runden ab. Bei jeder Runde wird diese Wahrscheinlichkeit halbiert.

Nun kommen wir zu den praktisch einsetzbaren Protokollen.

Das Fiat-Shamir-Protokoll

Im Jahre 1986 stellten die israelischen Mathematiker Adi Shamir (den schon in Kap. 1 erwähnt wurde und der uns in Kap. 5 nochmals begegnen wird) und Amos Fiat ein Protokoll vor, das eine neue Dimension bei der Benutzerauthentifikation eröffnet hat. Genau betrachtet handelt es sich um eine Rechner-Rechner Authentifikation, wobei einer der Rechner die Chipkarte des Benutzers ist. Das Fiat-Shamir-Protokoll [FS87] gründet sich auf die Untersuchungen [GMR89] von Shafi Goldwasser und Silvio Micali vom Massachusetts Institute of Technology sowie Charles Rackoff von der University of Toronto. Die Version, die wir beschreiben werden, ist eine Verallgemeinerung des Quadratwurzelspiels.

Die Sicherheit des Protokolls beruht darauf, dass es außerordentlich schwierig ist, die modulare Quadratwurzel einer Zahl v zu finden. Genauer gesagt: Sei eine natürliche Zahl n gegeben, die keine Primzahl ist. *Wenn man die Primfaktorzerlegung von* n *nicht kennt, ist es praktisch unmöglich, eine Zahl* s *zu finden mit* $s^2 \bmod n = v$. Es wird empfohlen, n so groß zu wählen, dass n etwa 200 Dezimalstellen hat; das bedeutet, dass n in normaler Schriftgröße eine Zahl ist, die etwa 1 m lang ist. Wir werden die Schwierigkeit der Faktori-

sierung so großer Zahlen in Kap. 5 ausführlich diskutieren. Hier akzeptieren wir einfach die Tatsache, dass die Kenntnis einer modularen Quadratwurzel s von v ein wertvolles Geheimnis ist.

Lange bevor ein Authentifikationsprozess mit Hilfe des *Fiat-Shamir-Protokolls* durchgeführt werden kann, wurden in einer Schlüsselzentrale zwei Primzahlen p und q gewählt und ihr Produkt $n = p \cdot q$ gebildet. Es ist entscheidend, dass die Zentrale p und q geheim hält, während n öffentlich ist. Daher muss n so groß sein, dass Mr. X die Faktorisierung von n für ein hoffnungsloses Unterfangen hält. Der Grund dafür ist, dass man relativ einfach modulare Quadratwurzeln mod p und mod q berechnen und aus diesen Zahlen dann auch Quadratwurzeln mod n „zusammensetzen" kann. Dies darf aber nur für die Zentrale und nicht für Mr. X möglich sein. Überzeugen Sie sich, dass die Zentrale diese Aufgaben wirklich bearbeiten kann, indem Sie die Übungsaufgaben 15, 16 und 17 lösen.

Die Schlüsselzentrale berechnet für jeden Benutzer eine Zahl s, die das Geheimnis des Benutzers ist, und eine Zahl v derart, dass $v = s^2 \bmod n$ gilt. Die Zahl v dient zur öffentlichen Verifizierung des Benutzers. Die Zentrale könnte etwa für v die Identifizierungsdaten (in binärer Form) des Benutzers wählen und daraus s entsprechend berechnen. Die Zahl n ist eine öffentliche Systemkonstante.

Damit sind bereits alle Aufgaben der Zentrale beschrieben. Insbesondere spielt sie bei den aktuellen Authentifikationsprozessen keinerlei Rolle.

In Abb. 4.14 wird das Protokoll dargestellt, mit dem die Benutzerin A einen Authentifikationsrechner davon überzeugt, dass sie das Geheimnis s besitzt, ohne dem Rechner dabei das Geringste von s preiszugeben.

Einige Beobachtungen sind sofort einsichtig.

- Beide Rechner müssen nur sehr wenige Berechnungen mod n ausführen: Auf der Seite des Benutzerin muss nur die Zufallszahl r quadriert werden; in der Hälfte aller Fälle muss noch $r \cdot s$ berechnet werden. Der Authentifikationsrechner muss y quadrieren und in der Hälfte der Fälle x mit v multiplizieren.
- Der Authentifikationsrechner verwendet nur öffentliche Information, während der Benutzer das Geheimnis s ganz wesentlich einsetzt.
- Der Verifizierer wird irgendwann davon überzeugt sein, dass die Benutzerin A wirklich das Geheimnis s hat. Die Wahrscheinlichkeit, dass sie (auch mit weiblicher Intuition) jedes Mal das Bit b richtig vorhersagt, ist bei t-maliger Ausführung des Protokolls nur $1/2^t$. Im Falle $t = 20$ ist diese Wahrscheinlichkeit kleiner als 1 zu einer Million.
- Auch nach dem Authentifikationsprozess bleibt das Geheimnis s absolut geheim.

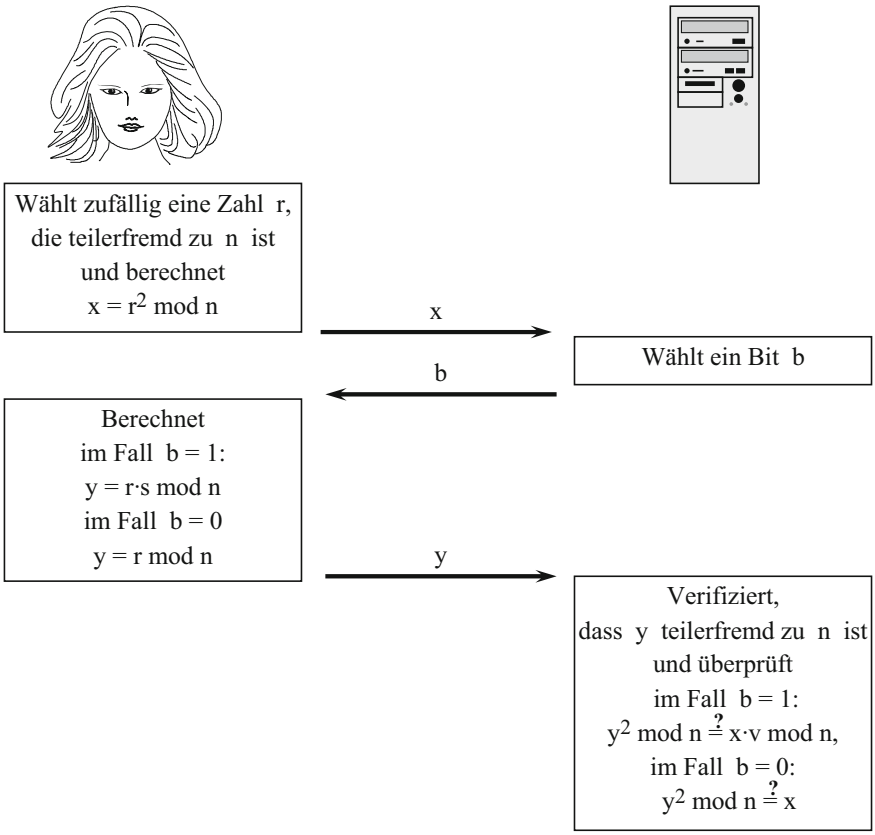


Abb. 4.14 Das Fiat-Shamir-Protokoll

- Die Sicherheit des Protokolls hängt in entscheidender Weise davon ab, dass die Berechnung modularer Quadratwurzeln so schwierig ist, dass selbst der grässliche Mr. X nicht die Quadratwurzel von v berechnen kann – und zwar selbst dann nicht, wenn er Tausende von Transaktionen abgehört hat.

Zero-Knowledge-Protokolle bilden ein blühendes Teilgebiet der Kryptologie. Details des Fiat-Shamir-Protokolls sind in [FS87] zu finden. Es wurde auch ein allgemeines Schema für Zero-Knowledge-Protokolle vorgeschlagen; der interessierte Leser sei auf [BDPW90] hingewiesen.

4.3 Chipkarten

Was ist eine Chipkarte? Antwort: Eine *Chipkarte* ist eine Plastik*karte* in der Größe einer üblichen Scheck- oder Kreditkarte, in die ein *Chip* eingebettet ist,

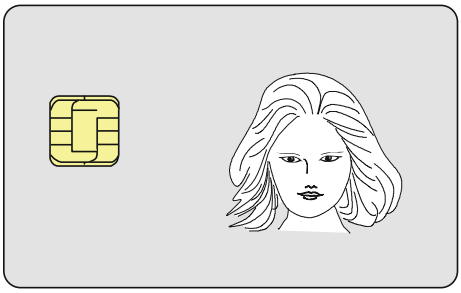


Abb. 4.15 Eine Chipkarte

der Daten speichern kann und in der Lage ist, Berechnungen durchzuführen. Mit anderen Worten: Eine Chipkarte ist ein wirklicher Minirechner! Heutige Chipkarten haben keine Batterie, so dass der Strom während des Betriebs von außen zugeführt werden muss. Das ist eine der Funktionen der ins Auge stechenden Goldkontakte, an denen man eine intelligente Chipkarte von ihren zurückgebliebenen Verwandten unterscheiden kann (siehe Abb. 4.15). Die andere wichtige Funktion der Kontakte ist, den Datentransfer von und zu der Karte zu bewerkstelligen.

Die Geburtsstunde der Chipkarte schlug im Jahre 1974, als der französische Wirtschaftsjournalist Roland Moreno ein *System zur Speicherung von Daten in einem unabhängigen tragbaren Gegenstand* zum Patent anmeldete. Vor Moreno hatte allerdings schon Jürgen Dethloff aus Hamburg zusammen mit Helmut Gröttrup grundlegende Arbeiten durchgeführt und entsprechende Patente erworben. Zunächst war Frankreich das Chipkartenland, wo die CP 8 Karte der Firma Bull Maßstäbe setzte. Aber seit den 80-er Jahren konnte die Erfolgsgeschichte der Chipkarten auch auf andere Länder übertragen werden: Die Telefonkarten, die Versichertenkarten, die Geldkarte, die SIM-Karte sind Chipkarten, um nur einige Beispiele zu nennen. Die Stückzahlen gehen inzwischen in die Zig Milliarden, wobei anzumerken ist, dass Chipkarten bislang weitgehend eine europäische (und keine amerikanische) Entwicklung darstellen.

Warum Chipkarten? Anders gefragt: Warum wird in dieser Einführung in die Kryptologie das neue technische Medium Chipkarte behandelt? Die Antwort ist einfach. Mit der Chipkarte steht zum ersten Mal in der Geschichte ein Medium zur Verfügung, das Sicherheit auf Basis von Kryptologie realisiert und zwar für jedermann und nicht nur für Experten. Das liegt daran, dass in der Chipkarte zwei Eigenschaften zusammenkommen, die bislang vollständig getrennt waren:

- Chipkarten sind *ideal für Kryptologie*: Sie können Kryptoalgorithmen ausführen und geheime Schlüssel auf sichere Weise speichern.

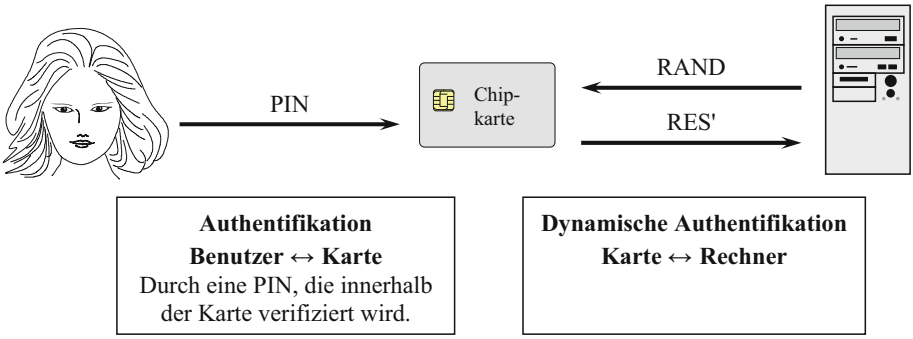


Abb. 4.16 Zwei Authentifikationsschritte

- Chipkarten sind ideal für Menschen: Sie sind extrem einfach zu benutzen. Wir werden sehen, dass die einzige Aufgabe für den Benutzer ist, sich seine Geheimzahl zu merken, um sich gegenüber seiner Karte zu authentifizieren.

Zusammenfassend halten wir fest: Chipkarten sind nicht schwieriger zu bedienen als die üblichen Kreditkarten; sie bieten aber Sicherheitsdienste auf höchstem Niveau, nämlich solche, die auf kryptologischen Mechanismen beruhen. Wir werden zwei typische Chipkartenanwendungen diskutieren, nämlich Zugangskontrolle und elektronisches Bezahlen. Eine Übersicht über Chipkarten und ihre Anwendungen findet man in [RE99]; eine Darstellung von Chipkarten als Sicherheitswerkzeug ist [BKP91].

4.3.1 Chipkarten zur Zugangskontrolle

Die Grundidee ist, das Sicherheitswerkzeug Chipkarte zwischen Benutzer und Rechner einzuführen, um die Authentifikation des Benutzers gegenüber einem Rechner sicherer zu machen. Der übliche, altmodische Passwortmechanismus wird dadurch in zwei unabhängige Schritte zerlegt (vergleiche Abb. 4.16).

- Der Benutzer authentifiziert sich gegenüber seiner Karte durch eine Geheimzahl, die üblicherweise *PIN* (Persönliche Identifizierungs-Nummer) genannt wird.
- Die Karte authentifiziert sich gegenüber dem Rechner durch ein dynamisches Authentifikationsprotokoll.

Wenn Mr. X sich unberechtigt Zugang zum System verschaffen will, so muss er also nicht nur dem rechtmäßigen Benutzer die Geheimzahl abluchsen, sondern er muss sich auch in den Besitz der die Karte bringen.

Sie kennen sicherlich das entsprechende Verfahren von den Geldausgabeautomaten. Jeder, der diesen Geldesel anzapfen will, muss zunächst durch Einschieben der Scheckkarte mitteilen, wer er ist und dann noch beweisen, dass er die zugehörige Geheimzahl kennt. Sie sehen: Diese Geheimzahl ist nicht eine Schikane der Bank, die ihren Kunden das Leben schwer macht, sondern die (einzige!) Sicherheit, die ein Kunde dem System gegenüber hat. Man sollte damit also äußerst sorgfältig und achtsam umgehen! Da unsere Geldausgabeautomaten aber nicht mit Chipkarten, sondern nur mit Magnetstreifenkarten arbeiten, kann die Authentifikation der Karte gegenüber dem Rechner (das heißt, dem Automaten) nur statisch, nicht dynamisch erfolgen; Chipkarten würden hier ein deutlich höheres Sicherheitsniveau bieten.

Wir betrachten nun beide Authentifikationsschritte im Einzelnen.

- Wenn der Benutzer seine PIN über eine Tastatur am Terminal eintippt, wird die Geheimzahl direkt zur Karte übertragen und innerhalb der Karte mit einem gespeicherten Referenzwert verglichen. Wenn diese beiden Werte nicht übereinstimmen, verweigert die Karte jeden weiteren Dienst. (Natürlich wird man den üblichen Mechanismus haben, dass die Karte ihren Dienst nur dann verweigert, wenn ein Benutzer (Mr. X?) dreimal hintereinander eine falsche PIN eingegeben hat.) Man beachte: Da die Geheimzahl des Benutzers nur innerhalb seiner Karte überprüft wird, braucht man keine zentrale PIN-Datei. Man kann sich gut Chipkartensysteme vorstellen, bei denen der Benutzer seine PIN ändern kann, wenn er möchte, und bei denen die PIN variable Länge hat.
- Die Authentifikation der Karte gegenüber dem System erfolgt durch ein Challenge-and-Response-Protokoll, wie sie in Abschn. 4.2.2 beschrieben ist. Dafür brauchen die Karte und der Rechner einen gemeinsamen Algorithmus f und einen gemeinsamen geheimen Schlüssel k. Der Rechner fordert die Karte heraus, indem er ihr eine Zufallszahl RAND schickt. Die Karte wendet den Algorithmus f unter dem Schlüssel k auf RAND an und schickt das Ergebnis RES als Antwort zurück zum Rechner. Dieser hat in der Zwischenzeit mit seinem Schlüssel ebenfalls RES berechnet und kann also überprüfen, ob die beiden Resultate übereinstimmen. Zugangskontrolle mit symmetrischen Algorithmen ist ausführlicher in [BR89] dargestellt.

Die Vorteile einer solchen Prozedur liegen auf der Hand: Die Zahlen verändern sich von Mal zu Mal; daher kann Mr. X nicht vorhersagen, welche Zahl als nächste RAND oder als nächste RES kommen wird. Insbesondere bleiben geheime Daten, wie etwa Schlüssel, geheim.

Die heutigen Chipkarten haben nur kleine Prozessoren und sehr geringe Speicherfähigkeit (wenige Kilobyte). Dies liegt daran, dass die Chipfläche

nicht größer als 25 mm^2 sein darf, denn sonst würde der Chip beim Durchbiegen der Karte brechen. Daher verwendet man heute vorwiegend symmetrische Algorithmen auf Chipkarten, also Algorithmen, bei denen beide Seiten denselben geheimen Schlüssel benutzen. Es ist selbstverständlich äußerst wünschenswert, dass Chipkarten auch Zero-Knowledge-Protokolle oder die im nächsten Kapitel diskutierten Public-Key-Algorithmen unterstützen können. In der Tat wurde das Fiat-Shamir-Protokoll für den Einsatz in Chipkarten entwickelt! Heutige Chipkarten können mit Hilfe eines Krypto-Coprozessors Public-Key-Algorithmen mit großer Schlüssellänge und guter Geschwindigkeit ausführen.

4.3.2 Einkaufen mit der Karte

Die Idee des *elektronischen Einkaufens*, auf gut deutsch auch *electronic cash* oder *POS-Banking* genannt (POS: Point Of Sale) ist die folgende: Die Kundin Anna Trocken geht ins Kaufhaus, sucht sich ihre Waren zusammen, bezahlt dann und zwar weder bar noch mit Scheck, noch mit Kreditkarte – sondern mit ihrer Chipkarte.

Auf den ersten Blick sieht es so aus, als müsste man nur das alte Problem der Authentifikation der Kundin A. Trocken gegenüber dem Kaufmann K. Rotte lösen. Aber die Situation ist komplexer, da nun drei grundsätzlich verschiedene Gruppierungen mit verschiedenen Interessen beteiligt sind: Der Kunde, der Kaufmann und die Bank. Sogar dieses Szenario ist noch fast unzulässig vereinfacht; wir nehmen zum Beispiel an, dass Kunde und Kaufmann dieselbe Bank haben. (Siehe Abb. 4.17.)

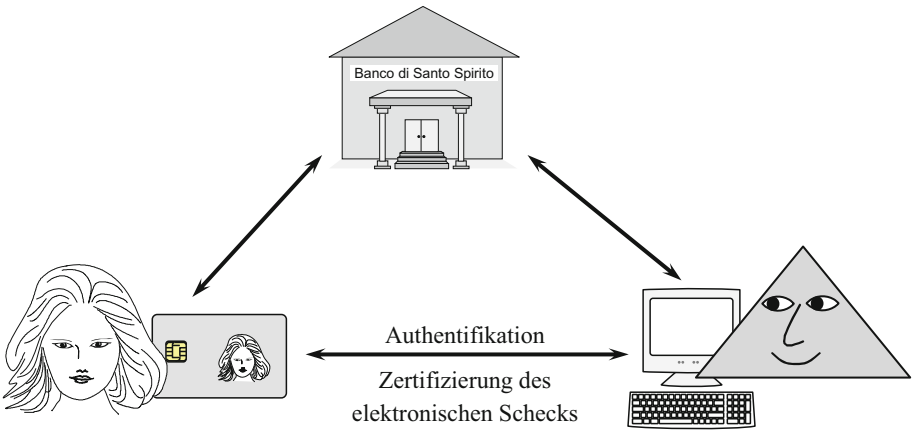


Abb. 4.17 POS-Banking

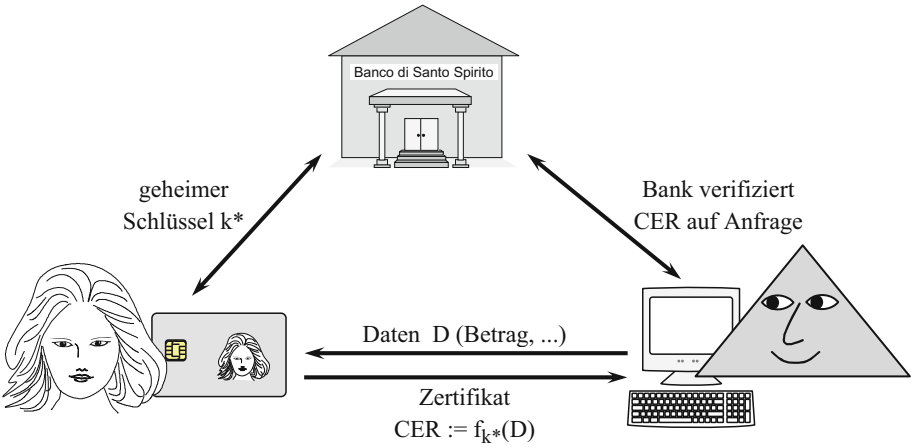


Abb. 4.18 Zertifizierung

Man muss zwischen zwei Diensten unterscheiden, nämlich der Authentifikation der Kommunikationspartner und der Authentifikation der Nachricht, die in der jetzt betrachteten Anwendung eine Art elektronischer Scheck ist.

Die Authentifikation des Kunden gegenüber dem Terminal des Kaufmanns wird in der Tat mit Hilfe eines üblichen Protokolls zur Benutzerauthentifikation durchgeführt, wie es etwa im vorigen Abschnitt beschrieben wurde. In der Regel wird man bei einer solchen Anwendung eine gegenseitige Authentifikation verlangen: Auch das Händlerterminal muss sich gegenüber der Chipkarte als authentisch ausweisen. Denn die Chipkarte hat auf diese Weise die Möglichkeit, manipulierte Terminals von echten zu unterscheiden. Ein solches Protokoll läuft auch nach der Methode „challenge and response" ab – mit dem einzigen Unterschied, dass jetzt die Karte eine Zufallszahl schickt, damit den Rechner herausfordert und dieser die richtige Antwort geben muss.

Eine andere Sache ist die Authentifikation des elektronischen Schecks. Dieses Dokument wird mittels einer Art MAC „unterschrieben" und dann von der Karte zum Händlerterminal geschickt. Es ist klar, dass der Herr Rotte den Scheck nicht ändern können darf; andererseits ist es wünschenswert, dass er die Gültigkeit des Schecks überprüfen kann. Aber der Pferdefuß bei der Verwendung symmetrischer Algorithmen ist, dass jeder, der einen MAC prüfen kann, auch in der Lage ist, diesen fälschen zu können: Er ändert einfach das Dokument und berechnet mit Hilfe des geheimen Schlüssels den zugehörigen neuen MAC. Hier bieten sich asymmetrische Signaturschemata (siehe Kap. 5) als optimale Lösung an.

Wenn man aus praktischen Gründen mit symmetrischen Algorithmen auskommen muss, so kann man vorgehen wie in Abb. 4.18 dargestellt.

Die Kundin, Frau Trocken, und ihre Bank haben einen gemeinsamen geheimen Schlüssel k*, den K. Rotte nicht kennt. Auf den elektronischen Scheck D wird dann wie in Abschn. 4.2.1 beschrieben eine MAC-Prozedur angewandt. Nachdem Kunde und Karte akzeptiert sind, können sie den Kauf tätigen. Dazu müssen sie die Buchungsdaten D (Betrag, Art der Ware, Datum, Name des Kaufhauses, usw.) bestätigen und dem Händler so übermitteln, dass er

- eine Garantie hat, dass er das Geld von der Bank bekommt, und
- an den Daten nichts mehr „drehen" kann. (Ein betrügerischer Kaufmann könnte, nachdem er die Ware geliefert hat, auf die Idee kommen, den Betrag zu erhöhen und den erhöhten Betrag von der Kundenbank einzufordern.)

Um diesen Betrug zu verhindern, besitzt die Karte den Geheimschlüssel k*. Dieser Schlüssel ist dem Kaufmann K. Rotte und wie jedem anderen potentiellen Betrüger völlig unbekannt!

Dieser Schlüssel k* tritt jetzt in Aktion. Die Karte sendet nicht nur die Buchungsdaten D an das Händlerterminal zurück, sondern auch die mit Hilfe des Schlüssels k* „unterschriebenen" Daten. Der so konstruierte MAC wird auch *Zertifikat* genannt und als CER bezeichnet. Dieser CER garantiert, dass der Händler den bei ihm eingereichten Scheck nicht erfolgreich fälschen kann. Wenn er sich aber überzeugen will, dass der Scheck echt ist, kann er die Bank beauftragen, für ihn die Echtheit zu überprüfen.

Die Methode der Zertifizierung ist natürlich nicht ideal, und zwar aus zwei Gründen. Zunächst müssen beide Seiten, Kunde und Kaufmann, der Bank vertrauen. Ferner kann der Kaufmann den Scheck nicht sofort (*offline*) prüfen, sondern er muss gegebenenfalls eine Online-Anfrage an die Bank richten. Nach allgemeiner Überzeugung bietet aber bereits ein solches System eine weit höhere Sicherheit für alle Beteiligten als andere Zahlungssysteme auf Kartenbasis.

4.4 Übungsaufgaben

1 Mit welchem der Begriffe „Vertraulichkeit", „Integrität" oder „Authentifikation" assoziieren Sie die folgenden Vorgänge und Objekte?
Fingerabdruck,
Knopf im Ohr,
Siegel auf Dokumenten,
durch Sahne kunstvoll verzierter Schokoladenpudding,
Keuschheitsgürtel,

TÜV-Plakette,
Aktenkoffer mit Zahlenschloss,
Verplomben eines Zimmers.

2 Interpretieren Sie die folgenden Vorgänge als Einwegfunktionen:
Sich ein Bein brechen,
Zahnpasta (noch besser: Klebstoff) aus der Tube drücken,
Kinderzimmer in Unordnung bringen („spielen"),
Rührei zubereiten,
eine Gehirnwäsche,
Farben mischen,
zum ersten Mal „$e = m \cdot c^2$" denken,
sterben,
ein Kind gebären.

3 Die meisten der in Aufgabe 2 beschriebenen Vorgänge kann man interpretieren als „Übergang von Ordnung in Chaos". Alle? Versuchen Sie sich eine Einwegfunktion auszudenken, bei der das Chaos nicht vergrößert wird.

4 Ist die folgende Funktion eine Einwegfunktion? Es ist sehr einfach, in einem Telefonbuch zu einem Namen die zugehörige Telefonnummer zu finden. Aber es ist umgekehrt außerordentlich schwierig, zu einer gegebenen Nummer aus dem Telefonbuch den entsprechenden Namen herauszusuchen.
[Es gibt eine simple Methode, den Namen herauszubekommen: Rufen Sie die Nummer einfach an!]

5 Suchen Sie eine modulare Quadratwurzel von 34 mod 55 (das ist eine natürliche Zahl s mit $s^2 \bmod 55 = 34$).

6 Hat einer der Partner einen Nachteil, wenn in einem Challenge and Response-Protokoll die „Zufallszahl" RAND nicht zufällig gewählt wird? Unterscheiden Sie bei Ihren Überlegungen die Fälle

- RAND ist konstant,
- RAND ist nicht konstant, aber vorhersehbar (Beispiel: $RAND_{neu} = RAND_{alt} + 1$).
  Überlegen Sie insbesondere, ob irgendjemand außer dem Authentifikationsrechner einen Nachteil hat.

7 Führen Sie sich die folgende besonders tragische Ausprägung einer Einwegfunktion aus Otfried Preußlers *Neues vom Räuber Hotzenplotz* vor Augen. Dort erzählt Frau Schlotterbeck:
„*Ich weiß selbst nicht, weshalb ich ihn [den Dackel Wasti] eines Tages in einen Bernhardiner umhexen wollte. Aus Langeweile vermutlich, nur so zum Zeitvertreib … Was ich an jenem Unglückstag falsch gemacht habe, ist mir bis*

heute schleierhaft. Jedenfalls sieht mein armer Wasti seither wie ein Krokodil aus – auch wenn er im Grunde genommen der brave Dackel geblieben ist, der er immer war."

. . .

„*Und – haben Sie nie versucht, ihn zurückzuhexen?*"

„*Natürlich*", *sagte Frau Schlotterbeck.* „*Aber es hat nicht geklappt, und da habe ich's schließlich aufgegeben. Sie werden begreifen, dass mir seitdem die Lust am Hexen vergangen ist . . .* "

8 Im Jahre 1656 verkündigte der Niederländer Christiaan Huygens (1629 bis 1695) seine Entdeckung des Saturnrings auf folgende originelle Weise:
Aaaaaaaccccdeeeeeghiiiiiiillllmmnnnnnnnnnnoooppqrrsttttttuuuuu
So sicherte sich Huygens die Priorität dieser Entdeckung, ohne sie preisgeben zu müssen. Denn dieses Anagramm lautet in unverschlüsselter Form:
Annulo cingitur tenui, plano, nusquam cohaerente, ad eclipticam inclinato – er ist von einem dünnen, flachen Ring umgeben, der ihn nirgends berührt und gegen die Ekliptik geneigt ist.
Beschreiben Sie dieses Verfahren mit den Begriffen „Einwegfunktion" und „Authentizität".

9 Interpretieren Sie die folgenden Literaturstellen im Lichte von Geheimhaltung bzw. Authentifikation:

(a) „*Wann ihr euch einmal trennt, so stoßt dies Messer am Scheideweg in einen Baum; daran kann einer, wenn er zurückkommt, sehen, wie es seinem abwesenden Bruder ergangen ist; denn die Seite, nach welcher dieser ausgezogen ist, rostet, wann er stirbt; solange er aber lebt, bleibt sie blank.*"
(Brüder Grimm, Die zwei Brüder)

(b) *Im Departement du Gard – ganz richtig, da, wo Nîmes liegt und der Pont du Gard: im südlichen Frankreich – da saß in einem Postbüro ein älteres Fräulein als Beamtin, die hatte eine böse Angewohnheit: sie machte ein bisschen die Briefe auf und las sie. Das wusste alle Welt. Aber wie das so in Frankreich geht: Concierge, Telefon und Post, das sind geheiligte Institutionen, und daran kann man schon rühren, aber daran darf man nicht rühren, und so tut es denn auch keiner.*
Das Fräulein also las die Briefe und bereitete mit ihren Indiskretionen den Leuten manchen Kummer.
Im Departement wohnte auf einem schönen Schlosse ein kluger Graf. Grafen sind manchmal klug, in Frankreich. Und dieser Graf tat eines Tages folgendes:
Er bestellte einen Gerichtsvollzieher auf das Schloss und schrieb in seiner Gegenwart an einen Freund:

Lieber Freund!
Da ich weiß, dass das Postfräulein Emilie Dupont dauernd unsere Briefe öffnet und sie liest, weil sie vor lauter Neugier platzt, so sende ich Dir anliegend, um ihr einmal das Handwerk zu legen, einen lebendigen Floh.
Mit vielen schönen Grüßen
Graf Koks.
Und diesen Brief verschloss er in Gegenwart des Gerichtsvollziehers. Er legte aber keinen Floh hinein.
Als der Brief ankam, war einer drin.
(Kurt Tucholsky, Der Floh)

10 Geben Sie mindestens zwanzig Wörter an, von denen Sie keines als Ihr Passwort benutzen sollten.

11 Betrachten Sie folgende einfache MAC-Bildung. Für eine (deutsche) Nachricht ist der MAC nichts anderes als der entsprechende Geheimtext unter einer Verschiebechiffre.
   (a) Zeigen Sie: Unser böser Mr. X kann erreichen, dass eine beliebige Nachricht mit einer Wahrscheinlichkeit 1/26 akzeptiert wird.
   (b) Was passiert, falls Mr. X eine authentifizierte Nachricht aus mindestens zwei Buchstaben abhört?
   (c) Ist die Betrugswahrscheinlichkeit im Allgemeinen größer als 1/26?

12 Spielen Sie die Rolle von Mr. X und suchen Sie den Schlüssel aus dem Beispiel von Abb. 4.6.

13 Die tragische Entwicklung in William Shakespeares König Lear hängt entscheidend von vier Briefen ab. Überzeugen Sie sich, dass das Drama einen glücklichen Ausgang hätte nehmen können, wenn die Hauptfiguren dieses Buch gelesen hätten. Schauen Sie sich insbesondere die folgenden Szenen an.
   Erster Aufzug, zweiter Auftritt: Edmund, der illegitime Sohn von Gloster, macht Gloster glauben, ein von ihm geschriebener Brief käme von Edgar. In diesem Brief berichtet (scheinbar!) Edgar von seinen hinterlistigen Plänen. Wenn Gloster nachprüfen hätte können, ob der Brief authentisch ist, hätte er ihn als Fälschung erkannt.
   Zweiter Aufzug, vierter Auftritt: Einer von Lears Männern beobachtet, wie ein Brief Gonerils ihrer Schwester Regan durch einen Boten überbracht wird. Dieser Bericht führte (zusammen mit anderen Ereignissen) zu Lears Wahnsinn. Goneril hätte Methoden der Anonymität (Kap. 6) anwenden sollen.
   Dritter Aufzug, dritter Auftritt: Ein Brief an Gloster informiert diesen, dass das Heer, das zur Unterstützung des abgesetzten Königs zusammengestellt wurde, in die Hand des Feindes gefallen ist. Dies zeigt, dass jedes Sicher-

heitssystem zum Scheitern verurteilt ist, wenn ein leichtgläubiger Ehrenmann (Gloster) sein Vertrauen auf jemand setzt, der, wie Edmund, hinterlistig und entschlossen ist.
Vierter Aufzug, sechste Szene: Edgar bringt sich in Besitz eines Briefes von Goneril an Edmund, in dem sie vorschlägt, ihren Gatten aus dem Weg zu räumen. Sie hätte dieses Buch lesen und eine Chiffre benutzen sollen.

14 Interpretieren Sie die folgende Prozedur als Zero-Knowledge-Protokoll. Diese geniale Darstellung wurde von den Familien von Jean-Jaques Quisquater (Belgien) und Louis Guillou (Frankreich) gefunden. Sie sollten unbedingt die wunderschöne Arbeit [QG90] lesen. Dort wird eine Methode vorgestellt, wie Circe ihr Geheimnis vor Ulysses bewahren kann. Das Geheimnis besteht darin, dass sie weiß, wie eine Magische Tür geöffnet wird. Das Gebäude mit der magischen Tür hat einen raffinierten Grundriss (Abb. 4.19).

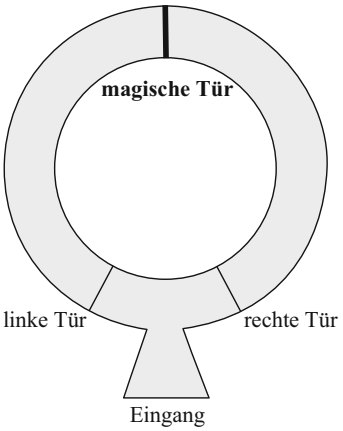


Abb. 4.19 Die Magische Tür

Das Protokoll ist das folgende: Circe geht durch die Eingangstür und entscheidet sich dann, nach links oder rechts zu gehen, tut das und schließt die entsprechende Tür hinter sich zu. Jetzt darf auch Ulysses durch die Eingangstür das Gebäude betreten, aber er muss dann stehen bleiben und weiß also nicht, hinter welcher Tür sich Circe versteckt. Jetzt denkt er sich „rechts“ oder „links“ und fordert Circe durch einen Ruf auf, durch die entsprechende Tür wieder zu erscheinen. Falls Circe die magische Tür wirklich öffnen kann, kann sie Ulysses immer zu Willen sein; falls sie das Geheimnis aber nicht kennt, hat sie nur mit der Wahrscheinlichkeit 1/2 Erfolg.
[Eine ausführliche Darstellung auf Deutsch finden Sie in [BS93].]

15 Bestimmen Sie diejenigen Zahlen, die mod 11 Quadrate sind. [Zum Beispiel ist 5 ein Quadrat mod 11, da $4^2 \bmod 11 = 5$ ist.]

16 (a) Überzeugen Sie sich, dass man die Quadratwurzel einer Zahl mod 11 erhält, indem man die Zahl zur dritten Potenz mod 11 erhebt. Mit anderen Worten: Diejenige Zahl y, die durch
$y := x^3 \bmod 11$
definiert ist, hat die Eigenschaft $y^2 \bmod 11 = x$.

(b) Was passiert im Fall $x = 2$? [Hinweis: Konsultieren Sie die vorige Aufgabe.]

17 Im Allgemeinen ist folgende Tatsache richtig: Sei p eine Primzahl mit $p \bmod 4 = 3$, und sei x eine natürliche Zahl derart, dass x ein Quadrat modulo p ist. Dann erhält man eine Quadratwurzel von $x \bmod p$, indem man x zur $(p + 1)/4$-ten Potenz mod p erhebt.
Überprüfen Sie diese Behauptung an mindestens zwei von 11 verschieden Primzahlen.
[Falls Sie wissen, dass die Zahlen $1, 2, \ldots, p - 1$ bei Multiplikation modulo p eine zyklische Gruppe bilden, können Sie die Behauptung auch *beweisen*.]

18 Warum wurde im Fiat-Shamir-Protokoll Wert auf die Tatsache gelegt, dass x teilerfremd zu n ist? [Hinweis: Betrachten Sie den Fall $x = 0$.]

19 Was ist die Funktion der Zufallszahl r im Fiat-Shamir-Protokoll?
Zeigen Sie: Wenn B die Zahl r kennt, kann er sein Bit b so schlau wählen, dass er das Geheimnis s berechnen kann.

20 Überzeugen Sie sich von folgender Eigenschaft des Fiat-Shamir-Protokolls: Wenn Mr. X sich als A ausgeben möchte, ohne das Geheimnis s von A zu kennen, kann ihm das in jeder Runde mit der Wahrscheinlichkeit ½ gelingen.
Zeigen Sie dazu folgendes: Angenommen, Mr. X weiß, welches Bit b ihm geschickt werden wird. Dann kann er y und daraus abgeleitet ein x so wählen, dass der Verifikation auf Seiten von B nichts im Wege steht.

5

Die Zukunft hat schon begonnen oder Public-Key-Kryptographie

Der Regen fließt eben herunter
Und fließt eben nicht hinauf
(Bertolt Brecht).

Bei den bisher behandelten Kryptosystemen ist uns schon mehrfach zweierlei aufgefallen:

- Bei einem Chiffriersystem gilt: Wer verschlüsseln kann, kann auch entschlüsseln; bei einem Authentifikationsverfahren führen beide Seiten dieselbe Prozedur durch.
- Je zwei Partner müssen einen gemeinsamen geheimen Schlüssel austauschen.

Die zweite Eigenschaft ist zweifellos ein Nachteil, während wir gewohnt sind, die erste (die die bislang behandelten *symmetrischen* Kryptosysteme definiert) als Vorteil anzusehen; denn sie sagt ja zum Beispiel, dass man zum Ver- und Entschlüsseln dieselbe Maschine verwenden kann. Die asymmetrischen Verfahren zeichnen sich dadurch aus, dass sie von der ersten Eigenschaft so weit wie möglich entfernt sind. Es wird sich zeigen, dass solche Systeme dann die zweite Eigenschaft nicht haben, dass also das Schlüsselmanagement, jedenfalls im Prinzip, sehr einfach wird.

Wir werden hauptsächlich den Begriff asymmetrischer Algorithmus verwenden; manchmal werden wir dafür auch die englische Formulierung Public-Key-Algorithmus benutzen. Diese Verfahren wurden 1976 von Whitfield Diffie und Martin Hellman in ihrer epochemachenden Arbeit *New Directions in Cryptography* [DH76] vorgeschlagen, einer Arbeit, die ihren Titel mit vollem Recht führt. Diffie schreibt in einem sehr empfehlenswerten Übersichtsartikel [Dif88]:

> Die Public-Key-Kryptographie wurde im Mai 1976 als Kind von zwei Problemen geboren, dem Schlüsselverteilungsproblem und dem Signaturproblem. Die Entdeckung bestand nicht in einer Lösung, sondern in der Erkenntnis,

A. Beutelspacher, *Kryptologie*, DOI 10.1007/978-3-658-05976-7_5,

dass die beiden Probleme, die beide aufgrund ihrer Definition unlösbar schienen – überhaupt lösbar sind und dass beide Probleme gemeinsam gelöst werden.

5.1 Public-Key-Kryptosysteme

In einem Public-Key-Kryptosystem hat jeder Teilnehmer T ein *Paar* von Schlüsseln:

- einen *öffentlichen* Schlüssel $E = E_T$ („Encryption“) zur Verschlüsselung und
- einen privaten (geheimen) Schlüssel $D = D_T$ („Decryption“) zum Entschlüsseln,

die sich durch die folgende entscheidende *Public-Key-Eigenschaft* auszeichnen:

- *Aus der Kenntnis des öffentlichen Schlüssels* E_T *ist der private Schlüssel* D_T *nicht zu erschließen*.

Ein System mit dieser Eigenschaft heißt *asymmetrisches Kryptosystem* oder *Public-Key-Kryptosystem*.

Alle öffentlichen Schlüssel sind in einer allgemein zugänglichen Datei (nach dem Modell des Telefonbuchs) untergebracht, während die privaten Schlüssel geheim sind, also nur ihren Besitzern (beziehungsweise deren Computern) bekannt sind.

An Abb. 5.1 kann man sich das einigermaßen vorstellen. Eigentlich müssten die Fensterläden aus Geheimhaltungsgründen geschlossen sein, aber dann würde man nichts mehr sehen.

Ein Public-Key-Kryptosystem an sich ist noch nicht aufregend. Spannend wird es, wenn es zum Verschlüsseln benutzt werden soll. Dazu definieren wir: Ein asymmetrisches Kryptosystem heißt *Public-Key-Verschlüsselungssystem* (*asymmetrisches Verschlüsselungssystem*), falls für jede Nachricht m gilt:

$$\text{Wenn } c = E(m) \text{ ist, dann gilt } D(c) = m.$$

Das bedeutet: D macht die Wirkung von E rückgängig. Mit anderen Worten: $D(E(m)) = m$.

Das heißt also schlicht: Wenn man auf eine Nachricht den öffentlichen Schlüssel E anwendet (encryption) und auf das Ergebnis den privaten Schlüssel D (decryption), dann erhält man wieder die Nachricht. Anders gesagt: Die

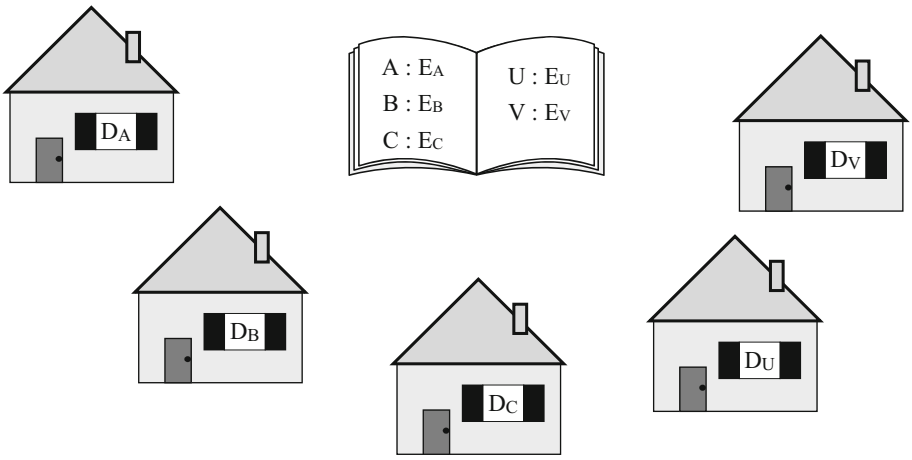


Abb. 5.1 Private und öffentliche Schlüssel

mit E verschlüsselte Nachricht wird mittels D wieder korrekt entschlüsselt. Diese Forderung ist nicht mehr als recht und billig.

Auf den ersten Blick scheint es völlig unglaublich zu sein, dass es Public-Key-Verschlüsselungssysteme gibt. Also Systeme mit Public-Key-Eigenschaft, die eine korrekte Entschlüsselung ermöglichen. Wir werden aber später sehen, dass tatsächlich Realisierungen existieren.

Ein asymmetrisches Kryptosystem heißt *Signaturschema*, falls für jede Nachricht m mit Hilfe des öffentlichen Schlüssels E überprüft werden kann, ob m und D(m) zusammenpassen. Man nennt in diesem Zusammenhang D(m) auch die *elektronische* (oder *digitale*) *Signatur* und schreibt D(m) = sig.

Hier sind folgende *Bemerkungen* angebracht.

- Es bietet sich in diesem Zusammenhang an, für den mit E verschlüsselten Geheimtext E(m) zu schreiben. Das heißt, wir fassen E auch als die Chiffrierfunktion auf, die Klartext in Geheimtext überführt. Diese Schreibweise unterscheidet sich leicht von der in Kap. 3 verwendeten.
- Die Definition eines asymmetrischen Signaturschemas wurde absichtlich so allgemein gehalten. Wir werden später sehen, wie sich die Forderung konkretisieren lässt, dass die Nachricht m, die Signatur sig = D(m) und der öffentliche Schlüssel E „zusammenpassen“ sollen.
- Es gibt asymmetrische Verschlüsselungsschemata, asymmetrische Signaturschemata und es gibt Systeme, die beide Aufgaben zugleich erfüllen.

Stellen wir uns vor, dass wir ein asymmetrisches Verschlüsselungssystem haben. Die öffentlichen und die privaten Schlüssel müssen bereitgestellt sein,

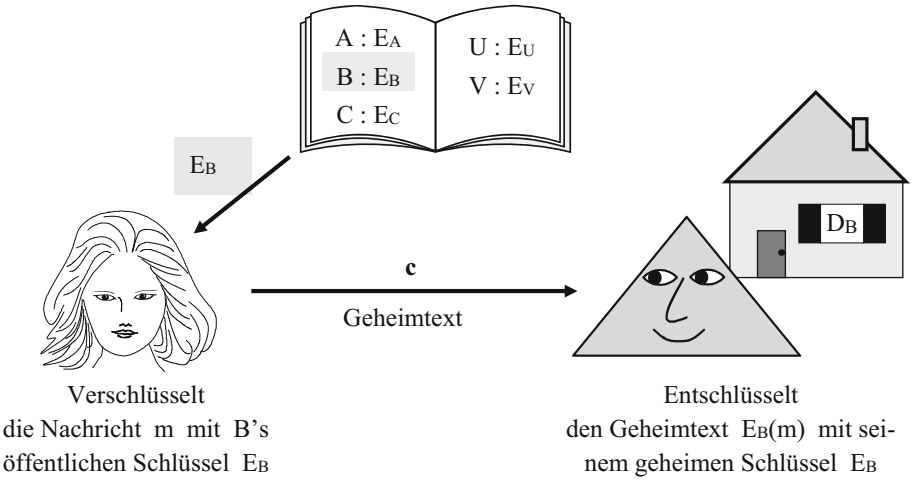


Abb. 5.2 Asymmetrisches Verschlüsselungssystem

um mit der Verschlüsselung überhaupt anfangen zu können. Diese erfolgt in drei Schritten (siehe Abb. 5.2):

1. Wenn A an B eine Nachricht m senden will, so
   - sucht sie den öffentlichen Schlüssel E_B von B,
   - verschlüsselt die Nachricht m mittels E_B und …
   - sendet den Geheimtext $c = E_B(m)$ an B.
2. B kann den Geheimtext c entschlüsseln, indem er seinen privaten Schlüssel D_B anwendet:
   $D_B(c) = D_B(E_B(m)) = m$.
3. Kein anderer Teilnehmer kann c entziffern, da er nach Voraussetzung aus der Kenntnis von E_B nicht auf D_B schließen kann.

Algorithmus 5.1: Public-Key-Verschlüsselung

Chiffrieren:	Um eine Nachricht zu verschlüsseln, wendet man den öffentlichen Schlüssel des Empfängers auf die Nachricht an.
Dechiffrieren:	Um eine für den Empfänger B chiffrierte Nachricht c zu dechiffrieren, wendet B auf c seinen privaten Schlüssel an.

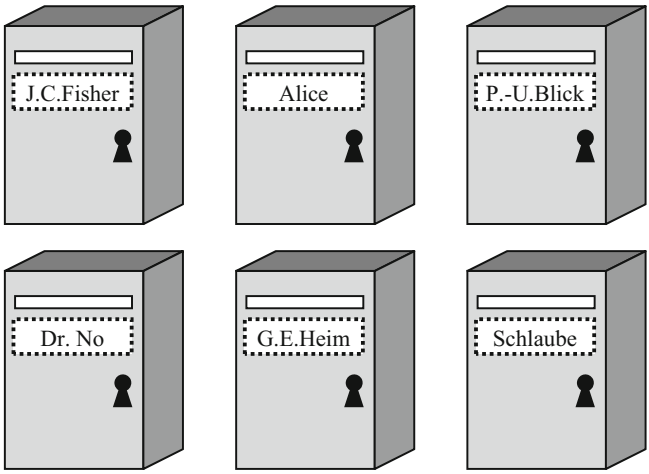


Abb. 5.3 Briefkästen

Man beachte, dass nur Schlüssel von B verwendet werden. Insbesondere muss A keinen eigenen Schlüssel besitzen, um für einen Empfänger eine Nachricht zu verschlüsseln.

Auf den ersten Blick erscheint ein Public-Key-Verschlüsselungsverfahren vielleicht verwirrend und kompliziert. Um seine grundsätzliche Einfachheit (das heißt, seine Genialität) einsehen zu können, illustrieren wir es mit einem Beispiel aus alltäglichen Begriffen. Das Beispiel zeigt auch, dass die Idee der Public-Key-Kryptographie im Grunde schon vor 1976 präsent war.

Stellen wir uns eine Reihe von Briefkästen wie in Abb. 5.3 vor. Jeder Teilnehmer an unserem Spielkommunikationssystem hat einen Briefkasten mit Namensschild und Schloss und einem zugehörigen Schlüssel. Jeder Schlüssel passt nur zu dem zugehörigen Briefkasten. Die individuellen Briefkastenschlüssel entsprechen den geheimen Schlüsseln.

Wenn jemand eine Nachricht an Frau Heim schicken möchte, so steckt er die Nachricht in Frau Heims Briefkasten. Dies entspricht der Verschlüsselung, und in der Tat ist die Nachricht jetzt nicht mehr lesbar. Man beachte die Analogie zur Grundeigenschaft eines asymmetrischen Kryptosystems: Das Verschlüsseln ist ganz einfach, aber die inverse Operation, also das Aufschließen mit Hilfe des zugehörigen Schlüssels wird dadurch um keinen Deut einfacher!

Jetzt kann jeder, auch der Sender, versuchen, mit seinem Schlüssel an dem Briefkasten herumzufummeln – ohne Erfolg! Nur Frau Heim persönlich kann das Schloss öffnen und zwar ohne jede Mühe.

Dieses Beispiel mag banal erscheinen; es zeigt aber auf sehr schöne Art die Wirkungsweise einer Verschlüsselung mit öffentlichen Schlüsseln.

Ein asymmetrisches Verschlüsselungssystem bietet folgende ins Auge springenden Vorteile:

- Kein Schlüsselaustausch ist notwendig!

Das alte Hauptproblem der symmetrischen Verfahren wurde also mit einem Schlag äußerst elegant und wirkungsvoll gelöst. Eine Konsequenz ist, dass mit Hilfe eines asymmetrischen Verfahrens eine *spontane Kommunikation* möglich wird. Ich kann also mit einer wildfremden Person Geheimnisse austauschen, ohne vorher lange Verabredungen über Schlüssel treffen zu müssen. Man sagt auch: Asymmetrische Verfahren sind ideal für *offene* Kommunikation, also solche Systeme, in denen die Partner nicht erst einen Vertrag schließen müssen, bevor sie miteinander kommunizieren können.

- Man braucht sehr wenige Schlüssel!

Bei symmetrischen Verfahren müssen bekanntlich je zwei Teilnehmer einen Schlüssel austauschen; n Teilnehmer brauchen $n(n-1)/2$ Schlüssel. Die Anzahl der Schlüssel steigt somit *quadratisch* mit der Zahl der Teilnehmer. Demgegenüber braucht bei einem asymmetrischen Verfahren jeder Teilnehmer nur zwei Schlüssel, von denen nur einer geheimgehalten zu werden braucht. Die Anzahl der Schlüssel ist also gerade doppelt so groß wie die Anzahl der Teilnehmer. Beispielsweise brauchen 1001 Teilnehmer in einem symmetrischen System 500.500 Schlüssel, in einem asymmetrischen aber nur 2002, weniger als ein halbes Prozent.

- Neue Teilnehmer können ohne Probleme hinzugefügt werden.

Wenn ein neuer Teilnehmer zu einem symmetrischen System hinzukommt, müssen alle alten Teilnehmer einen Schlüssel mit ihm austauschen. In einer asymmetrischen Situation hingegen brauchen die alten Teilnehmer ihre Daten nicht zu aktualisieren;

- Manche asymmetrischen Kryptosysteme bieten eine ausgezeichnete Möglichkeit für eine *elektronische Unterschrift.*

Was das heißt, werden wir in Abschn. 5.2 erörtern. Zuvor muss jedoch noch auf Nachteile hingewiesen werden. die diese Verfahren leider auch haben.

- Bislang ist kein asymmetrisches Kryptosystem bekannt, das sowohl als sicher gilt, als auch wirklich schnell ist.

Die wichtigsten Verfahren sind der RSA-Algorithmus, der vor allen andern die Diskussionen beherrscht, und seit einer Reihe von Jahren auch Algorithmen. die auf dem „diskreten Logarithmus“ beruhen. Wir werden beide Verfahren behandeln.

- Ein weiterer Nachteil liegt darin, dass asymmetrische Algorithmen – entgegen dem ersten Anschein – auch ein gewisses Schlüsselmanagement benötigen.

Auf den ersten Blick sieht alles ja ganz einfach aus: Jeder neue Teilnehmer wählt sich selbst ein Schlüsselpaar (oder bekommt eines geliefert), und damit hat sich's. Denkste! Denn: Was passiert, wenn ich (an unserem Beispiel argumentiert) meinen Briefkasten mit dem (falschen!) Namensschild „Heim“ versehe, zu dem aber nur mein Schlüssel passt. Dann kann ich sämtliche Nachrichten. die für Frau Heim bestimmt sind, abfangen.

Diesem Dilemma kann man entkommen, indem man eine trickreiche Schlüsselverwaltung mit Hilfe so genannter „Zertifikate“ durchführt, die gewährleistet, dass die Schlüssel authentisch sind. Dies wird in Abschn. 5.6 erläutert.

5.2 Die elektronische Signatur

Eine weitere, äußerst wichtige Idee von Diffie und Hellman ist die elektronische (oder digitale) Unterschrift. Zuerst machen wir uns die Eigenschaften einer gewöhnlichen handschriftlichen Unterschrift klar.

Nehmen wir an, eine Ärztin, Frau Dr. Abele, hat ihre Unterschrift unter ein Rezept gesetzt. Dann hat diese Unterschrift im Idealfall folgende Eigenschaften:

- Nur Frau Dr. Abele kann diese Unterschrift produzieren,
- jeder andere kann verifizieren, dass diese Unterschrift von Frau Dr. Abele stammt.

Es ist klar, dass Frau Dr. Abele über ein gewisses Etwas verfügen muss, sonst könnte ja jeder, insbesondere Mr. X, ihre Unterschrift produzieren. Sie braucht also ein Geheimnis. Mit Hilfe eines solchen geheimen Schlüssels kann sie auch unter Einsatz eines asymmetrischen Signaturschemas eine elektronische Unterschrift auf folgende Weise realisieren (siehe Abb. 5.4):

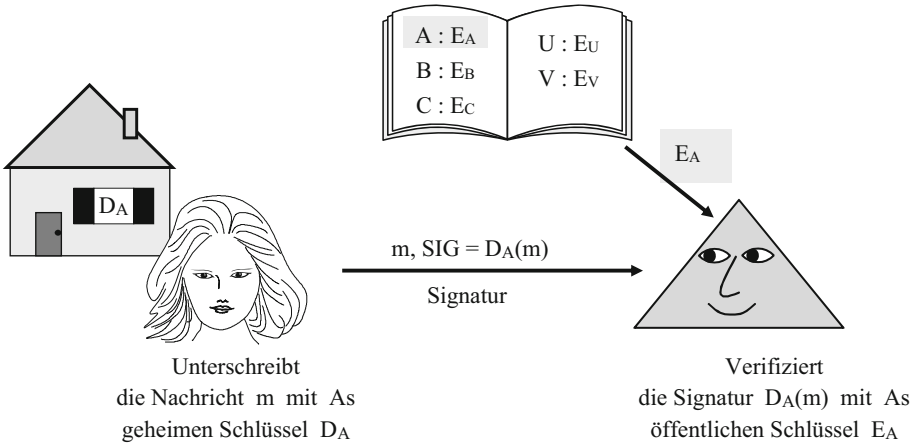


Abb. 5.4 Asymmetrisches Signaturschema

1. Wenn Frau Dr. Abele eine Nachricht m signieren möchte, so

   - wendet sie den (nur ihr bekannten) privaten Schlüssel D_A auf m an,
   - veröffentlicht die signierte Nachricht sig = D_A(m).

2. Jeder andere Teilnehmer kann diese Signatur sig verifizieren (vergleiche Abb. 5.4), indem er mit dem öffentlichen Schlüssel E_A von Dr. Abele überprüft, ob m und sig zusammenpassen.

Algorithmus 5.2: Erstellen und Verifizieren einer elektronischen Signatur

Erstellen der Signatur:	Wenn A eine Nachricht signieren will, wendet sie ihren privaten Schlüssel auf die Nachricht an.
Verifizieren der Signatur:	Um eine von A signierte Nachricht m zu verifizieren, überprüft man, ob die Nachricht, die Signatur und der öffentliche Schlüssel von A „zusammenpassen".

Beim bekanntesten Signaturverfahren, dem RSA-Algorithmus (siehe Abschn. 5.3) erfolgt die Verifizierung so, dass *man* E_A *auf die Signatur* sig = D_A(m) *anwendet und überprüft, ob die folgende Gleichung gilt:*

$$E_A(\text{sig}) = E_A(D_A(\text{m})) = \text{m}.$$

Man spricht von einem Signaturverfahren *mit Nachrichtenrückgewinnung*. In einem solchen Falle gibt es manchmal noch eine andere Möglichkeit der Ve-

rifikation einer elektronischen Unterschrift: Der Unterschreibende veröffentlicht nur die Signatur sig, nicht aber die Nachricht m. *Wenn der Verifizierende beim Anwenden von* E_A *eine sinnvolle Nachricht erhält (etwa einen Teil eines Rezepts), so lässt er das als Beweis für die Korrektheit der elektronischen Unterschrift gelten*. Dies setzt natürlich voraus, dass m auch eine „sinnvolle" Nachricht ist. und nicht etwa nur eine zufällige Folge von Symbolen. Diese Methode kann zum Beispiel eingesetzt werden, wenn natürlichsprachiger Texte signiert wurden.

Man beachte, dass beim Erstellen und Verifizieren einer elektronischen Unterschrift nur Schlüssel, die zu dem Sender A gehören, benutzt wurden. Insbesondere braucht man keinen eigenen Schlüssel, um eine Signatur zu verifizieren. Jeder kann die elektronische Unterschrift verifizieren, also nachweisen, wer diese Nachricht erstellt und signiert hat. Obwohl die heutige Rechtsprechung die elektronische Unterschrift noch nicht ohne weiteres als Ersatz für die handschriftliche Unterschrift anerkennt, tut ein Richter gut daran, bei seiner Entscheidung die elektronische Unterschrift entsprechend zu würdigen: Es würde Frau Dr. Abele sehr schwer fallen, abzuleugnen, die entsprechende Nachricht signiert zu haben.

Der entscheidende Unterschied zur gewöhnlichen Unterschrift besteht darin, dass die elektronische Signatur sig untrennbar mit der Nachricht m verbunden ist. Demgegenüber wird die gewöhnliche Unterschrift ja der unterschriebenen Nachricht nur hinzugefügt. Als Konsequenz ergibt sich. dass man an der Signatur kein Iota ändern kann, ohne dass dies bemerkt würde. Wird nämlich an $\text{sig} = D_A(m)$ auch nur ein Bitchen geändert, so wird bei Anwendung des öffentlichen Schlüssels E_A die veränderte Signatur sig' und der Klartext m in aller Regel nicht zusammenpassen.

Leider kann man das Prinzip der elektronischen Unterschrift nicht mit dem Briefkastenbeispiel erklären. Denn das System muss die Eigenschaft haben, dass man die ursprüngliche Nachricht wieder erhält, wenn man zunächst den privaten und dann den öffentlichen Schlüssel anwendet. Im Beispiel würde das heißen, dass man zunächst einen Brief aus dem (leeren!) Briefkasten herausholt, und ihn dann wieder einwirft (Anwendung des öffentlichen Schlüssels), um einen offenen Briefkasten zu erhalten. Das Beispiel hinkt in diesem Fall deswegen so fürchterlich, weil hier die beiden Schlüsseloperationen nicht vertauschbar sind. Um diesem Mangel abzuhelfen, stellen wir nun einen Schließmechanismus vor, bei dem Aufschließen und Zuschließen vertauschbar sind; es handelt sich um den „asymmetrischen Riegel", einer Erfindung, die die Welt Herrn Peter A. Schlaube verdankt. Dieses technische Wunderwerk können Sie auf Abb. 5.5 bewundern.

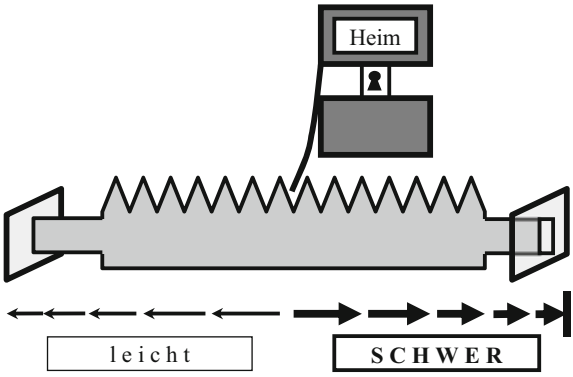


Abb. 5.5 Der asymmetrische Riegel

Sie sehen einen Riegel, der beidseitig schließt, d. h. sowohl, wenn er nach rechts geschoben wird, als auch wenn er nach links geschoben wird. Die Schieberichtung ist allerdings (für den Normalverbraucher) vorgegeben: Eine Blattfeder sichert, dass der Riegel nur linksseitig schließt. Der Eigentümer allerdings (in unserem Beispiel Frau G. E. Heim) kann mit ihrem Privatschlüssel den Sperrblock wegschließen und dann den Riegel unschwer nach rechts schieben.

Stellen wir uns vor, die Briefkästen in unserem Beispiel seien alle mit einem solchen asymmetrischen Riegel versehen. Das bedeutet, dass jeder Briefkasten geöffnet werden muss, um einen Brief einwerfen zu können – und dass dieses Öffnen durch einen asymmetrischen Riegel abgesichert ist. Wenn nun Frau Heim ein Dokument signieren will, so geht sie folgendermaßen vor: Sie legt das Dokument in ihren Briefkasten und schließt diesen, indem sie (nachdem sie zuvor den Sperrblock gelöst hat) den Riegel nach *rechts* schiebt. Anschließend setzt sie den Sperrmechanismus wieder in Kraft. Nun kann jeder ausprobieren, ob dieser verschlossene Briefkasten ein signiertes Dokument darstellt: Dies ist genau dann der Fall, wenn der Briefkasten dadurch geöffnet werden kann, dass der Riegel nach *links* geschoben wird (Anwenden des öffentlichen Schlüssels).

Sie haben bestimmt bemerkt, dass es in diesem Fall ein Problem ist, jedem zu erlauben, die Unterschrift zu verifizieren. Denn wenn die Unterschrift einmal verifiziert ist (d. h., wenn der Riegel einmal nach links geschoben wurde), so kann kein anderer diese Unterschrift mehr verifizieren.

Wenn nur eine Person, etwa Herr P.-U. Blick, die Unterschrift von Frau Heim überprüfen können soll, so muss sie folgendermaßen vorgehen: Sie unterschreibt die Nachricht; d. h. sie steckt die Nachricht in ihren Briefkasten, öffnet die Sperre und schließt den Briefkasten, indem sie den Riegel nach rechts schiebt. Nun verschlüsselt Frau Heim die unterschriebene Nachricht mit dem öffentlichen Schlüssel von Herrn Blick; Dazu gibt sie ihren ver-

schlossenen Briefkasten in den (gigantischen) Briefkasten von Herrn Blick und verschließt den äußeren Briefkasten, indem sie den Riegel nach links schiebt. Dann kann Herr Blick (und nur er!) die verschlüsselte unterschriebene Nachricht mit seinem geheimen Schlüssel entschlüsseln, er erhält die unterschriebene Nachricht und kann schließlich die Unterschrift verifizieren, indem er den öffentlichen Schlüssel von Frau Heim darauf anwendet. Wir verzichten darauf, dies zeichnerisch nachzuvollziehen.

5.3 Der RSA-Algorithmus

Der Leser wird bei aller Faszination, die diese Überlegungen zweifellos ausstrahlen. nun doch endlich wissen wollen, ob es konkrete Realisierungen eines asymmetrischen Verschlüsselungsverfahrens gibt. Denn so schön und technisch ausgefeilt die Schloss und Riegelbeispiele sind, so werden sie doch einen Menschen, der Sicherung elektronisch übermittelter und gespeicherter Daten betreibt, nicht restlos befriedigen. Gibt es eine „echte“ Realisierung asymmetrischer Kryptosysteme?

> 1977 nahmen die drei Personen, die den spektakulärsten Einzelbeitrag zur Public-Key-Kryptographie leisten sollten, Ronald Rivest, Adi Shamir und Leonard Adleman, die Herausforderung an und produzierten ein alle Erwartungen erfüllendes Public-Key-Kryptosystem. Der Prozess dauerte mehrere Monate, während derer Rivest Vorschläge machte, Adleman sie angriff und Shamir sich erinnert, zu beidem beigetragen zu haben. Im Mai 1977 wurden sie mit Erfolg gekrönt. Sie hatten entdeckt, wie ein einfaches Stück klassischer Zahlentheorie benutzt werden kann, um das Problem zu lösen [Dif88].

Es gibt eine ganze Reihe von asymmetrischen Kryptosystemen, zum Beispiel den Diffie-Hellman-Schlüsselaustausch (siehe Abschn. 5.4), der als erster publiziert wurde. Aber keiner ist so berühmt und so heiß diskutiert wie der RSA-Algorithmus, dessen Name aus den Initialen seiner Erfinder zusammengesetzt ist.

Um diesen Algorithmus zu beschreiben, brauchen wir ein bisschen Mathematik, insbesondere einen Satz des Schweizer Mathematikers Leonhard Euler (1707 bis 1783) über eine Eigenschaft natürlicher Zahlen. Für das weitere Verständnis sind nur die Ergebnisse der folgenden beiden Abschnitte wichtig, nicht die Einzelheiten der Erörterungen. Sie können die Abschn. 5.3.1 und 5.3.2 getrost überblättern. Sollten Sie anschließend doch noch Genaueres über den tatsächlichen Ablauf wissen wollen, steht Ihnen selbstverständlich die Möglichkeit offen, zurückzublättern und diese Abschnitte genau zu studieren.

5.3.1 Ein Satz von Euler

Der RSA-Algorithmus ist eine direkte Anwendung des Satzes von Euler. Für eine natürliche Zahl n definieren wir $\varphi(n)$ als die Anzahl der zu n teilerfremden natürlichen Zahlen, die nicht größer als n sind. Man nennt φ (sprich „phi“) die *Eulersche φ-Funktion*. Wir suchen also alle natürlichen Zahlen $\leq n$, deren größter gemeinsamer Teiler mit n gleich 1 ist.

Das klingt komplizierter als es ist. Überlegen wir uns einige Beispiele:

$$\varphi(1) = 1,\ \varphi(2) = 1,\ \varphi(3) = 2,\ \varphi(4) = 2,$$
$$\varphi(6) = 2,\ \varphi(10) = 4,\ \varphi(15) = 8.$$

Die letzte Aussage folgt so: Die zu 15 teilerfremden Zahlen ≤ 15 sind: 1, 2, 4, 7, 8, 11, 13 und 14. Das sind 8 Zahlen: also ist $\varphi(15) = 8$.

Nun machen wir uns einige allgemeine Aussagen klar:

1. *Wenn* p *eine Primzahl ist, so gilt*

$$\varphi(p) = p - 1.$$

   Da p eine Primzahl ist, ist jede der $p-1$ Zahlen 1, 2, 3, …, $p-1$ teilerfremd zu p.
2. *Wenn* p *und* q *zwei verschiedene Primzahlen sind, so gilt*

$$\varphi(pq) = (p-1)(q-1).$$

   Dies folgt so: Es gibt insgesamt genau $pq-1$ natürliche Zahlen, die kleiner als pq sind.

   Anstatt die zu pq teilerfremden Zahlen $\leq pq$ zu zählen, zählen wir – weil's einfacher ist – diejenigen Zahlen $\leq pq$, die *nicht* teilerfremd zu pq sind.

   Das sind zum einen die $q-1$ Vielfachen von p, nämlich

$$p, 2p, 3p, \ldots, (q-1)p,$$

und zum andern die $p-1$ Vielfachen

$$q, 2q, 3q, \ldots, (p-1)q.$$

von q.

Da dies alle Zahlen zwischen 1 und pq sind, die *nicht* teilerfremd zu pq sind, ergibt sich

$$\varphi(pq) = pq - 1 - (q-1) - (p-1) = pq - q - p + 1 = (p-1)(q-1).$$

Exkurs: Die Modulo-Rechnung.

Die Grundlage für fast alle Public-Key-Algorithmen ist die Modulo-Rechnung. Die Grundidee ist ganz einfach: Wir identifizieren eine natürliche Zahl mit ihrem Rest, der bei Division durch eine andere Zahl n entsteht. Das bedeutet: Wir teilen eine natürliche Zahl durch n und betrachten den Rest, der sich dabei ergibt.

Zum Beispiel ergibt 15 geteilt durch 7 den Rest 1. Entsprechend erhält man bei Division von 15 durch 8 den Rest 7 und bei Division von 15 durch 5 den Rest 0.

Für diesen Rest hat man ein besondere Schreibweise eingeführt, nämlich mod (gesprochen „modulo“). Wir schreiben 15 mod 7 für den Rest, der bei Division von 15 durch 7 entsteht. Also ist $15 \bmod 7 = 1$. Entsprechend ist $15 \bmod 8 = 7$ und $15 \bmod 5 = 0$.

Nochmals allgemein: Für natürliche Zahlen a und n ist a mod n diejenige Zahl, die sich als Rest, ergibt, wenn man a durch n teilt.

Man kann natürlich auch schreiben $(5 + 7) \bmod 3$ oder $5 \cdot 8 \bmod 7$ oder $2^5 \bmod 11$. Die Vorstellung ist dabei die, dass man zunächst die Rechenoperation (Addition, Multiplikation, Potenzierung) durchführt und dann die Modulo-Operation anwendet. Es ist also $(5 + 7) \bmod 3 = 12 \bmod 3 = 0$, $5 \cdot 8 \bmod 7 = 40 \bmod 7 = 5$, $2^5 \bmod 11 = 32 \bmod 11 = 10$.

Nun können wir den Satz von Euler formulieren:

Satz von Euler *Seien* m und n *zwei teilerfremde natürliche Zahlen. Dann gilt:*

$$m^{\varphi(n)} \bmod n = 1.$$

Das heißt: Wenn man $m^{\varphi(n)}$ *durch* n *dividiert, erhält man den Rest* 1.
Insbesondere gilt dann für jede natürliche Zahl k

$$m^{1+k\varphi(n)} \bmod n = m \cdot m^{k\varphi(n)} \bmod n = m \cdot 1 = m.$$

Für unsere Zwecke ist derjenige Fall besonders wichtig, in dem n das Produkt von zwei verschiedenen Primzahlen ist. Aufgrund des Satzes von Euler gilt in diesem Fall:

Folgerung aus dem Satz von Euler

Seien p *und* q *zwei verschiedene Primzahlen, und sei* m *eine natürliche Zahl* $\leq pq$. *Dann gilt für jede natürliche Zahl* k*:*

$$m^{k(p-1)(q-1)+1} \bmod pq = m.$$

Wir machen uns die Aussage des Satzes an zwei Beispielen klar: Einerseits kann man die Aussage

$$5^{\varphi(6)} \bmod 6 = 5^2 \bmod 6 = 25 \bmod 6 = 1$$

noch ziemlich leicht einsehen. Andererseits sagt einem der Satz aber auch, dass die folgende Beziehung richtig ist:

$$\begin{aligned} 31^{792} \bmod 851 &= 31^{22\cdot 36} \bmod 851 = 31^{\varphi(23\cdot 37)} \bmod 851 \\ &= 31^{\varphi(851)} \bmod 851 = 1. \end{aligned}$$

Man kann sich den Satz von Euler – wie viele Sätze der Mathematik – als kleinen Zaubertrick vorstellen. Der Zauberer bittet einen Freiwilligen aus dem Publikum um Mithilfe. Dieser darf sich eine natürliche Zahl m und eine Zahl k wählen. Dann muss der Freiwillige die Zahl $m^{k(p-1)(q-1)+1} \bmod pq$ berechnen. Armer Freiwilliger! Der Zauberer kann sich in aller Ruhe zurücklehnen, er weiß nämlich schon, was rauskommt: die vom Freiwilligen gewählte Zahl m.

Bemerkung: In der Folgerung muss m nicht notwendig teilerfremd zu $n = pq$ sein. Die Aussage gilt für *alle* Zahlen $m \leq n$.

Wegen der Wichtigkeit für den RSA-Algorithmus *beweisen wir die Folgerung aus dem Satz von Euler*. Wir setzen dazu den Satz von Euler für den einfachsten Fall, nämlich, dass n eine Primzahl ist, voraus. Diese Aussage heißt auch *kleiner Satz von Fermat*. In den Übungsaufgaben 13, 14 und 15 wird dieser Satz bewiesen.

Zur Abkürzung setzen wir $h := k(p-1)(q-1) + 1$. Dann ist zu zeigen: $m^h \bmod n = m$, mit anderen Worten: $(m^h - m) \bmod n = 0$. Dies geschieht in drei Schritten.

Schritt p. Es gilt

$$(m^h - m) \bmod p = 0.$$

Wir erinnern uns daran, dass $h := k(p-1)(q-1) + 1$ ist.

Wir wenden den kleinen Satz von Fermat an. Dazu brauchen wir die Voraussetzung, dass m und p teilerfremd sind. Das ist im Allgemeinen natürlich nicht richtig, da die Behauptung ja für *alle* Zahlen m gelten soll.

Was passiert, wenn m und p nicht teilerfremd sind? Da p eine Primzahl ist, muss dann notwendigerweise p ein Teiler von m sein. Wenn p die Zahl m teilt, so teilt p jede Potenz von m, insbesondere also die Zahl m^h. Also teilt p nicht nur m, sondern auch m^h, und somit $m^h - m$. Mit anderen Worten:

$$(m^h - m) \bmod p = 0.$$

Also gilt unsere Behauptung in diesem Spezialfall, und wir können („o. B. d. A.“, „ohne Beschränkung der Allgemeinheit“, wie der Mathematiker weise bemerken würde) nun wirklich voraussetzen, dass m und p teilerfremd sind. Dann sagt der kleine Satz von Fermat

$$m^{\varphi(p)} \bmod p = 1.$$

Da $\varphi(p) = p - 1$ ist, ergibt sich daraus sukzessive

$$\begin{aligned} m^h \bmod p &= m^{l+k\cdot\varphi(n)} \bmod p = m \cdot m^{k\cdot\varphi(n)} \bmod p = m \cdot m^{k\cdot(q-l)(p-l)} \bmod p \\ &= m \cdot (m^{p-l})^{k\cdot(q-l)} \bmod p = m \cdot 1^{k\cdot(q-l)} \bmod p = m \bmod p. \end{aligned}$$

Also gilt die Behauptung von Schritt p.

Wie der nächste Schritt lautet, ist wohl jedem klar; wir verzichten auf den Beweis:

Schritt q. Für jede natürliche Zahl m gilt

$$(m^h - m) \bmod q = 0.$$

Nun müssen wir nur noch die beiden Schritte zusammenführen:

Schritt n. Es gilt

$$m^h \bmod n = m.$$

Das folgt wie folgt: Nach den Schritten p und q gilt:

$$p \text{ teilt } (m^h - m) \text{ und } q \text{ teilt } (m^h - m)$$

Die beiden Primzahlen p und q teilen also dieselbe Zahl $z := m^h - m$. Da p und q *verschiedene* Primzahlen sind, muss dann auch ihr Produkt $p \cdot q$ die Zahl z teilen. Wenn wir das wieder zurückübersetzen, erhalten wir die Aussage

$$m^h - m \bmod p \cdot q = 0, \text{ oder } m^h \bmod n = m.$$

Das ist die Aussage von Schritt n, und damit ist auch die Aussage des Satzes bewiesen.

Für eine umfassendere Darstellung der Zahlentheorie sei Ihnen [BRK08] empfohlen.

5.3.2 Der euklidische Algorithmus

Wir werden den euklidischen Algorithmus benutzen, um den öffentlichen und den privaten Schlüssel der Teilnehmer zu berechnen. Das ursprüngliche Ziel des euklidischen Algorithmus ist es, den größten gemeinsamen Teiler

ggT(a, b) zweier natürlicher Zahlen a und b ($a \geq b$) zu berechnen. Vielleicht denken Sie: Es ist doch ganz einfach, den größten gemeinsamen Teiler zu bestimmen: Man nehme die Primfaktorzerlegung von a und b her, und suche sich die gemeinsamen Faktoren. Auf den ersten Blick (das heißt: bei kleinen Zahlen) scheint diese Methode auch gut zu sein. Wir werden aber im weiteren Verlauf dieses Kapitels zweierlei lernen:

- Es ist äußerst schwierig, die Primfaktorzerlegung einer (großen) natürlichen Zahl zu finden.
- Es ist mit dem euklidischen Algorithmus sehr einfach, den größten gemeinsamen Teiler zu ermitteln. Dieser zweiten Behauptung wollen wir uns zuerst zuwenden.

5.3.2.1 Berechnung des ggT

Die effizienteste Methode, den größten gemeinsamen Teiler zweier Zahlen zu bestimmen, ist *nicht* die über die Primfaktorzerlegung. Mit Hilfe des *Euklidischen Algorithmus* (das heißt: Division mit Rest) ist dies viel einfacher.

Wir beginnen mit einem Beispiel. Um den ggT der Zahlen $a = 792$ und $b = 75$ zu bestimmen, teilen wir 792 durch 75 mit Rest, und führen dieses Verfahren dann weiter:

$$\begin{aligned} 792 &= 10 \cdot 75 + 42 \\ 75 &= 1 \cdot 42 + 33 \\ 42 &= 1 \cdot 33 + 9 \\ 33 &= 3 \cdot 9 + 6 \\ 9 &= 1 \cdot 6 + 3 \\ 6 &= 2 \cdot 3. \end{aligned}$$

Also ist 3 der größte gemeinsame Teiler von 792 und 75. Dies erkennt man, wenn man sich folgendes klar macht:

Seien q und r nichtnegative ganze Zahlen mit $a = bq + r$. Wir teilen also a durch b mit Rest. Dann ist $ggT(a, b) = ggT(b, r)$. (Siehe dazu Übungsaufgabe 4.) Daraus ergibt sich, dass in obiger Gleichungskette sukzessive folgt:

$$\begin{aligned} ggt(792, 75) &= ggT(75, 42) \\ &= ggT(42, 33) \\ &= ggT(33, 9) \\ &= ggT(9, 6) \\ &= ggT(6, 3) \\ &= 3. \end{aligned}$$

Algorithmus 5.3: Berechnung des ggT

Seien a und b nichtnegative ganze Zahlen. Man bestimmt natürliche Zahlen q und r mit

$$a = bq + r \text{ und } 0 \leq r \leq b - 1.$$

Dann ist ggT(a, b) = ggT(b, r).

Anschließend wendet man das Verfahren auf b und r an. Dies führt man so lange fort, bis man den ggT direkt bestimmen kann.

Für Informatikfreaks und zur Vorbereitung von Übungsaufgabe 4 soll der euklidische Algorithmus programmnah beschrieben werden:

Read a, b
While $b \neq 0$ do
  $r := a \bmod b$
  $a := b, b := r$
Write a.

5.3.2.2 Berechnung der modularen Inversen

Für unsere Zwecke ist die folgende Aussage von größter Wichtigkeit:

Satz von der Vielfachsummendarstellung *Sei* d *der größte gemeinsame Teiler der Zahlen* a *und* b. *Dann gibt es ganze Zahlen* x *und* y *mit der Eigenschaft, dass*

$$d = xa + yb$$

ist. Eine solche Darstellung nennt man *Vielfachsummendarstellung* des größten gemeinsamen Teilers d von a und b.

Beispiel: Seien a = 71, b = 15. Zunächst berechnen wir den größten gemeinsamen Teiler von a und b:

$$\begin{aligned} 71 &= 4 \cdot 15 + 11 \\ 15 &= 1 \cdot 11 + 4 \\ 11 &= 2 \cdot 4 + 3 \\ 4 &= 1 \cdot 3 + 1. \end{aligned}$$

Nun lösen wir diese Gleichungskette von unten nach oben auf. Allerdings rechnen wir die Gleichungen nicht aus – sonst würde sich nur das nutzlose

1 = 1 ergeben – sondern fassen die Terme nur geeignet zusammen:

$$\begin{aligned} 1 &= 4 - 1 \cdot 3 \\ &= 4 - 1 \cdot (11 - 2 \cdot 4) = 3 \cdot 4 - 1 \cdot 11 \\ &= 3 \cdot (15 - 1 \cdot 11) - 1 \cdot 11 = 3 \cdot 15 - 4 \cdot 11 \\ &= 3 \cdot 15 - 4 \cdot (71 - 4 \cdot 15) = 19 \cdot 15 - 4 \cdot 71. \end{aligned}$$

Also gilt

$$1 = 19 \cdot 15 + (-4) \cdot 71.$$

Wir machen uns nur das Prinzip des Beweises klar. Sei $a = bq + r$. Wenn wir eine Vielfachsummendarstellung für b und r haben, dann können wir daraus eine für a und b ableiten. Dies ist genau das Prinzip des „Hinaufhangelns", welches wir am Beispiel durchgeführt haben.

Sei also $1 = xr + yb$ mit ganzen Zahlen x und y. Wegen $r = a - bq$ können wir diese Gleichung wie folgt umschreiben:

$$1 = x(a - bq) + yb.$$

Wir lösen die Klammer auf und erhalten mit

$$1 = xa + (y - xq)b = xa + y'b$$

eine Vielfachsummendarstellung für a und b.

Das Verfahren, mit dem wir die Vielfachsummendarstellung erhalten haben, nennt man auch den *erweiterten euklidischen Algorithmus*.

Überzeugen Sie sich selbst davon, dass dieses Verfahren weit weniger Furcht erregend ist als es aussieht: Lösen Sie die Übungsaufgaben 5 und 6.

Die nächste Folgerung ist die (für unsere Absichten) wichtigste Aussage dieses Abschnitts.

Satz von der modularen Inversen *Sind* a *und* n ***teilerfremde*** *Zahlen (das bedeutet, dass ihr größter gemeinsamer Teiler* 1 *ist), so gibt es eine ganze Zahl* b *mit der Eigenschaft*

$$a \cdot b \bmod n = 1.$$

Das bedeutet, dass $a \cdot b$ *geteilt durch* n *den Rest* 1 *hat. Vornehm ausgedrückt*: a *ist modulo* n *invertierbar*.

Diese Tatsache folgt mit unserem jetzigen Wissensstand ganz einfach: Da der ggT von a und n gleich 1 ist, gibt es ganze Zahlen x und y mit folgender Eigenschaft:

$$1 = a \cdot x + n \cdot y.$$

Da ny durch n teilbar ist, ergibt $a \cdot x$ bei Division durch n den Rest 1, also ist

$$a \cdot x \bmod n = 1.$$

Mit $b = x$ folgt die Behauptung.

Algorithmus 5.4: Berechnung der modularen Inversen

Seien a und n ganze Zahlen mit ggT(a, n) = 1.
Man berechne zunächst die Vielfachsummendarstellung $1 = xa + yn$. Dann ist x die modulare Inverse von a modulo n.

Beispiel: Was ist die Inverse von 15 modulo 71?

$$\text{Wegen } 1 = 19 \cdot 15 + (-4) \cdot 71.$$
$$19 \cdot 15 \bmod 71 = (1 + 4 \cdot 71) \bmod 71 = 1.$$

Also ist 19 die modulare Inverse von 15 modulo 71.

Die Herleitungen dieses Abschnitts mögen Ihnen vielleicht langwierig vorgekommen sein. Deshalb seien die Erkenntnisse, die wir erhalten haben, noch einmal betont:

- *Der euklidische Algorithmus kann von einem Rechner äußerst bequem ausgeführt werden. Der größte gemeinsame Teiler zweier Zahlen kann einfach berechnet werden. Insbesondere kann leicht entschieden werden, ob zwei Zahlen den größten gemeinsamen Teiler* 1 *haben.*
- *Wenn* a *und* n *teilerfremde Zahlen sind, so kann man leicht eine Zahl* b *berechnen mit der Eigenschaft*

$$a \cdot b \bmod n = 1.$$

5.3.3 Schlüsselerzeugung

Nun kehren wir zur Beschreibung des RSA-Algorithmus zurück. Bevor wir die RSA-Verschlüsselung beschreiben, müssen wir die Voraussetzungen an das System klären.

Zunächst muss jeder Teilnehmer ein Schlüsselpaar bestehend aus einem privaten und dem zugehörigen öffentlichen Schlüssel erhalten. Grundsätzlich kann entweder jeder Teilnehmer selbst seine Schlüssel erzeugen, oder die Schlüsselerzeugung kann zentral von einer Schlüsselerzeugungszentrale erfolgen. Wir benutzen im Folgenden das neutrale „man“.

Man wählt für jeden Teilnehmer zwei große Primzahlen p und q und bildet ihr Produkt n = pq. Sodann berechnet man

$$\varphi(n) = \varphi(pq) = (p-1)(q-1).$$

(siehe Abschn. 5.3.2). Schließlich sucht man zwei Zahlen e und d mit

$$e \cdot d \bmod \varphi(n) = 1.$$

Danach werden dem Teilnehmer die Zahlen e und n als *öffentlicher Schlüssel* zugewiesen und die Zahl d als *geheimer Schlüssel* mitgeteilt.

Algorithmus 5.5: Schlüsselerzeugung für den RSA-Algorithmus

Man wählt zwei verschiedene große Primzahlen p und q.
Man berechnet das Produkt n = pq.
Man berechnet $\varphi(n) = (p-1)(q-1)$.
Man wählt eine Zahl e, die teilerfremd zu $\varphi(n)$ ist und berechnet d, so dass gilt

$$ed \bmod \varphi(n) = 1.$$

Privater Schlüssel: d (zusammen mit n).
Öffentlicher Schlüssel: e, n
Geheime Parameter bei der Schlüsselerzeugung: p, q, $\varphi(n)$.

Bemerkungen: Eine der beiden Zahlen e und d kann frei gewählt werden. Es ist sinnvoll, e zu wählen, und daraus d zu berechnen; denn so wird d eine „allgemeine Zahl" $\leq n$.

Man kann e so wählen, dass die Berechnung der Potenzen $m^e \bmod n$ leicht wird. Beispielsweise wurde mehrfach vorgeschlagen, die vierte Fermat-Zahl $F_4 = 2^{16} + 1 = 65.537$ für e zu wählen. Diese Zahl lautet in binärer Darstellung einfach 1 0000 0000 0000 0001; daher ist es vergleichsweise sehr einfach, mit dieser Zahl zu potenzieren (vergleiche dazu Abschn. 5.3.5). In einem solchen Fall müssen die Teilnehmer natürlich verschiedene Moduli n haben; damit ist dann aber schon gewährleistet, dass ihre geheimen Schlüssel verschieden sind. Auch ansonsten müssen verschiedene Teilnehmer auf jeden Fall verschiedene Moduli n haben, da sonst Angriffe möglich sind.

Wo liegen Probleme bei der Schlüsselerzeugung? Einfachere Frage: Wo liegen die Probleme *nicht?* Sie liegen nicht in der Berechnung von d und e. Man sucht sich nämlich zunächst ein e, das teilerfremd zu $\varphi(n)$ ist (dazu kann man z. B. jede Primzahl wählen, die größer als $\varphi(n)$ ist), und berechnet dann mit Hilfe des euklidischen Algorithmus die Zahl d (vergleiche Abschn. 5.3.2). Natürlich ist es auch kein Problem, $\varphi(n)$ zu berechnen, *wenn man* p *und* q *kennt!*

Hier kommen wir dem eigentlichen Problem schon näher; dies besteht nämlich darin, Primzahlen p und q zu finden. Dies ist ja bekanntlich eines der Hauptprobleme der Mathematik. Sie alle werden die Berichte in den Zeitungen kennen, die davon künden, dass wieder mal jemand den alten Weltrekord für große Primzahlen überboten hat. Der aktuelle Weltrekord aus dem Jahre 2013 besteht darin, nachgewiesen zu haben, dass die Zahl $2^{57.885.161} - 1$ eine Primzahl ist. Diese Zahl hat die unglaubliche Zahl von 17.425.170 Dezimalstellen.

Das macht Sie vielleicht dem RSA-Algorithmus gegenüber skeptisch; denn hier braucht man ja offenbar für jeden Benutzer zwei Primzahlen, das heißt man braucht Primzahlen wie Sand am Meer. Allerdings haben die Primzahlen, die man für den RSA-Algorithmus verwendet eine Größe von 100 bis 200 Dezimalstellen. Auch das ist eine gigantische Größenordnung, aber die gute Nachricht ist: Man ist nicht auf die singulären Weltrekordprimzahlen angewiesen, sondern bewegt sich in „mittleren Gefilden", in denen man hoffen kann, viele Primzahlen real zu finden.

Diese Hoffnung trügt nicht, denn uns kommen dabei zwei mathematische Erkenntnisse zu Hilfe. Zum einen gibt es tatsächlich außerordentlich viele Primzahlen. Der so genannte Primzahlsatz, ein Glanzstück der Mathematik des 19. Jahrhunderts, sagt, dass es unglaublich viele Primzahlen gibt. In dem Primzahlsatz wird das präzise durch eine Formel beschrieben, er sagt aber zum Beispiel, dass unter den 100-stelligen Zahlen im Durchschnitt immer noch jede 230-te Zahl (also jede 115-te ungerade Zahl) eine Primzahl ist.

Zum anderen kennt man sehr effiziente Primzahltests. Dazu gehören insbesondere probabilistische Tests, etwa der Miller-Rabin-Test. Diese weisen zwar nur mit einer gewissen Wahrscheinlichkeit nach, dass eine vorgelegte Zahl eine Primzahl ist, sind aber außerordentlich effizient.

Nun kombiniert man diese beiden Erkenntnisse: Man wählt im gewünschten Bereich eine Zufallszahl und testet, ob diese eine Primzahl ist. Wenn nicht, geht man zur nächsten zufällig gewählten Zahl über und testet diese auf Primalität. Und so weiter. Der Primzahlsatz garantiert nun, dass man so nach erstaunlich wenigen Versuchen eine Primzahl findet. Wir wollen das hier nicht vertiefen, sondern verweisen auf die Spezialliteratur [Gor85] und auch [Mau90].

5.3.4 Anwendung des RSA-Algorithmus

Soll an Frau G.E. Heim eine Nachricht geschickt werden, so muss man zunächst ihren öffentlichen Schlüssel in Erfahrung bringen. Dieser besteht, wie wir erörtert haben, aus dem *Modul* n und dem *Exponenten* e.

Tab. 5.1 Einige ASCII-Zeichen

Zeichen	ASCII-Zeichen	Zeichen	ASCII-Zeichen
Zwischenraum	00100000	H	01001000
!	00100001	I	01001001
0	00110000	J	01001010
1	00110001	K	01001011
2	00110010	L	01001100
3	00110011	M	01001101
4	00110100	N	01001110
5	00110101	O	01001111
6	00110110	P	01010000
7	00110111	Q	01010001
8	00111000	R	01010010
9	00111001	S	01010011
A	01000001	T	01010100
B	01000010	U	01010101
C	01000011	V	01010110
D	01000100	W	01010111
E	01000101	X	01011000
F	01000110	Y	01011001
G	01000111	Z	01011010

Nun wird die Nachricht dargestellt in Form einer oder mehrerer natürlicher Zahlen m, die kleiner als n sind. Dies kann man auf mehrere Arten erreichen. Die Standardmethode ist die folgende. Die Nachricht möge aus Buchstaben, Ziffern und Sonderzeichen (Punkt, Komma, Zwischenraum, usw.) bestehen. Jedes solche Zeichen kann durch ein bestimmtes Muster aus acht Bits (Nullen und Einsen) dargestellt werden. Meist wird hierzu das Standardsystem *ASCII* (gesprochen Asskie, American Standard Code for Information Interchange) benutzt.

In Tab. 5.1 finden Sie einige ASCII-Zeichen. Zum Beispiel wird die Aussage

ALLES KLAR!

übersetzt in

01000001 01001100 01001100 01000101 01010011 00100000
01001011 01001100 01000001 01010010 00100001

Damit ist klar, wie man Nachrichten als Folgen von Nullen und Einsen darstellen kann. Die Übersetzung einer solchen Folge in natürliche Zahlen ist dann einfach: Wenn n eine 1024-Bit Zahl ist (dies ist eine typische Größenordnung), so fasst man jeweils 128 Zeichen zusammen und erhält dadurch Blöcke von $128 \cdot 8 = 1024$ Bit. Jede dieser Bitfolgen wird dann als binäre Darstellung einer natürlichen Zahl m interpretiert.

Uns braucht die Art der Codierung aber nicht zu kümmern. Wir nehmen einfach an, dass unsere Nachricht eine natürliche Zahl $m < n$ ist.

Eine solche Zahl m wird *verschlüsselt*, indem sie mit e potenziert und dann modulo n reduziert wird. Mit anderen Worten: Die Zahl

$$c = m^e \bmod n,$$

ist der zum Klartext m gehörige Geheimtext.

Wie wird *entschlüsselt*? Dies muss so geschehen, dass nur der Empfänger, Frau Heim, dies tun kann. Sie macht dies, indem sie ihren geheimen Schlüssel d auf den Geheimtext c anwendet. Genauer gesagt geht sie wie folgt vor: Die Zahl

$$m' = c^d \bmod n,$$

ist der zu dem empfangenen Geheimtext c gehörige Klartext.

Algorithmus 5.6: Verschlüsselung mit RSA

Eine Nachricht m wird als Zahl $< n$ dargestellt.

Chiffrieren:	Der Sender verschlüsselt, indem er den Geheimtext $c = m^e \bmod n$ berechnet, wobei (e, n) der öffentliche Schlüssel des Empfänger ist.
Dechiffrieren:	Der Empfänger berechnet $m = c^d \bmod n$, wobei (d, n) sein privater Schlüssel ist.

Eine Frage stellt sich an dieser Stelle sofort: Für Frau Heim ist dieser Dechiffrieralgorithmus natürlich nur dann von Wert, wenn korrekt dechiffriert wird, d. h. wenn ihr der ursprüngliche Klartext wieder geliefert wird, wenn also $m' = m$ ist. Dass dies so ist, wird in dem folgenden Satz festgestellt:

Satz vom korrekten Dechiffrieren *Für jede natürliche Zahl* $m < n$ *gilt* $m' = m$. *Mit anderen Worten: Der oben angegebene Dechiffrieralgorithmus arbeitet korrekt.*

Zum *Beweis* dieser Tatsache machen wir uns zunächst klar: $m' = m^{ed} \bmod n$. Dies folgt so:

$$m' = c^d \bmod n = (m^e)^d \bmod n = m^{e \cdot d} \bmod n.$$

Das heißt: Der Dechiffrieralgorithmus arbeitet genau dann korrekt, wenn

$$m^{e \cdot d} \bmod n = m$$

gilt.

Nach Definition von e und d gilt aber $e \cdot d = 1 + k\varphi(n)$. Damit ergibt sich zusammen mit der Folgerung aus dem Satz von Euler

$$m^{e \cdot d} \bmod n = m^{1+k\varphi(n)} \bmod n = m.$$

Somit ergibt sich nach dem Dechiffrieren mit dem privaten Schlüssel tatsächlich wieder m.

Eine charakteristische Eigenschaft des RSA-Algorithmus ist, dass der Übergang zum Signaturschema besonders einfach ist: Wenn A eine Nachricht m signieren möchte, so stellt sie diese zunächst als Zahl < n oder als ein oder eine Folge von solchen Zahlen dar. Sie erhält die Signatur, indem sie m mit ihrem privaten Schlüssel potenziert:

$$sig = m^d \bmod n.$$

Jeder kann die Signatur verifizieren, indem er auf diese den öffentlichen Schlüssel von A anwendet und das Ergebnis mit m vergleicht:

$$sig^e \bmod n = m?$$

Wenn sich A und der Verifizierer an das Protokoll hält, wird die Signatur zu Recht akzeptiert, denn es gilt

$$sig^e \bmod n = (m^d)^e \bmod n = m^{d \cdot e} \bmod n = m.$$

Algorithmus 5.7: Signatur mit RSA

Erstellen der Signatur:	Um eine Nachricht m zu signieren, berechnet A die Signatur $sig = m^d \bmod n$, wobei (d, n) der private Schlüssel von A ist.
Verifizieren der Signatur:	Eine von A erstellte Signatur wird verifiziert, indem überprüft wird, ob $sig^e \bmod n = m$ ist. Dabei ist (e, n) der öffentliche Schlüssel von A.

Man beachte, dass bei der Verifikation aus der Signatur sig die Nachricht m rekonstruiert wird; man spricht daher auch von einem *Signaturschema mit Nachrichtenrückgewinnung*.

5.3.5 Die Kunst zu potenzieren

Die rechentechnische Grundfunktion des RSA-Algorithmus (und vieler anderer Public Key-Algorithmen) ist die modulare Potenzierung, also die Berechnung von m^a mod n. In diesem Abschnitt diskutieren wir, wie man ma effizient berechnen kann.

Wie berechnet man m^3? Ganz einfach, man schreibt m dreimal hin, setzt Malpunkte dazwischen und rechnet es aus. Wie berechnet man m^{20}? Genau: Man schriebt m 20 mal hin, setzt Malpunkte dazwischen und rechnet das Ganze aus. Das mag noch gehen wenn a = 20 ist. Aber für a = 10.000 wird diese Methode schon sehr mühsam, und wenn a in der Größenordnung von 10^{100} ist, funktioniert diese Methode überhaupt nicht mehr. Denn man müsste m 10^{100} mal aufschreiben – und das ist vollkommen unmöglich, weil es im Universum gar nicht so viele Atome gibt Und auch 10^{100} mal multiplizieren ist völlig unmöglich.

Aber bei RSA kommen Exponenten in dieser Größenordnung vor. Was tun? Betrachten wir ein Beispiel. Wenn wir m^{37} ausrechnen wollen, können wir wie folgt vorgehen.

Wir stellen eine Tabelle auf. In die linke Spalte schreiben wir oben die Zahl 37. Darunter die Hälfte von 37, wobei wir großzügig abrunden: für uns ist die Hälfte von 37 gleich 18. Dann kommt die Hälfte von 18, also 9; dann die abgerundete Hälfte von 9, also 4, dann 2 und schließlich 1.

In der rechten Spalte begonnen wir oben mit m und schreiben dann darunter das Quadrat von m, darunter das Quadrat von m^2, also m^4, dann m^8, dann m^{16} und schließlich m^{32}. Jeder Eintrag entsteht aus dem darüberstehenden durch Quadrieren. Das bedeutet, dass wir, um diese Tabelle zu berechnen, insgesamt nur 5 mal multiplizieren müssen.

37	**m**
18	m^2
9	$\mathbf{m^4}$ $\mathbf{(= (m^2)^2)}$
4	m^8 $(= (m^4)^2)$
2	m^{16} $(= (m^8)^2)$
1	$\mathbf{m^{32}}$ $\mathbf{(= (m^{16})^2)}$

Nun betrachten wir in der linken Spalte die Zahlen, die *ungerade* sind (das sind die fettgedruckten Zahlen 37, 9 und 1). Wir schauen dann in diesen Zeilen nach rechts, also zu den Ausdrücken m, m^4 und m^{32} und multiplizieren diese. Das ergibt $m \cdot m^4 \cdot m^{32} = m^{1+4+32} = m^{37} \cdot 10$.

Dazu müssen wir zwei weitere Male multiplizieren. Das bedeutet: Um m^{37} auf diese Weise zu berechnen, brauchten wir nur 6 Multiplikationen – im

Gegensatz zu 36 bei der „naiven“ Methode. Man nennt dieses Verfahren den „square-and-multiply“-Algorithmus, weil man nur quadrieren und multiplizieren muss.

Um noch genauer zu sehen, wie diese Methode funktioniert, schreiben wir die Zahlen der linken Spalte noch in ihrer binären Form:

100101	**37**	**m**
10010	18	m^2
1001	**9**	$\mathbf{m^4}$
100	4	m^8
10	2	m^{16}
1	**1**	$\mathbf{m^{32}}$

An dieser Tabelle erkennt man verschiedenes: Wenn man die linke und die mittlere Spalte vergleicht, sieht man dass man die ungeraden Zahlen daran erkennen kann, dass sie in binärer Darstellung eine 1 am Ende haben. Auch das „großzügige Halbieren“ ist in binärer Form besonders einfach zu beschreiben: Man streicht einfach das letzte Bit.

Nun können wir auch den Aufwand, das heißt die Anzahl der Multiplikationen abschätzen, die man zur Potenzierung braucht. Man muss zunächst die Anzahl der Bits bestimmen, die man für die Darstellung des Exponenten braucht. Diese Anzahl gibt die Anzahl der Potenzen von m an, die man auf jeden Fall berechnen muss, um die Tabelle zu erhalten. Dann muss man im schlimmsten Fall noch genau so oft multiplizieren. Also ist die Anzahl der Multiplikationen höchstens 2 mal die Länge des Exponenten in Bit.

Einige Beispiel machen klar, wie effizient dieses Verfahren ist:

Ein Exponent a, der in der Größenordnung von 1000 ist, benötigt nur 10 Bits. Also braucht man maximal 20 Multiplikationen (und nicht 1000).

Wenn der Exponent in der Größenordnung von einer Million (ca. 10^{20}) ist, braucht man nur 40 Multiplikationen.

Nun betrachten wir einen Exponenten in der Größenordnung 10^{100}. Wie viele Bits brauchen wir zur Darstellung einer solchen Zahl? Es gilt: $10^{100} = (10^3)^{33} \cdot 10 = (2^{10})^{33} \cdot 10 = 2^{330} \cdot 10 \sim 2^{333}$. Die Zahlen in der Größenordnung 10^{100} sind also 333 Bit-Zahlen. Das heißt: um mit diesen zu potenzieren, braucht man höchstens 666 Multiplikation (im Gegensatz zu unvorstellbaren 10^{100} Multiplikationen bei der „naiven“ Methode).

5.3.6 Die Stärke des RSA-Algorithmus

Im vorigen Abschnitt haben wir uns klargemacht, dass der Empfänger einer Nachricht diese korrekt dechiffrieren kann. Aber warum sollte dies nur der

Empfänger können? Die Antwort scheint einfach zu sein: Zum Dechiffrieren braucht man ja den geheimen Schlüssel d – und dieser befindet sich allein in der Hand des Empfängers!

Wie gesagt, die Antwort *scheint* einfach zu sein. Denn man sollte zwei Fragen stellen:

Kann man aus der Kenntnis des öffentlichen Schlüssels den geheimen ableiten? Das heißt: Gilt die Public-Key-Eigenschaft? Es ist klar, dass man für den oben beschriebenen Dechiffrieralgorithmus den geheimen Schlüssel braucht. Aber es könnte ja sein, dass es *grundsätzlich andere Dechiffrierverfahren* gibt. Die zweite Frage lautet also: *Kann man dechiffrieren, ohne den geheimen Schlüssel zu benutzen* ?

Wir werden beide Fragen diskutieren, aber – das sei jetzt schon verraten – ohne zu einer endgültigen Klärung zu kommen.

Wir gehen davon aus, dass jedermann den öffentlichen Schlüssel, also die Zahlen e und n, kennt. Um zu dechiffrieren, müsste Mr. X die „e-te modulare Wurzel" aus einer Zahl (nämlich dem Geheimtext) berechnen. Eine Möglichkeit, dies zu tun, besteht darin, den geheimen Schlüssel, also die Zahl d, zu berechnen und den Geheimtext mit d zu potenzieren.

Wenn Mr. X auch die Zahl $\varphi(n)$ kennen würde, so hätte er leichtes Spiel: Er könnte dann nämlich genauso vorgehen wie die Schlüsselvergabezentrale und mit Hilfe des euklidischen Algorithmus die Zahl d berechnen.

So weit, so gut. Wir haben das Problem darauf reduziert, die Zahl $\varphi(n)$ zu berechnen. Wie könnte Mr. X das bewerkstelligen? Wenn er n in seine Primfaktoren p und q zerlegen könnte, so wäre (für ihn) alles gut. Dann hätte er $n = pq$ und folglich $\varphi(n) = (p-1)(q-1)$. Interessanterweise ist auch die Umkehrung richtig: Wenn man n und $\varphi(n)$ kennt, dann kann man n faktorisieren! In dieser Situation gelten nämlich die folgenden Gleichungen in den Unbekannten p und q:

$$pq = n \text{ und } (p-1)(q-1) = \varphi(n),$$

und man kann daraus die Unbekannten p und q bestimmen. Zusammenfassend können wir also feststellen: Für eine natürliche Zahl n, die das Produkt zweier Primzahlen ist, bedeutet es dasselbe, $\varphi(n)$ zu berechnen oder n zu faktorisieren..

Es ist also alles ganz einfach – wenn Mr. X die Zahl n faktorisieren kann. Und genau hier liegt der Hase im Pfeffer! Mit ungeübtem Auge ist das freilich auf den ersten Blick gar nicht zu sehen; denn landläufig geht man ja von der Vorstellung aus, dass es kein Problem ist, Zahlen, wie zum Beispiel 72, 123 oder auch 221, zu zerlegen. Doch schon bei etwas „größeren" Zahlen, wie etwa 1763 oder 8633 wird es deutlich schwieriger. Und wenn man daran denkt,

dass beim RSA-Algorithmus die Zahl n etwa 200 Dezimalstellen haben sollte, merkt man schon rein gefühlsmäßig, dass Faktorisieren von großen Zahlen ein sehr schwieriges Problem sein dürfte.

Nun muss ich zu meiner Schande bekennen, dass auch die Mathematik, die für solche Probleme ja zuständig ist, nicht viel mehr als dieses Gefühl äußern kann – allerdings außerordentlich präzise.

Was kann die Mathematik zu diesem Problem Positives sagen? Wie kann man eine große Zahl n faktorisieren? Nun, man kann systematisch probieren. Das heißt, dass man für jede Zahl $m \leq n$ testet, ob sie ein Teiler von n ist oder nicht. Natürlich muss man nicht alle Zahlen durchprobieren: Man kann sich auf Primzahlen beschränken und man muss auch nur die Zahlen von 2 bis $\sqrt{n}$ testen. Denn wenn $n = pq$ ist, so ist $p \leq \sqrt{n}$ oder $q \leq \sqrt{n}$. Das ist ein sicheres Verfahren, aber ein sehr, sehr ineffizientes. Zum Beispiel müsste man, um eine 200-stellige Zahl zu testen, im schlimmsten Fall alle Primzahlen zwischen 2 und 10^{100} durchprobieren. Das sind mehr als 10^{97} Primzahlen, weit mehr als die Anzahl der Atome im Universum. Kein Mensch kann das machen.

Die Mathematiker waren natürlich nicht faul und haben sich viel kompliziertere Sachen als diesen simplen Algorithmus einfallen lassen. Aber: Für unser Problem nützen alle diese schwierigen Algorithmen (fast) nichts.

Einige Beispiele mögen den Fortschritt in der Kunst, große Zahlen zu faktorisieren, illustrieren:

Am 15. Juni 1990 verkündeten Arjen K. Lenstra und Mark S. Manasse, dass sie die „neunte Fermat-Zahl" $F_9 = 2^{2^9} + 1 = 2^{512} + 1$ vollständig in ihre Primfaktoren zerlegen konnten!

Fünf Jahre später präsentierte Herman te Riele vom Center for Mathematics and Computer Science (CWI) „with algorithmic and sieving contributors Stefania Cavallar, Bruce Dodson, Arjen Lenstra, Walter Lioen, Peter L. Montgomery, Brian Murphy" stolz die Nachricht, dass sie am 22. August 1999 eine 512 Bit lange RSA-Zahl faktorisiert haben. RSA-Zahlen sind Zahlen, die Produkt von zwei Primzahlen sind, die die gleiche Größenordnung haben. RSA-Zahlen gelten als besonders schwer zu faktorisierende Zahlen.

Wieder etwa fünf Jahre später wurde ein neuer Weltrekord aufgestellt. Am 9. Mai 2005 veröffentlichte das Bundesamt für die Sicherheit in der Informationstechnik (BSI) in Bonn, dass es ihnen zusammen mit der Universität Bonn und dem CWI in Amsterdam gelungen ist, erstmals eine RSA-Zahl mit 200 (Dezimal-)Stellen in ihre Primfaktoren zu zerlegen.

Am 12. Dezember 2009 wurde der neueste Weltrekord vorgestellt. Das Team des vorigen Rekords ergänzte sich um weitere Forschungszentren: das schweizerische EPFL, das französische INRIA und das japanische NTT. Wissenschaftlern aus all diesen Instituten gelang es, in einer gemeinsamen Arbeit,

eine RSA-Zahl mit 768 Bits (das sind 232 Dezimalstellen) in ihre Primfaktoren zu zerlegen. Es handelte sich um die folgende Zahl:

RSA768 = 1.230.186.684.530.117.755.130.494.958.384.962.720.772
.853.569.595.334.792.197.322.452.151.726.400.507.263
.657.518.745.202.199.786.469.389.956.474.942.774.063
.845.925.192.557.326.303.453.731.548.268.507.917.026
.122.142.913.461.670.429.214.311.602.221.240.479.274
.737.794.080.665.351.419.597.459.856.902.143.413

Das Wissenschaftlerteam fand heraus, dass die Zahl RSA-768 das Produkt der folgenden beiden Primzahlen p und q ist:

p = 33.478.071.698.956.898.786.044.169.848.212.690.817.704.794
.983.713.768.568.912.431.388.982.883.793.878.002.287.614
.711.652.531.743.087.737.814.467.999.489
q = 36.746.043.666.799.590.428.244.633.799.627.952.632.279.158
.164.343.087.642.676.032.283.815.739.666.511.279.233.373
.417.143.396.810.270.092.798.736.308.917

Es ist keine Kunst, die Zahlen p und q zu multiplizieren. (Sie können leicht(!) überprüfen, dass die letzte Ziffer des Produkts stimmt, und fast ebenso leicht können Sie die vorletzte und vorvorletzte Ziffer verifizieren.) Die Herausforderung war die umgekehrt: Gegeben war RSA-768, gesucht waren p und q.

Die Wissenschaftler, die diesen Weltrekord aufgestellt haben, sind der Überzeugung, dass die Faktorisierung eines 1024 Bit langen RSA-Moduls „tausend mal schwieriger“ sein würde. Sie verweisen aber auch auf den großen Fortschritte beim Faktorisieren in den vergangenen Jahrzehnten und sehen das Ende von 1024 Bit langen RSA-Moduln n voraus.

Dies sind in der Tat großartige Erfolge, die auch in der Öffentlichkeit mit Interesse zur Kenntnis genommen werden. Bei jedem neuen Faktorisierungsweltrekord flammt eine neue und oft heftige Debatte über die Sicherheit des RSA-Algorithmus und verwandter Algorithmen auf. Mitarbeiter der Faktorisierungsfactory und Gegner des RSA-Algorithmus jubelten und sahen sich bestätigt, während Skeptiker und Befürworter des RSA-Algorithmus zu Recht zu bedenken gaben, dass niemand so „kleine“ Zahlen als Moduli im RSA-Algorithmus verwenden würde. Wir halten die folgenden Punkte fest.

- Kein Mensch hat bis jetzt einen effizienten (das heißt „polynomiellen“) Faktorisierungsalgorithmus gefunden. Darauf gründen manche Leute ihren Glauben, dass es keinen guten Faktorisierungsalgorithmus gebe. Mit den derzeitigen Methoden scheinen „allgemeine Zahlen“ mit etwa 300 Dezimalstellen unangreifbar zu sein.
- Kann man wenigstens beweisen, dass man keinen guten Algorithmus finden kann? Nein, auch das hat man bis jetzt nicht geschafft. Etwas überspitzt gesagt: Das Faktorisierungsproblem ist so schwer, dass man nicht einmal beweisen kann, wie es schwer ist!
- Die Antwort auf die Frage, wie viele Bit der Modul n haben muss, um sicher gegen eine Faktorisierungsattacke zu sein, ist nach wie vor eine Glaubensfrage. Die RSA-Anhänger argumentieren, dass bis jetzt nur wenige RSA-Zahlen zwischen 512 und 768 Bits faktorisiert worden seien und deswegen bis auf weiteres Moduli der Länge 1024 ohne Bedenken verwendet werden könnten. Demgegenüber weisen sowohl Faktorisierungsfreaks als auch sicherheitsbewusste Anwender auf den rasanten Fortschritt bei der Faktorisierung hin und meinen: Wenn schon RSA, dann bitte doch 2048 Bit. Dies ist auch die offizielle Empfehlung des BSI (www.bsi.de) für Anwendungen, die über den Tag hinaus Gültigkeit haben sollen.

Wir fassen zusammen: Eine Attacke auf den RSA wäre, die Zahl n zu faktorisieren. Darüber hinaus gilt: Bis jetzt ist keine andere erfolgversprechende Attacke bekannt! Das Faktorisierungsproblem gehört aber zu den schwierigsten Problemen der Mathematik. Dass man ernsthaft daran denkt, ein Kryptosystem einzuführen, dessen Stärke darauf basiert, dass es bis jetzt kein Mathematiker geschafft hat, einen vernünftigen Faktorisierungsalgorithmus anzugeben, ist zwar eine massive Beleidigung der Mathematik, die aber produktiv genutzt wird.

Zum Schluss noch eine Frage: Über den RSA-Algorithmus wird viel gesprochen und geschrieben, und er wird durchaus auch praktisch eingesetzt – aber man könnte sich fragen, warum er nicht noch viel häufiger Verwendung findet. Warum? Das liegt daran, dass man erst seit wenigen Jahren in der Lage ist, den RSA-Algorithmus vernünftig zu implementieren. Es gibt Softwareversionen, die für elektronische Unterschriften eingesetzt werden. Seit einigen Jahren sind auch RSA-Chips, sogar für Chipkarten, auf dem Markt. Diese schaffen es, eine RSA-Operation in wenigen Millisekunden durchzuführen. Wenn man aber daran denkt, dass man im kommerziellen Bereich (etwa in „lokalen Netzen“) mit 100 MBit/s (also 100.000.000 Bits pro Sekunde) und mehr arbeitet, so erkennt man, dass der RSA-Algorithmus noch lange nicht, vielleicht überhaupt nie, für Verschlüsselungszwecke, sondern hauptsächlich für das Schlüsselmanagement und für die elektronische Signatur eingesetzt

werden wird. Bei der Bildung einer elektronischen Unterschrift wird man nämlich sinnvollerweise den Text zunächst mittels einer Hashfunktion komprimieren und den Signaturalgorithmus nur auf das Komprimat anwenden. So schlägt man zwei Fliegen mit einer Klappe: Zum einen muss nur eine sehr kleine Datenmenge signiert werden, zum andern hat die Signatur immer eine feste, kurze Länge.

Eine *Hashfunktion* ist eine Funktion, die Nachrichten beliebiger Länge auf Nachrichten einer festen Länge k komprimiert und die *kollisionsresistent* ist. Das soll bedeuten, dass es sehr schwer sein soll, zwei Nachrichten zu finden, die auf den gleichen Wert komprimiert werden. Ein typischer Wert für k ist 256 Bit. Es scheint außerordentlich schwierig zu sein, kollisionsresistente Hashfunktionen zu finden. Die meisten Vorschläge für Hashfunktionen wurden mehr oder weniger schnell geknackt, Empfohlen werden heute Verfahren aus der SHA-Familie (SHA = Secure Hash Algorithm).

Algorithmus 5.8: Signatur mit Hilfe einer Hashfunktion

Ein Signaturschema kann zusammen mit einer Hashfunktion benutzt werden.

Erstellen der Signatur:	Um eine Nachricht m zu signieren, berechnet A zunächst den Hashwert h(m) und signiert nur diesen, indem sie auf diesen ihren privaten Schlüssel D anwendet: sig = D(h(m)). Veröffentlicht wird sowohl sig als auch m.
Verifizieren der Signatur:	Eine von A erstellte Signatur wird verifiziert, indem zunächst der Hashwert h(m) berechnet wird und dann überprüft wird, ob h(m), sig und der öffentliche Schlüssel E von A „zusammenpassen“.

5.4 Schlüsselaustausch

Alle bislang bekannten asymmetrischen Algorithmen sind so langsam, dass es unvorstellbar ist, große Datenmengen in akzeptabler Zeit zu chiffrieren. Das liegt daran, dass die Algorithmen auf dem exakten Rechnen mit sehr großen Zahlen beruhen – und das kostet Zeit. Auf der anderen Seite gibt es gute und sehr schnelle symmetrische Algorithmen. Daher ist man schon frühzeitig auf die Idee gekommen, die Vorteile der beiden Systeme zu kombinieren. Die grundlegende Idee eines sogenannten *hybriden Systems* ist die folgende.

Im Wesentlichen wird zum Verschlüsseln ein symmetrischer Algorithmus benutzt. Man verwendet ein asymmetrisches Verschlüsselungssystem nur für den Austausch der Schlüssel. Da die auszutauschenden Schlüssel Nachrichten

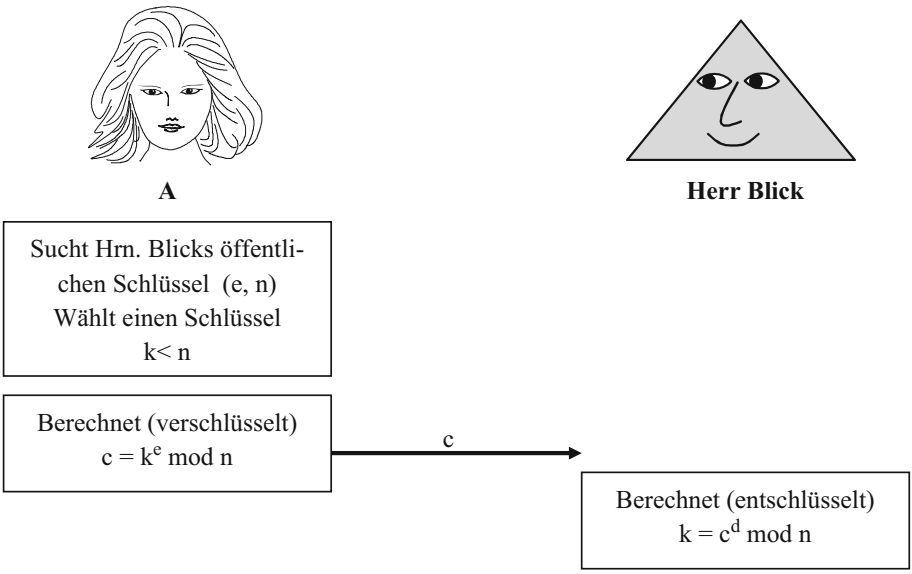


Abb. 5.6 RSA-Schlüsselvereinbarung

von äußerst geringer Länge sind (typische Werte sind 56 oder 128 Bits), ist die schlechte Performance des asymmetrischen Verfahrens kein Flaschenhals für das Gesamtsystem. Außerdem werden Schlüssel relativ selten gewechselt. Ein Schlüsselwechsel bei jeder Sitzung wird, obwohl dies in manchen Anwendungen wünschenswert wäre, in der Praxis nur sehr selten realisiert.

Um einen geheimen Schlüssel k an Herrn Blick zu schicken, verschlüsselt man k mit Hilfe von Herrn Blicks öffentlichem Schlüssel. Dann kann Herr Blick den erhaltenen Wert c mit seinem geheimen Schlüssel entschlüsseln und erhält k (vergleiche Abb. 5.6).

Algorithmus 5.9: Hybride Verschlüsselung

Verschlüsselung: Um eine Nachricht m verschlüsselt an B zu schicken, wählt A zunächst einen Schlüssel k für einen symmetrischen Algorithmus f und bestimmt den Geheimtext $c_1 = f_k(m)$. Dann verschlüsselt A mit Hilfe des öffentlichen Schlüssels von B den gewählten Schlüssel; sie erhält $c_2 = E_B(k)$. Sie sendet sowohl c_1 als auch c_2 an B.

Entschlüsselung: B dechiffriert zunächst c_2 mit Hilfe seines privaten Schlüssels und erhält $D_B(c_2) = k$. Damit kann er nun die eigentliche Nachricht entschlüsseln: $f_k^{-1}(c_1) = m$.

Diese Methode ist vollkommen; sie hat nur einen einzigen, winzigen, „philosophischen“ Nachteil: A wählt den Schlüssel und schickt ihn an B, um B die Kommunikation zu ermöglichen. Von einem höheren Standpunkt aus könnte man fordern, dass beide Partner einen Schlüssel vereinbaren sollten und nicht einer den Schlüssel auswählen sollte, um ihn anderen zu schicken. Bei einer idealen Schlüsselvereinbarung sollten beide Partner die gleiche Rolle spielen.

Überraschenderweise gibt es ein solches System. Noch überraschender: Es wurde in der allerersten Arbeit der Public-Key-Kryptographie vorgeschlagen, in der berühmten „New Directions“-Arbeit von Diffie und Hellman [DH76]. Diffie [Dif88] schreibt:

> Marty und ich spielten in unseren Gedanken und Diskussionen immer mit Exponenten herum und versuchten, sie hinzukriegen. An einem frühen Morgen im Mai 1976 gelang Marty schließlich der Durchbruch. Marty rief mich an und erklärte mir den exponentiellen Schlüsselaustausch in seiner herausfordernden Einfachheit. Beim Zuhören merkte ich, dass mir die Idee schon lange halb bewusst gewesen, aber niemals zum Durchbruch gekommen war.

Dieser exponentielle Schlüsselaustausch heißt heute *Diffie-Hellman-Schlüsselaustausch*. Wir beschreiben dieses System nun zuerst mathematisch und dann mit Begriffen aus jedermanns Alltag.

Die Partner, die einen Schlüssel vereinbaren wollen, müssen sich schon vorab über zwei Dinge einig sein: Eine Primzahl p und eine natürliche Zahl $g < p$. Diese Zahlen bergen kein Geheimnis, sie können völlig öffentlich sein; beispielsweise könnten alle Teilnehmer dasselbe p und g haben.

Das Schema ist ganz einfach: A und B wählen sich je eine natürliche Zahl a bzw. b mit $a, b < p - 1$, die sie geheim halten. A berechnet

$$\alpha = g^a \bmod p,$$

und B berechnet

$$\beta = g^b \bmod p.$$

Dann schickt jeder sein Ergebnis dem anderen. Schließlich bildet A die Zahl

$$k = \beta^a \bmod p,$$

und B berechnet

$$\alpha^b \bmod p.$$

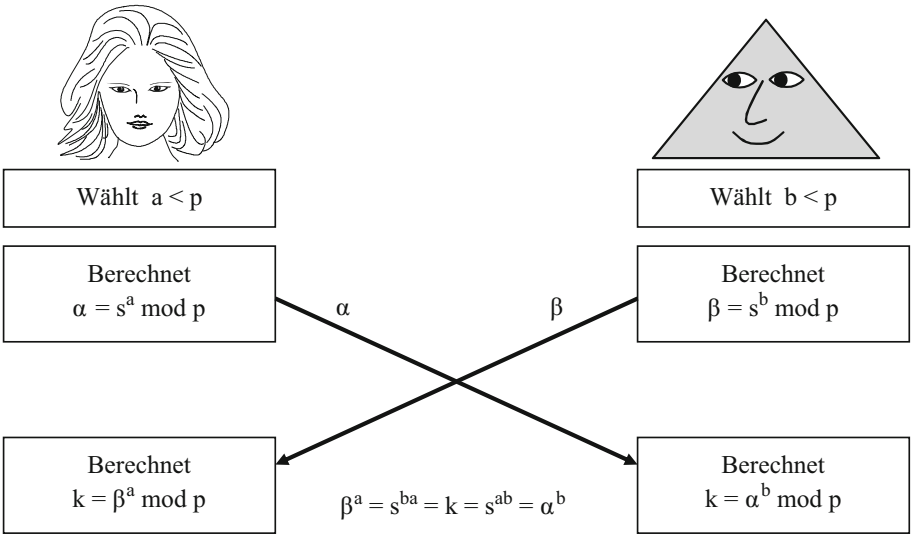


Abb. 5.7 Diffie-Hellman-Schlüsselaustausch

Da

$$\alpha^b \bmod p = (g^a)^b \bmod p = g^{ab} \bmod p$$

und

$$\beta^a \bmod p = (g^b)^a \bmod p = g^{ba} \bmod p = s^{ab} \bmod p$$

ist, erhalten beide denselben Wert $k = g^{ab} \bmod p$. Abbildung 5.7 macht deutlich, dass beide Seiten die gleiche Rolle spielen.

Die Partner A und B können die Zahl k direkt als Schlüssel verwenden oder sie wenden eine öffentliche Funktion auf k an, um den Schlüssel für das symmetrische Verschlüsselungsverfahren zu erhalten. Beispielsweise könnten sie die binäre Darstellung von k wählen (oder eine Auswahl von Bits davon).

Algorithmus 5.10: Diffie-Hellman-Schlüsselaustausch

Öffentliche Daten: Eine Primzahl p und eine natürliche Zahl $g \leq p$.

Zwei Personen A und B vereinbaren einen gemeinsamen Schlüssel, indem sie jeweils eine Zahl a bzw. b geheim wählen, die Werte $\alpha = g^a \bmod p$ bzw. $\beta = g^b \bmod p$ berechnen, die Ergebnisse dem Partner zukommen lassen und schließlich $\beta^a \bmod p$ bzw. $\alpha^b \bmod p$ berechnen Diese Werte sind identisch und können als gemeinsamer Schlüssel benutzt werden.

Frage: Warum ist dieses System sicher gegen einen Angriff von Mr. X? Mr. X kennt α und β. Er könnte von α auf a oder von β auf b schließen, um dann den Schlüssel k genauso bequem zu berechnen wie A oder B.

Es ist aber außerordentlich schwer, von $\alpha = g^a$ *auf* a *zu schließen.* Das Problem, aus gegebenem g^a mod p die Zahl a zu berechnen, heißt das *Problem des diskreten Logarithmus* (zur *Basis* g), ein Problem, das ähnlich schwierig ist wie das Problem der Faktorisierung (manche Experten meinen: sogar noch schwieriger).

Ein Beispiel macht die Schwierigkeit des Problems augenscheinlich. Angenommen, wir haben $p = 11$ und $g = 4$. Die Berechnung von Potenzen ist einfach:

$$g^4 \bmod 11 = 4^4 \bmod 11 = 4^2 \cdot 4^2 \bmod 11 = 5 \cdot 5 \bmod 11 = 3.$$

Aber, falls Mr. X nur weiß, dass $4^a \bmod 11 = 3$ gilt, ist es für ihn sehr schwierig herauszufinden, was a ist. Sie sollten diese vergleichsweise einfache Aufgabe lösen und sich dann an Übungsaufgabe 19 versuchen.

Ein Punkt könnte allerdings noch zugunsten von Mr. X sprechen: Er muss vielleicht nicht notwendigerweise eine der Zahlen a oder b finden. Da er sowohl α als auch β kennt, ist es vorstellbar, dass Mr. X (dem wir ja unglaubliche Intelligenz zutrauen) einen Weg findet, die Zahl k direkt aus α und β abzuleiten. Das Dumme (für Mr. X) ist nur, dass bislang niemand einen solchen Weg gefunden hat. Aber: Auch niemand hat bewiesen, dass das Knacken des Diffie-Hellman-Schlüsselaustauschprotokolls gleichwertig ist zum Berechnen des diskreten Logarithmus. Wenn man die Sicherheit des RSA-Algorithmus mit einer 200-stelligen Zahl n erreichen will, so muss man hier eine 200-stellige Primzahl p wählen.

Kurz: die Sicherheit des Diffie-Hellman-Schlüsselaustauschs beruht wesentlich darauf, dass das Potenzieren modulo p eine Einwegfunktion ist.

Abbildung 5.8 zeigt eine Visualisierung des Diffie-Hellman-Schlüsselaustauschs. Diese Darstellung ist wie folgt zu interpretieren. Beide Seiten erhalten identische Kopien eines Koffers (der den gemeinsamen Schlüssel k enthält); diese Koffer sind jeweils mit zwei Schlössern verschlossen, von denen jeweils eines nur von A, das andere nur von B geöffnet werden kann. Zunächst entfernen beide Seiten die Schlösser, die sie öffnen können und schicken den Koffer zur anderen Seite. Nun erhalten beide einen Koffer, den sie öffnen können; sie tun das und können das Geheimnis k des Koffers lüften.

Das Diffie-Hellman-Schlüsselaustauschprotokoll ist nicht nur möglich über Alphabeten mit p Elementen, sondern auch über Alphabeten mit p^m Elementen, wobei p eine Primzahl ist. Im Falle $m > 1$ benutzt man zu diesem Zweck nicht die natürlichen Zahlen modulo p^m, sondern den endlichen

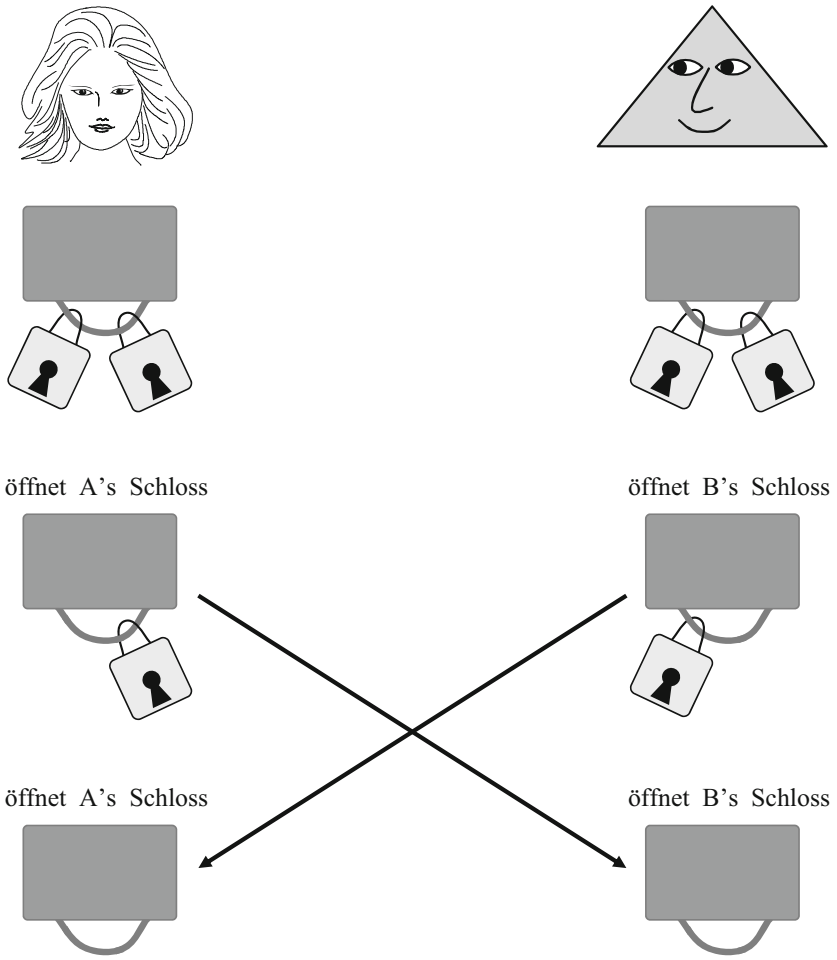


Abb. 5.8 Diffie-Hellman-Schlüsselaustausch mit Koffern

Körper mit p^m Elementen, der oft mit $GF(p^m)$ bezeichnet wird (Galoisfeld, nach dem genialen, aber tragisch bei einem Duell umgekommenen französischen Mathematiker Evariste Galois (1811–1832)). Als besonders erfolgreich hat sich erwiesen, das Problem des diskreten Logarithmus insbesondere in so genannten *elliptischen Kurven* für kryptographische Zwecke zu studieren. Elliptische Kurven sind keineswegs Ellipsen, sondern Kurven 3. Grades der Form $y^2 = x^3 + ax + b$, also etwa $y^2 = x^3 - x + 1$. Überraschenderweise kann man die Punkte einer solchen Kurve als „Zahlen höherer Art" ansehen, mit denen man vernünftig rechnen kann. Es ist möglich, alle Protokolle, die auf dem diskreten Logarithmus modulo p beruhen, auf elliptische Kurven zu übertragen. Kryptologisch betrachtet scheinen die elliptischen Kurven viel

sicherer zu sein als der diskrete Logarithmus modulo p. Das bedeutet, dass man für das gleiche Sicherheitsniveau mit wesentlich kleineren Schlüssellängen auskommt. Die Behandlung elliptischer Kurven würde aber weit über das Ziel dieses Buches hinausführen; der interessierte Leser sei zum Beispiel auf [MOV96] verwiesen.

5.5 Weitere Anwendungen des diskreten Logarithmus

Interessanterweise hat die Schwierigkeit, diskrete Logarithmen zu berechnen, nicht nur Anwendungen für das Schlüsselmanagement, sondern kann auch zur Konstruktion von Verschlüsselungs- und Signaturschemata benutzt werden. Man kennt verschiedene Arten der Verwendung des diskreten Logarithmus; eine Methode geht auf Taher ElGamal vom Hewlett-Packard Forschungslaboratorium in Palo Alto zurück [EIG85], eine andere auf Jim Massey von der ETH Zürich und Jim Omura, der damals an der University of California war. Das ElGamal-Verschlüsselungsschema kann als Variante des Diffie-Hellman-Schemas mit eingebautem Verschlüsselungsalgorithmus aufgefasst werden. Wir werden zunächst kurz dieses Protokoll diskutieren und dann das Massey-Omura-Schema vorstellen, das prinzipiell hübsch einfach ist.

Für das ElGamal-Schema müssen die Teilnehmer sich über eine Primzahl p und eine natürliche Zahl g einig sein. Jeder Teilnehmer B hat als seinen geheimen Schlüssel eine natürliche Zahl b und als seinen öffentlichen Schlüssel die Zahl $\beta = g^b \bmod p$.

Falls A die Nachricht m verschlüsselt an B schicken will, so wählt sich A zunächst eine natürliche Zahl a und berechnet damit zwei Zahlen: Zuerst benutzt sie Bs öffentlichen Schlüssel und berechnet $k = \beta^a \bmod p$. Dann benutzt A diese Zahl als Kommunikationsschlüssel für die Nachricht m: A wendet einen Verschlüsselungsalgorithmus f unter dem Schlüssel k auf m an und erhält den Geheimtext $c = f_k(m)$. Schließlich schickt sie zwei Werte an B, nämlich $\alpha = g^a \bmod p$ und den Geheimtext c.

Mit seinem geheimen Schlüssel b kann B aus α den Schlüssel $k = \alpha^b \bmod p$ berechnen und damit c dechiffrieren.

Im ursprünglichen ElGamal-Schema war der Algorithmus f einfach die Multiplikation mod p, also $c = k \cdot m \bmod p$ (eine Art multiplikatives One-Time-Pad), aber dies ist eine unnötig starke Einschränkung.

Algorithmus 5.11: ElGamal-Verschlüsselung

Chiffrieren: Um eine Nachricht m verschlüsselt an B zu schicken, berechnet A zunächst $k = \beta^a \bmod p$, wobei β der öffentliche Schlüssel von B und a eine von A zufällig gewählte Zahl ist. Dann wird die Nachricht m verschlüsselt: $c = f_k(m)$. A sendet sowohl c als auch $\alpha = g^a \bmod p$ an B.

Dechiffrieren: B wendet zunächst seinen privaten Schlüssel auf α an und erhält $k = \alpha^b \bmod p$. Damit entschlüsselt er den Geheimtext c: $m = f_k^{-1}(c)$.

Es gibt auch ein ElGamal-Signaturschema; dies hat dadurch besondere Bedeutung erlangt, dass das amerikanische *National Institute of Standards and Technology* im Sommer 1991 eine Variante dieses Schemas zur Standardisierung vorgeschlagen hat [NIST91].

Wir kommen nun zum Massey-Omura-Schema. Die zugrundeliegende Idee ist ganz einfach; man nennt sie auch *Shamir's no-key-Algorithmus* (siehe Abb. 5.9).

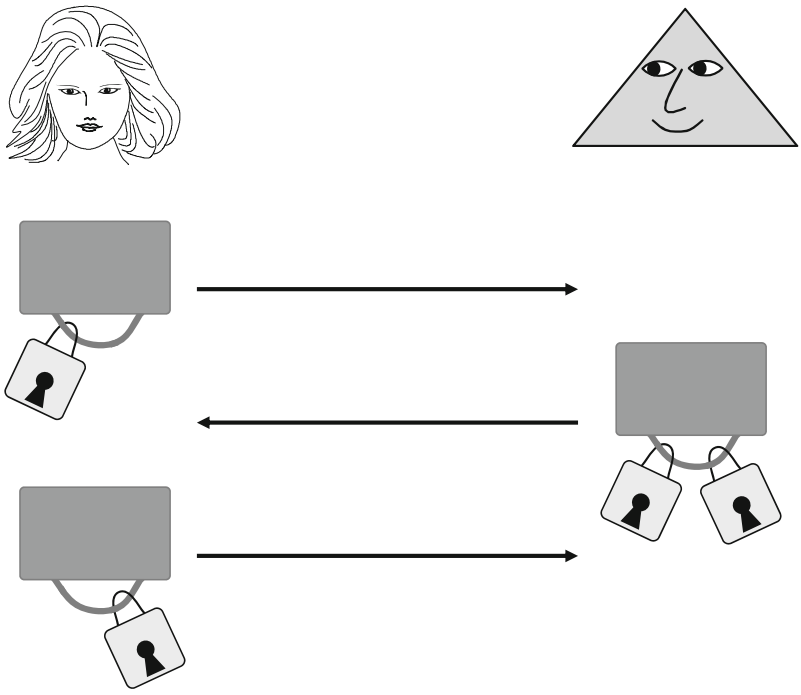


Abb. 5.9 Shamir's no-key-Algorithmus

Wenn A eine Nachricht verschlüsselt an B schicken möchte, gibt sie die Nachricht in einen Koffer, verschließt diesen mit ihrem Schloss und schickt das ganze zu B. Dieser kann natürlich den Koffer nicht öffnen, aber er kann den Koffer ein zweites Mal verschließen, und zwar mit seinem Schloss. Anschließend schickt er diese doppelt gesicherte Nachricht zurück zu A. Diese kann nun zwar ihr Schloss entfernen, aber die Nachricht trotzdem nicht mehr lesen. Wird der Koffer nun wieder an B geschickt, so kann dieser ihn bequem öffnen und die Nachricht zur Kenntnis nehmen.

Kann man diese Idee benutzen. um die Übertragung elektronischer Daten zu sichern? Die Antwort lautet: Ja – und zwar mit Hilfe diskreter Logarithmen. Wir stellen uns vor, dass alle Teilnehmer sich über eine Primzahl p einig sind. Jeder Teilnehmer T wählt sich zwei natürliche Zahlen e_T und d_T mit der Eigenschaft, dass

$$e_T \cdot d_T \bmod p - 1 = 1$$

gilt. In Abschn. 5.3.2.2 wurde beschrieben, wie man d_T aus e_T berechnen kann. Wir wissen ferner (siehe Abschn. 5.3.1), dass aufgrund des Satzes von Euler für alle natürlichen Zahlen $m < p$ die Gleichung

$$m^{eTdT} \bmod p = m$$

gilt.

Im Gegensatz zum RSA-Algorithmus behalten die Teilnehmer beide ihre Zahlen geheim und veröffentlichen keine. Das ist absolut notwendig; denn wenn Mr. X etwa e_T wüsste, könnte er daraus leicht d_T berechnen, da der Modul p öffentlich bekannt ist. Wir stellen uns vor, dass A eine Nachricht m verschlüsselt an B senden möchte. Wir können annehmen, dass m durch eine Zahl (oder eine Folge von Zahlen) kleiner p dargestellt ist.

Zunächst berechnet A die Zahl $m^{eA} \bmod p$ und schickt diese an B; B berechnet dann die e_B-te Potenz der erhaltenen Zahl und schickt das Ergebnis $m^{eAeB} \bmod p$ zurück zu A. Nun wendet A ihre Zahl d_A auf die erhaltene geheimnisvolle Zahl an und erhält $m^{eAeBdA} \bmod p$, was sich nach kurzem Umrühren zu

$$m^{eAeBdA} \bmod p = m^{eAdAeB} \bmod p = m^{eB} \bmod p,$$

vereinfacht. Das Ergebnis wird an B geschickt, und dieser erhält ohne Mühe (vergleiche Abb. 5.10)

$$m^{eBdB} \bmod p = m.$$

Inwiefern basiert dieses Protokoll auf der Schwierigkeit, diskrete Logarithmen zu berechnen? Wie könnte Mr. X die Geheimnisse e_B und d_B erlangen?

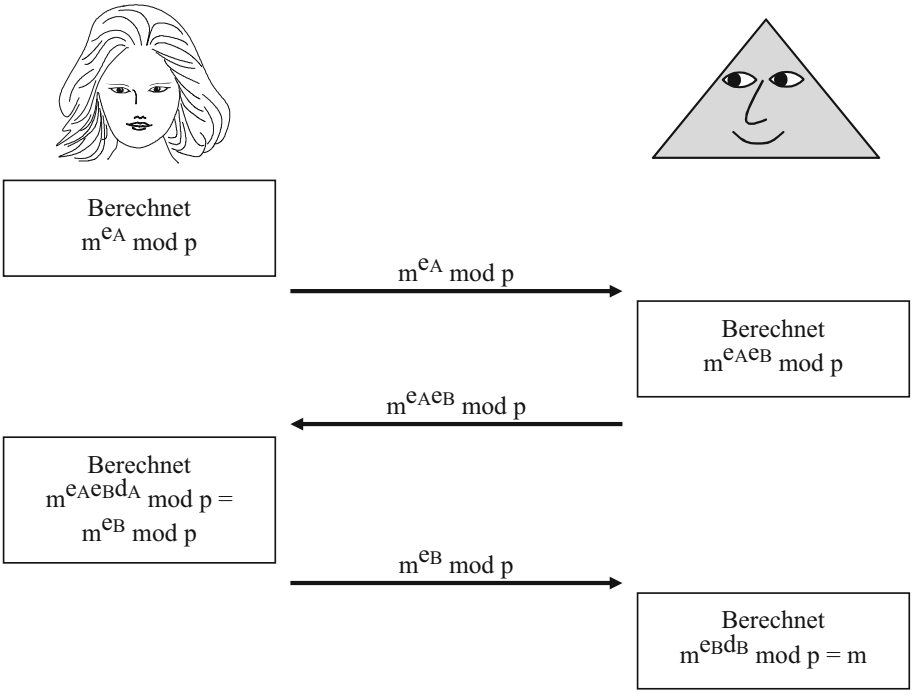


Abb. 5.10 Das Massey-Omura-Protokoll

Ganz einfach – falls er diskrete Logarithmen berechnen kann. Er müsste nämlich nur an B eine beliebige Zahl m schicken; B würde diese gemäß dem Protokoll behandeln und also die Zahl

$$m' = m^{e_B} \bmod p$$

zurückschicken. Dann würde Mr. X den diskreten Logarithmus von m' berechnen und hätte e_B erhalten; mit dem euklidischen Algorithmus könnte er daraus auch d_B berechnen.

Es sind zwei weitere asymmetrische Verschlüsselungsschemata bekannt, das Schema von R.J. McEliece [McE78] und der Knapsack-Algorithmus, der von Merkle und Hellman [MH78] vorgeschlagen wurde. Dieser Algorithmus hat allerdings einen zweifelhaften Ruf, da er von Shamir [Sha82] geknackt wurde.

5.6 Die Authentizität der öffentlichen Schlüssel

Sowohl der Schlüsselaustausch mit Hilfe eines asymmetrischen Verschlüsselungsverfahrens als auch der Diffie-Hellman-Schlüsselaustausch haben den

unbestreitbar großen Vorteil, dass man damit zum Schlüsselaustausch keinen geheimen Kanal mehr braucht. Aber ohne *Vertrauen* geht es auch hier nicht. Denn die Verfahren funktionieren nur dann gut, wenn keiner der Beteiligten betrügen kann. Jeder, der einen öffentlichen Schlüssel benutzt, muss sicher sein, dass dieser wirklich der öffentlich Schlüssel der angegebenen Person ist – und nicht etwa der von Mr. X, den dieser untergeschoben hat. Mit anderen Worten: Das kryptographische System muss einen unverbrüchlichen Zusammenhalt zwischen Namen und öffentlichem Schlüssel garantieren.

Vertrauen ist gut, Kontrolle ist besser! Kann mir irgendjemand garantieren, dass der öffentliche Schlüssel, den ich benutze, authentisch ist, das heißt wirklich zu dem Partner gehört, dem ich in diesem Moment eine geheime Nachricht zukommen lassen möchte? Die Antwort darauf ist „ja“, und das Schöne ist, dass sich die Public-Key-Kryptographie gewissermaßen an den eigenen Haaren aus dem Sumpf zieht.

Das Problem können Sender und Empfänger im Normalfall nicht alleine lösen, denn sie haben ja normalerweise keinen direkten Kontakt miteinander. Man braucht eine dritte, vertrauenswürdige Instanz, die üblicherweise CA (*Certification Authority*) genannt wird. Diese überzeugt sich auf direkte Weise, dass der öffentliche Schlüssel eines Teilnehmers authentisch ist und signiert dann mit ihrem privaten Schlüssel das Paar bestehend aus dem Namen und dem öffentlichen Schlüssel des Teilnehmers. Man bezeichnet Name, öffentlicher Schlüssel und Signatur der CA insgesamt als *Zertifikat*.

Jeder, der einen öffentlichen Schlüssel benutzt, muss in seinem eigenen Interesse zuvor das Zertifikat verifizieren. Dazu benützt er den öffentlichen Schlüssel der CA. Beim Verschlüsseln prüft der Sender das Zertifikat des Empfängers, bei einer Signaturanwendung prüft der Verfizierer das Zertifikat dessen, der die Signatur – angeblich – geleistet hat.

Algorithmus 5.12: Zertifikaterstellung und Zertifikatüberprüfung

Erstellen des Zertifikats:	Die Certification Authority überprüft in direkter Weise die Authentizität des öffentlichen Schlüssel, d. h. ob ein vorgelegter öffentlicher Schlüssel wirklich zu dem angegebenen Teilnehmer gehört. Dann signiert sie das Paar (Name, öffentlicher Schlüssel). Das Zertifikat besteht (im Wesentlichen) aus Name, öffentlichem Schlüssel und Signatur der CA.
Verifizieren des Zertifikats:	Bevor ein öffentlicher Schlüssels eines Teilnehmers verwendet wird, muss seine Authentizität (mit Hilfe des öffentlichen Schlüssels der CA) überprüft werden.

Man kann sich das Verfahren gut an dem Briefkastenbeispiel aus Abschn. 5.1 klarmachen: Die Signatur der CA ist gewissermaßen ein Stempel, der das Namensschild unauflöslich mit dem Briefkasten, d. h. dem öffentlichen Schlüssel verbindet. Man wird nur einen Brief in den Briefkasten werfen, wenn man sich überzeugt hat, dass der Stempel unversehrt ist.

Der Zertifikatsdienst wurde erstmals in dem Standard X.509 [CCITT] ausführlich vorgestellt; er ist heute ein zentraler Bestandteil jeder *Public-Key-Infrastruktur* (PKI).

Zum Schluss dieses langen Kapitel erwähnen wir kurz das Programm PGP (Pretty Good Privacy). In ihm sind viele der in diesem Kapitel erwähnten Mechanismen und Algorithmen realisiert: PGP verwendet zur Verschlüsselung ein hybrides System (vgl. Abschn. 5.4) auf Basis des RSA-Algorithmus. Die Authentizität der öffentlichen Schlüssel wird durch einen trickreichen Zertifikatsmechanismus gewährleistet. (Siehe z. B. [MOV96]).

5.7 Seitenkanalangriffe

In den letzten Jahren wurden Angriffe auf real existierende Krypto-Verfahren gefunden, die ausgesprochen gemein sind. Diese so genannten Seitenkanalangriffe nutzen Informationen aus, die häufig „einfach so“ anfallen, mit denen man aber gewissermaßen „mit einem Trick“ ein ganzes Verfahren aushebeln kann.

Ein klassischer Seitenkanal ist beziehungsweise waren die Klingeltöne beim Telefonieren. Man rief einfach jemanden an und legte nach dreimaligem Klingeln wieder auf. Das bedeutete dann „ich komme“. So konnte man Information übermitteln, ohne den Netzbetreiber dafür zu bezahlen.

Wir schildern hier nur einen typischen Seitenkanalangriff auf den RSA-Algorithmus. Dieser beruht auf zwei Voraussetzungen: 1. Die Potenzierung geschieht mit dem Square-and-Multiply-Algorithmus (siehe Abschn. 5.3.5). Standardmäßig wird dieser so implementiert, dass der Exponent in binärer Form dargestellt wird. Dann wird bei jedem Bit des Exponenten quadriert, und wenn das Bit gleich Eins ist, erfolgt zusätzlich eine Multiplikation. 2. Der Angreifer hat Zugriff auf das Kryptogerät, und zwar in dem Sinne, dass er den Stromverbrauch genau messen kann.

Damit kann der Angreifer herausbekommen, an welcher Stelle der Exponent eine Eins, und wo er eine Null hat. Wenn er so zum Beispiel die Operation c^d (Entschlüsselung) verfolgt, kann er den geheimen Schlüssel d Bit für Bit mitlesen. Ich finde an diesem Angriff besonders gemein, dass er gerade die mathematisch schöne und klare Struktur der Potenzierung ausnutzt.

Gegen diesen Angriff hilft nicht die Vergrößerung der Schlüssellänge; auch ein 100.000 Bit-RSA könnte so geknackt werden. Man kann dagegen nur so vorgehen, dass man an der Ursache ansetzt: Man könnte das Stromablesen durch einen Hardwareschutz erschweren oder eine andere Variante der Potenzierung verwenden.

5.8 Übungsaufgaben

1 Schulen Sie Ihr Vorstellungsvermögen. Interpretieren Sie das Arrangement von Abb. 5.11 als asymmetrisches Verschlüsselungsschema: Jeder Teilnehmer stellt einen Koffer mit seinem Namen zur Verfügung. Die geöffneten Koffer sind öffentlich zugänglich. Sie können geschlossen werden, indem man sie einfach zudrückt, aber für das Öffnen braucht man einen Schlüssel – den des rechtmäßigen Eigentümers.

2 Wenden Sie eine Transpositionschiffre auf den Namen Peter A. Schlaube an; Sie erhalten einen Namen, den Sie bestimmt schon mal gelesen haben.

3 (a) Sei z eine natürliche Zahl in Dezimaldarstellung. Zeigen Sie, dass z^5 die gleiche Einerziffer hat wie z.
(b) Berechnen Sie im Kopf die fünfte Wurzel aus 69.343.957.

4 Seien a und b nichtnegative ganze Zahlen.
(a) Seien q und r ganze Zahlen mit $a = bq + r$. Zeigen Sie: $ggT(a, b) = ggT(b, r)$. [Tipp: Überlegen Sie sich einerseits, dass jede Zahl, die a und b teilt, auch r teilt, und andererseits, dass jede Zahl, die b und r teilt auch ein Teiler von a ist.]
(b) Zeigen Sie, dass der euklidische Algorithmus wirklich den größten gemeinsamen Teiler der Zahlen a und b liefert.

5 @Schreiben Sie ein Programm zum euklidischen Algorithmus.

6 Bestimmen Sie ganze Zahlen x und y mit der Eigenschaft $3 = 792x + 75y$.

7 Berechnen sie ganze Zahlen x und y mit der Eigenschaft $1 = 17x + 55y$.

8 @Schreiben Sie ein Programm, das bei gegebenen natürlichen Zahlen a und b eine Vielfachsummendarstellung des größten gemeinsamen Teilers von a und b liefert.

9 Berechnen Sie den größten gemeinsamen Teiler aus der Nummer Ihres Personalausweises und Ihrer Telefonnummer.

10 Berechnen Sie für $p = 23$ und $q = 37$ Zahlen d und e gemäß dem RSA-Algorithmus und verschlüsseln Sie damit die Nachricht $m = 537$.

11 (a) Wie viele Multiplikationen braucht man, um m^{16} auszurechnen? Wie viele Multiplikationen braucht man, um m^{32}, m^{64}, … zu berechnen?
(b) Wie viele Multiplikationen braucht man, um m^{21} auszurechnen?

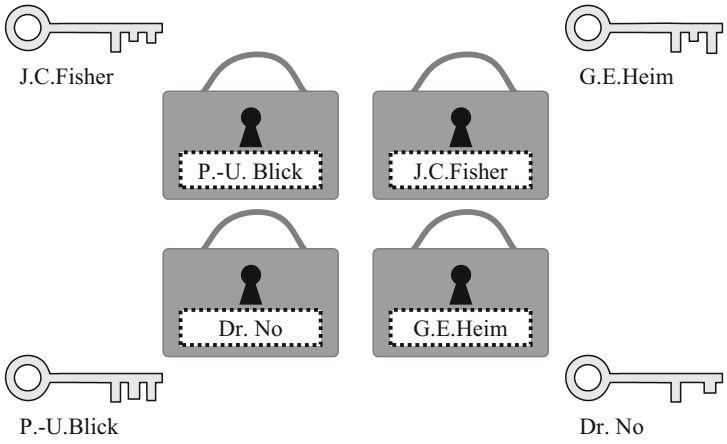


Abb. 5.11 Das Koffer-Beispiel

(c) Wie viele Multiplikationen braucht man, um mit 341 (= 256 + 64 + 16 + 4 + 1) zu potenzieren?
(d) Wie viele Multiplikationen braucht man, um mit der vierten Fermat-Zahl $F_4 = 2^{16} + 1 = 65.537$ zu potenzieren?
(e) Wie viele Multiplikationen braucht man, um mit 2^{1000} zu potenzieren?

In den folgenden drei Übungsaufgaben leiten wir den Leser zu einem Beweis des kleinen Satzes von Fermat an (vergleiche [Hon73] für Details).

12 Sei p eine Primzahl. Zeigen Sie, dass der Binomialkoeffizient

$$\binom{p}{i} = \frac{p \cdot (p-1) \cdot \ldots \cdot (p-i+1)}{i \cdot (i-1) \cdot \ldots \cdot 2 \cdot 1}$$

eine ganze Zahl ist, die im Falle $1 < i < p$ durch p teilbar ist.

13 Sei p eine Primzahl. Dann gilt für je zwei ganze Zahlen a und b

$$(a + b)^p \bmod p = a^p + b^p \bmod p.$$

[Hinweis: Benutzen Sie den binomischen Lehrsatz und die vorige Aufgabe.]

14 Sei p eine Primzahl. Dann gilt für jede ganze Zahl m der kleine Satz von Fermat:

$$(m^p - m) \bmod p = 0.$$

[Hinweis: Führen Sie Induktion nach m durch und benutzen Sie die vorige Aufgabe.]

15 (a) Zeigen Sie: Für jede natürlich Zahl m gilt $m^{13} \bmod 10 = m \bmod 10$.
(b) Die Zahlen m und m^{13} haben im Dezimalsystem die gleichen Endziffern
(c) Bestimmen Sie im Kopf die 13. Wurzel aus 96.889.010.407.
(d) Für welche Hochzahlen funktionieren ähnliche Tricks?

16 Ich verrate Ihnen zwei Dinge:
(i) Die Zahl 14.803 ist Produkt von genau zwei Primzahlen,
(ii) $\varphi(14.803) = 14.560$.
Können Sie mit Hilfe dieser Information die Zahl 14.803 faktorisieren ?

17 @Schreiben Sie ein Programm zur Faktorisierung von natürlichen Zahlen, das 8-stellige Zahlen faktorisieren kann. Überprüfen Sie, wie lange Ihr Programm braucht, um 12.863.273 ($= 3259 \cdot 3947$) und 1.111.111 ($= 239 \cdot 4649$) zu faktorisieren.

18 Angenommen, Sie kennen die Gleichung $7^x \bmod 31 = 10$. Bestimmen Sie x.

19 Stellen sie das EIGamal-Verschlüsselungsschema mit Hilfe eines „Pfeildiagramms“ dar.

20 Erklären Sie genau den Unterschied zwischen dem Diffie-Hellman-Schema und dem ElGamal-Schema. Ist der Unterschied kryptographischer Natur?

21 Hat das Massey-Omura-Protokoll Nachteile? Welche?

22 Wie viele Stellen (ungefähr) hat eine Dezimalzahl, die in binärer Darstellung 2048 Bit hat? [Hinweis: Wie viele Dezimalstellen hat eine 10 Bit-Zahl?]

23 Lesen Sie [CBZ99] und identifizieren Sie die in PGP verwendeten kryptographischen Mechanismen.

6

Ach wie gut, dass niemand weiß, dass ich Rumpelstilzchen heiß oder Wie bleibe ich anonym?

6.1 Was ist Anonymität?

Kryptographie hat es per definitionem mit Verheimlichung zu tun. Verschlüsselt man eine Nachricht, so wird der *Inhalt* dieser Nachricht verborgen. Hingegen wird die *Tatsache, dass* eine Nachricht gesendet wurde, nicht verheimlicht, und ebenso sind *Sender* und *Empfänger* nicht verborgen. Man erreicht so nur ein äußerst bescheidenes Maß an *Anonymität*. Wir können drei Arten von Anonymität unterscheiden:

- Anonymität des *Senders*,
- Anonymität des *Empfängers*,
- Anonymität der *Kommunikationsbeziehung*.

Vielleicht werden Sie einwenden: „Anonymität ist doch reiner Luxus! Ich will gerne zugeben, dass man Anonymität theoretisch machen kann; aber jeder – jedenfalls jeder, der nichts zu verbergen hat – wird doch spätestens dann, wenn er merkt, dass Anonymität Geld kostet, darauf verzichten." Ein solcher Einwand kann nicht einfach vom Tisch gewischt werden. Lassen Sie uns deshalb darüber nachdenken, ob beziehungsweise wann Anonymität notwendig, sinnvoll, überflüssig oder schädlich ist.

Ein Grundproblem der modernen Demokratie ist der Ausgleich zwischen den Rechten der einzelnen Bürger und den Rechten des Staates. Jedes Mehr an Rechten für das Individuum bedeutet ein Weniger an Rechten für den Staat, und umgekehrt. Beispielsweise bedeutet ein besserer Schutz der Privatsphäre der Bürger eine geringere Kontrollmöglichkeit für den Staat.

A. Beutelspacher, *Kryptologie*, DOI 10.1007/978-3-658-05976-7_6,

Die immer weiter wachsende Durchdringung auch des täglichen Lebens mit Computern und elektronischer Datenverarbeitung bringt nun die frühere Balance aus dem Gleichgewicht: Staaten und Konzerne haben Zugang zu einer riesigen Menge von Daten über die einzelnen Bürger; dadurch wird ein Missbrauch vorstellbar, gegenüber dem die in George Orwells Roman „1984" geschilderten Verhältnisse harmlos sind. Der einzelne Bürger hat keine Chance, zu wissen, welche Daten von ihm wo gespeichert sind, was mit diesen geschieht und was mit ihnen geschehen könnte. Das haben die Enthüllungen über die Aktivitäten der Geheimdienste überdeutlich gezeigt. Hier setzen die Datenschutzgesetze an. Sie versuchen, Probleme, die durch den Einsatz neuer Technik entstanden sind, juristisch zu lösen. Wir werden uns in den weiteren Abschnitten dieses Kapitels fragen, ob diese Probleme nicht auch technisch gelöst werden könnten (indem man nämlich die Systeme von vornherein so konzipiert, dass die Datenschutzprobleme nicht auftreten).

Genauso beängstigend ist die Tatsache, dass die großen Computersysteme und die mit ihnen realisierten Dienste durch Einzelpersonen bedroht sind. Finanzielle Transaktionen sind leichter manipulierbar, Hacker können erheblichen Schaden an fremden Daten und Programmen anrichten und so weiter. Ein Gegenmittel, das häufig empfohlen wird, sind mächtige und effiziente Systeme, die alle Handlungen der Benutzer aufzeichnen. Aber das erfordert noch mehr zentral verfügbare Informationen über die einzelnen Benutzer. Was tun? Vielleicht gibt es ja keine einfache Antwort. Lassen Sie uns einige Beispiele genauer diskutieren.

1. Es gibt Situationen, bei denen man allgemein überzeugt ist, dass Anonymität unvermeidbar ist. Eine Errungenschaft der modernen Gesellschaft, die eine Grundlage jeder Demokratie ist, sind geheime Wahlen. Stellen Sie sich nun vor, man würde elektronische Wahlen einführen. Dann müsste das Ergebnis der Wahl natürlich korrekt (sogar nachprüfbar) sein, die einzelnen Wähler (also die „Sender der Stimmen") müssten aber anonym bleiben.
   Die wissenschaftliche Forschung, die auf sensiblen persönlichen Daten beruht, ist ein weiteres Beispiel. Man denkt sofort an medizinische Untersuchungen über Drogenabhängigkeit oder AIDS. Hier ist es entscheidend, dass der Wissenschaft alle relevanten Daten zugänglich sind, ohne dass irgendjemand in der Lage ist, auf eine Person zurückzuschließen, von der die Daten stammen. Mit anderen Worten, man braucht Senderanonymität.
   Schließlich gibt es in jeder Gesellschaft Nischen, die auf einer (relativen) Anonymität basieren. Denken Sie zum Beispiel an die Telefonseelsorge oder an die „Anonymen Alkoholiker". Bei diesen Einrichtungen ist Anonymität eine entscheidende Voraussetzung für ihren Erfolg.

2. Im täglichen Leben haben wir ein glänzendes Beispiel von Senderanonymität, nämlich unser *Geld*, genauer gesagt *Münzgeld*. Wie bitte? Nun, eine Münze zeigt ihre Geschichte nicht: Ich kann mich vielleicht daran erinnern, von wem ich eine bestimmte Münze erhalten habe, aber ich habe keine Informationen darüber, wer früher diese Münze zu welchem Zweck benutzt hat.
   Dieses System ist so gut, dass es als Vorbild für viele Systeme mit Senderanonymität dient; wir werden in Abschn. 6.3 ein solches System vorstellen. Man darf aber nicht verdrängen, dass zumindest ein Teil des Übels, das mit Geld verbunden ist, von seiner Anonymität herkommt; Geld ist nämlich ein ideales Medium für Verbrechen. Manche Leute sind bereit, für Geld alles zu tun: Sie sind anonym, wenn sie ihr schmutziges Geld „verdienen", aber sie sind reich, mächtig, angesehen – und unverfolgbar –, wenn sie es ausgeben.
3. Mit der allgemeinen Verbreitung des Internets scheint auch ein Bedürfnis nach Anonymität zu wachsen, für dessen Befriedigung manche Leute viel Zeit investieren und sogar bereit sind, dafür Geld auszugeben. In Chats und Diskussionsforen kann jeder mitreden. Und wenn man nicht unter seinem Namen auftreten will, kann man ein Pseudonym benutzen. Natürlich wird dort nur ein relativ niedriges Anonymitätsniveau geboten, da alle Teilnehmer dem Betreiber bekannt sind. Dieser sorgt dafür, dass die Teilnehmer gegenseitig anonym bleiben können. Auch der Internetdienst par excellence, nämlich E-Mail bietet fast perfekte Anonymität: Ich kann mich bei einem Betreiber unter jedem Namen anmelden und so E-Mails versenden.
   Aber auch hier gibt es eine dunkle Seite: In Chats und Foren sagen Leute Dinge, die sie nie sagen würden, wenn sie nicht sicher wären, dass keiner herausfindet, wer es gesagt hat. Dies hängt mit einem Massenphänomen zusammen: Leute tun Dinge in der Masse (das heißt anonym), die sie ohne Anonymität nie in Erwägung zögen.
4. Eine wichtige Anwendung von Anonymität innerhalb des Gebietes der Kryptologie ist die Schlüsselerzeugung. Vermutlich gibt es in jedem Land, dessen Regierung die politisch relevante Information kryptographisch schützt, eine Zentrale, die die Schlüssel erzeugt, die später dann benutzt werden, um den diplomatischen Verkehr zwischen der Regierung und den einzelnen Botschaften zu verschlüsseln. Ich bin überzeugt, dass der arme Mensch, der die Erzeugung der Schlüssel wirklich vornimmt, nicht besonders gut bezahlt wird. Wenn er also wüsste, dass ein bestimmter Schlüssel dafür vorgesehen ist, im nächsten Monat den Verkehr, sagen wir, zwischen Berlin und Bagdad zu verschlüsseln, dann entstünde eine echte Gefahr (für die Nachrichten oder den Schlüsselerzeuger oder für beide). Was man hier

braucht, ist Anonymität. Man benötigt aber eigentlich noch mehr: Man muss in der Lage sein, nachweisen zu können, dass die Schlüsselerzeugung anonym erfolgt. (Stellen Sie sich eine Situation vor, in der der arme Mensch wirklich nichts weiß, der Gegner ihm aber nicht glaubt!)

5. Die Bedeutung der Anonymität verändert sich mit der Zeit, wie alle kulturellen oder politischen Phänomene. Ein Beispiel eines Aktes, den wir heute ohne Zweifel der Intimsphäre des Individuums zuordnen, der aber früher ganz anders betrachtet wurde, ist folgender: Noch vor 200 Jahren wurde die Notdurft öffentlich verrichtet: Es gab öffentliche „Notdurftanbieter", die die Kunden notdürftig mit einem Mantel umhüllten: Es wurde keinerlei Anstrengung unternommen, die Tatsache an sich, oder die handelnden Person anonym zu halten; nur der Inhalt wurde verschleiert. Weitere Details zu diesem Thema findet der interessierte Leser in [MBeu86].
   Bevor Sie das als weiteres Beispiel für meinen zweifelhaften Geschmack abtun, beachten Sie, dass wir auch hier die beiden Seiten der Medaille sehen können: Die öffentlichen Toiletten bieten heute ihre Anonymität auch zu Zwecken, die nicht von jedem gutgeheißen werden; dies reicht von der mehr oder weniger künstlerischen Gestaltung der Wände und Vandalismus über Drogenmissbrauch bis zur Ausführung von Schwerverbrechen, wie wir uns täglich im Fernsehen informieren können.

Meiner Meinung nach ist Anonymität kein Wert an sich. Man sollte sich bei jeder Anwendung überlegen, ob Anonymität notwendig ist, um den entsprechenden Dienst zu realisieren, und ob der Nutzen in einem vernünftigen Verhältnis zu den Kosten steht.

Obwohl also Anonymität nicht einfach „gut" oder „böse" ist, bin ich überzeugt, dass es ein wichtiges Ziel der Wissenschaft ist, deutlich zu machen, was man technisch erreichen kann und was nicht. Dann kann die Entscheidung, ob und wie etwas (etwa Anonymität) realisiert wird, sich auf wissenschaftliche Untersuchungen stützen und weist der politischen oder kommerziellen Entscheidung den richtigen Spielraum zu.

Eine interessante und überraschende Tatsache ist, dass gewisse Modelle von Anonymität mit kryptographischen Mechanismen realisiert werden können. Das Ziel dieses Kapitels ist es, solche Verfahren zu studieren.

Schon beim oberflächlichen Überfliegen der Seiten dieses Kapitels werden Sie merken, dass unter all den in diesem Buch studierten Techniken die in diesem Kapitel behandelten am spekulativsten sind. Aber meiner Meinung nach ist es besonders wichtig, Systeme zu entwickeln, die Anonymität ermöglichen, da sie deutlich machen, dass der Computer nicht von Natur aus der „Big Brother" ist, für den er, manchmal mit Grund, gehalten wird.

6.2 Drei (zu) einfache Modelle

Nach diesen philosophischen Gedankenflügen kehren wir zurück zur prosaischen Wissenschaft. Wir werden zunächst drei ganz einfache Modelle für Anonymität darstellen; anschließend sollen dann zwei etwas ausgereiftere Systeme behandelt werden.

6.2.1 Anonymität des Empfängers: Broadcasting

Prinzipiell ist es ganz einfach, Anonymität des Empfängers zu erreichen: Man schickt die Nachricht generell an alle Teilnehmer – oder jedenfalls an sehr viele. Die Broadcastmedien Rundfunk und Fernsehen sind die Prototypen für diese Art von Anonymität.

Erinnern Sie sich an die Zeiten, als sich Politiker über Fernsehen direkt an Terroristen wandten? Ein solches Vorgehen bietet eine perfekte Anonymität für die Angesprochenen. Harmlosere Beispiele findet man in den Anzeigenseiten von Zeitungen und Zeitschriften, zum Beispiel:

6.2.2 Anonymität des Senders: Pseudonyme

Ein Sender kann sich zur Tarnung ein *Pseudonym* (ein Name, der die echte Identität verbergen soll), oder auch mehrere Pseudonyme zulegen. Im Extremfall verwendet er zur Kommunikation mit jedem anderen Teilnehmer je ein anderes Pseudonym. Will er noch vorsichtiger sein, so kann er für jedes neue Gespräch ein neues Pseudonym verwenden. Unter solchen Umständen kann der Sender seine Identität völlig verbergen; er ist ebenso schwer dingfest zu machen wie die berühmte Stecknadel im Heuhaufen.

Die Schwierigkeit liegt auf der Hand: Der Sender selbst darf den Überblick über seine Pseudonyme nicht verlieren. Ebenso ist die Frage, wie vertrauliche Information von einem Pseudonym einem anderen mitgeteilt werden kann, schwierig und sicherlich noch nicht endgültig gelöst.

6.2.3 Anonymität der Kommunikationsbeziehung: Rauschen

Um zu verhindern, dass eine spezielle Kommunikationsbeziehung erkannt werden kann, sollte man *häufig*, im Extremfall also *immer* senden. Man muss also „Rauschen" erzeugen, oder, anders ausgedrückt: man muss viele „Dummy-Nachrichten" senden. Ein Mr. X, der unter solchen Umständen ein Kommunikationsprofil eines Teilnehmers erstellen will, muss den gesamten Datenverkehr aufnehmen, speichern, analysieren – und wird in 99,99 % aller Fälle nichts rauskriegen.

Zwei berühmte Beispiele aus dem zweiten Weltkrieg sollten hier erwähnt werden:

- Die Amerikaner wussten bereits kurz vor dem Angriff der Japaner auf Pearl Harbour, dass ein Angriff drohte, einfach aufgrund der Tatsache, dass die Zahl der japanischen Geheimnachrichten deutlich zugenommen hatte.
- Lange vor der Invasion in der Normandie 1944 überfluteten die Alliierten Europa mit Dummy-Nachrichten. Das deutsche Oberkommando der Wehrmacht wusste offenbar nie, ob eine Nachricht „echt" war (also den Befehl zum Beginn der Invasion enthielt), oder ob sie nichts sagend war; es war also immer unklar, ob die untergetauchten Hilfstruppen mobilisiert werden sollten oder nicht.

Die Nachteile einer solchen „mit roher Gewalt" erzwungenen Anonymität sind klar: Ein permanentes Senden würde jedes Netz sofort zusammenbrechen lassen – und auch in Zukunft würde dieses Verfahren ein Netzvolumen beanspruchen, das die kühnsten Wunschträume unserer postalischen und industriellen Verkabler weit übertrifft.

Jetzt sollen zwei Modelle vorgestellt werden, die auf trickreiche Weise aus kryptographischen Mechanismen aufgebaut sind.

6.3 Elektronisches Geld

Gewöhnliches Geld bietet (neben anderen Vorteilen) ein hohes Maß an Anonymität: Eine Person betritt ein Geschäft, wählt einige mehr oder weniger brisante Waren aus, geht damit an die Kasse, bezahlt in bar, verschwindet wieder – und hat keine Spur hinterlassen. Natürlich könnte die Kassiererin die Person wieder erkennen; der entscheidende Punkt ist aber, dass der *Weg des Geldes nicht zurückverfolgbar* ist, da der Kunde anhand des Geldscheins nicht identifiziert werden kann. Dies gilt insbesondere dann, wenn mit *Münzen* bezahlt wurde: Münzen besitzen keine individuelle Geschichte, eine ist so gut

wie jede andere. Was mit Münzen bezahlt wurde, weiß der Kunde, die Verkäuferin und sonst niemand.

Stellen sie sich nun ein System für elektronisches Bezahlen vor. Sehr wahrscheinlich würden in einem solchen System alle Daten für eine gewisse Zeit gespeichert werden; dies ist sowohl für den Abrechnungsprozess notwendig, als auch als Beweismittel im Falle von Reklamationen (vgl. die Diskussion in Abschn. 4.3). Ein solches System ist das Gegenteil eines Systems mit Anonymität. Stellen Sie sich nun vor, ein solches System müsste in ein System zum elektronischen Bezahlen umgewandelt werden, das Kundenanonymität bietet. Das scheint unmöglich – und ist vermutlich auch unmöglich! Um ein elektronisches System mit Anonymität zu entwickeln, muss man von vorne anfangen. Aber wie? Man wagt kaum zu glauben, dass Anonymität in ähnlichem Maße gewährleistet sein kann, wenn man seine Ware elektronisch bestellt und bezahlt.

Überraschenderweise kann man Münzen elektronisch nachbilden! Diese Idee für elektronische Münzen stammt von David Chaum [Cha85]. Er stellte überzeugend dar, dass sein Schema zum Schutz der Privatsphäre der Bürger in gewissem Sinne auch den Banken mehr Sicherheit bieten könnte.

Die folgenden Überlegungen setzen ein asymmetrisches Signaturschema voraus. Der Einfachheit halber nehmen wir an, es sei der RSA-Algorithmus.

Stellen wir uns vor, dass Professor W. Assolldas von seiner Bank eine elektronische Zweieuromünze erwerben möchte. Wir nehmen der Einfachheit halber an, dass auch eine Bank Münzen prägen kann. In Wirklichkeit haben natürlich nur die Nationalbanken oder die Europäische Zentralbank dieses Recht. Prof. Assolldas erwartet von einer elektronischen Münze wenigstens die folgenden Eigenschaften:

- Dieses Stück Information wird vom jedem Händler (und jedem Automaten) als Äquivalent von 2 Euro akzeptiert.
- Wenn es akzeptiert worden ist, kann kein Mensch mehr nachweisen, dass dieses Ding von Prof. Assolldas kam.

Nun zur Sache: Die Bank möge ein spezielles Paar aus öffentlichem und geheimem Schlüssel haben, mit dem sie die elektronische Zweieuromünze „macht“ (zertifiziert, prägt, …). Dazu verwendet sie den RSA-Algorithmus. Die Schlüssel bestehen also aus dem Modul n, dem öffentlichen Exponenten e beziehungsweise dem geheimen Exponenten d.

Um ein elektronisches Zweieurostück zu konstruieren, muss zunächst Prof. Assolldas arbeiten. Er wählt zufällig zwei große Zahlen C und V. Wie wir sehen

werden, stellt dabei V das Rohmaterial für die Münze dar, während C nur zur Camouflage dient.

Danach bildet er die Zahl W, die durch zweimaliges Hintereinanderschreiben der Zahl V entsteht. Wenn beispielsweise V = 123.456 ist, dann ist W = 123.456.123.456. Die einzige schwierige Aufgabe für Prof. Assolldas ist es, die folgende Zahl S zu berechnen:

$$S = C^e \cdot W \bmod n.$$

Dann schickt er die Zahl S an seine Bank und bittet darum, aus diesem Zahlenwurm ein elektronisches Zweieurostück zu machen.

Jetzt ist die Bank an der Reihe. Vermutlich wird sie als erstes Prof. Assolldas' Konto um 2 Euro erleichtern. Dann potenziert sie die Zahl S mit ihrem geheimen 2 Euro-Schlüssel d:

$$T = S^d \bmod n.$$

Diese Zahl T wird an Prof. Assolldas zurückgeschickt. Sie ist jedoch noch nicht die Münze. Prof. Assolldas erhält seine Münze, indem er T durch die Verschleierungszahl C „teilt":

$$Z := T \cdot C^{-1} \bmod n.$$

Beachten Sie, dass C^{-1} eine natürliche Zahl ist, für die $C \cdot C^{-1} \bmod n = 1$ gilt. Die Existenz einer solchen „modularen Inversen" haben wir in Abschn. 5.3.2.2 nachgewiesen.

Diese Zahl Z ist nun endlich das elektronische Zweieurostück!

Was aber ist diese Zahl? Können wir sie ausrechnen? Nun, wenn alles mit rechten Dingen zugegangen ist, dann gilt

$$\begin{aligned} Z = T^{-1} \bmod n &= S^d \cdot C^{-1} \bmod n = C^{ed} \cdot W^d \cdot C^{-1} \bmod n \\ &= C \cdot W^d \cdot C^{-1} \bmod n = W^d \bmod n. \end{aligned}$$

Das elektronische Zweieurostück Z hängt also nicht von C ab, sondern nur von den geheimen 2 Euro-Schlüssel d der Bank und der Zahl V, die von Prof. Assolldas gewählt wurde. Die Zahl C dient also wirklich nur zur Verschleierung der wirklichen Verhältnisse. Man spricht auch von „Blendung".

Achtung: Obige Berechnung kann man nur von einem (nicht existenten) höheren Standpunkt aus nachvollziehen. In Wirklichkeit kennt ein Außenstehender weder die von Prof. Assolldas geheim gewählten Zahlen V und C noch den geheimen Schlüssel d der Bank.

Nun müssen wir uns davon überzeugen, dass die Zahl Z den obigen Forderungen genügt.

- *Jedermann kann leicht verifizieren, dass Z ein elektronisches Zweieurostück ist.*

Um diese Behauptung nachzuprüfen, potenziert man Z mit dem öffentlichen 2 Euro-Schlüssel der Bank; es ergibt sich

$$Z^e \bmod n = W^{de} \bmod n = W.$$

Das Kriterium dafür, ein Zweieurostück zu sein, ist also, dass Z^e (in Wirklichkeit also W) aus zwei gleichen Hälften besteht. In Übungsaufgabe 3 werden wir sehen, dass man an der Gestalt von Z nicht erkennen kann, ob Z^e aus zwei gleichen Hälften besteht. Mit anderen Worten: Man kann nicht die Bank umgehen und selbst Münzen „prägen".

Wenn Z bei der Bank eingereicht wird, so vergewissert sie sich zunächst, wie zuvor der Händler, dass es sich wirklich um ein elektronisches Zweieurostück handelt. Danach schreibt sie dem Konto des einreichenden Händlers 2 Euro gut – aber nur, wenn diese spezielle Münze zum ersten Mal eingereicht wurde! Ein betrügerischer Händler könnte ja auf die Idee kommen, zweimal oder noch öfter dieselbe Münze einzureichen. Denn eine Münze ist eben nur ein Bitmuster, und niemand kann daran gehindert werden, dieses zu duplizieren. Wir werden auf dieses Problem später eingehen.

- *Niemand kann von Z auf Prof. Assolldas schließen.*

Kann der Händler (oder irgendjemand, der im Besitz von Z ist) auf die Identität von Prof. Assolldas schließen? Die einzige Möglichkeit ist, bei der Bank nachzufragen, an wen denn das Zweieurostück Z ausgegeben wurde. Das kann aber die Bank nicht sagen. Sie hat Z ($= W^d \bmod n$) nie gesehen, sondern nur die Zahlen

$$S = C^e \cdot W \bmod n \text{ und } T = C^{ed} \cdot W^d = C \cdot W^d \bmod n.$$

Da in beiden der Bank bekannten Zahlen die Verschleierungszufallszahl C vorkommt, kann kein noch so mächtiger Inquisitor die Zahl W (und damit Z) rauskriegen.

David Chaum selbst hat dieses Verfahren mit alltäglichen Dingen spielerisch illustriert. Dabei handelt es sich natürlich nicht um einen ernstzunehmenden Produktvorschlag; vielmehr liegt der Nutzen dieser Beschreibung darin, dass man sich das Verfahren plastisch vorstellen kann.

Prof. Assolldas schickt ein leeres Blatt Papier zur Bank; dieses Blatt steckt er in einen Briefumschlag, in welchem auch ein Kohlepapier liegt – es handelt sich also um eine sehr rückständige Technik. Die Bank öffnet den Briefumschlag nicht, sondern sie drückt nur ihren Zweieurostempel auf den Briefumschlag. Dadurch drückt sich die Aufschrift „2 Euro“ mittels des Kohlepapiers auf das Blankoblatt durch. Prof. Assolldas entnimmt nun dieses Blatt dem Umschlag, packt es in einen anderen und bezahlt damit beim Händler.

Wie vorher hat auch in diesem Modell die Bank keine Kenntnis, was des Pudels Kern ist. Da sie nur die Umschläge, nicht aber die Inhalte kennt, kann sie nicht zurückverfolgen, an wen das elektronische Zweieurostück ausgegeben wurde.

Zum Abschluss dieses Abschnitts eine Bemerkung zum Vergleich der Sicherheit von elektronischen Münzen und unserem üblichen Hartgeld. Bei echten Münzen ist es für praktische Zwecke gleich schwer, aus dem Nichts eine Münze zu schaffen wie eine existierende Münze zu kopieren. Ganz anders bei elektronischen Münzen: Hier kann zwar mit kryptographischen Mechanismen auf sehr sichere Weise erreicht werden, dass niemand eine elektronische Münze „aus dem Nichts“ (in unserem Beispiel heißt das: ohne Kenntnis der Bankschlüssel) erzeugen kann. Aber, im Gegensatz dazu ist es ein Kinderspiel, eine Münze zu duplizieren: Man muss nur die entsprechende Zahl bzw. das entsprechende Bitmuster kopieren!

Folglich muss die Bank eine Liste aller bereits eingelösten Münzen führen. Es ist klar, dass man dafür nicht nur riesigen Speicherplatz braucht, sondern dass es jedes Mal, wenn eine Münze eingereicht wird, auch sehr viel Zeit kostet, um zu prüfen, ob ein Doppelgänger dieser speziellen Münze bereits eingelöst wurde. Ein weiterer Nachteil ist, dass jeder Händler, der eine Münze erhält, diese sofort einlösen muss – denn sonst könnte ihm ja jemand zuvor kommen. Man spricht von einem *Online-Münzsystem*.

Es gibt auch Vorschläge für *Offline-Münzsysteme* (siehe zum Beispiel [CFN90]; dieses Verfahren wird auch in [Beu12] beschrieben). Diese Münzsysteme haben folgende Eigenschaft: Wenn eine Münze nur einmal eingelöst wird, bleibt der Kunde vollkommen anonym; wenn allerdings versucht wird, die Münze ein zweites Mal einzureichen, kann die Bank denjenigen, an den die Münze ausgegeben wurde (in unserer Sprache Prof. Assolldas) dingfest machen.

6.4 MIX as MIX can

Stellen wir uns vor, ich möchte einen Brief an eine bestimmte Person (ich verrate nicht, an wen!) schicken, ohne dass jemand erfährt, wer an wen diesen Brief geschickt hat. Ich werde dann natürlich zunächst keinen Absender auf den Umschlag schreiben. Mit der Anschrift wird es schwieriger: Eigentlich darf niemand die Adresse kennen; sie muss also verborgen sein – aber mindestens der Postbote, der den Brief austrägt, muss doch die Anschrift wissen!

Die Lösung besteht darin, dass alle diejenigen, die solche Anonymitätsbedürfnisse haben, ihren Brief (mit Umschlag) an eine Deckadresse schicken. Dort wird der äußere Umschlag (mit der Deckadresse) entfernt, alle Briefe durcheinander gemixt (damit ein Beobachter aus der Reihenfolge der Ein- und Ausgänge keine Schlüsse ziehen kann) und dann an die richtigen Adressaten geschickt. (Siehe Abb. 6.1, 6.2, 6.3, 6.4) Diese Deckadresse nennt man *MIX*. Das Konzept der MIXe wurde in [Cha81] erstmalig vorgestellt.

Beachten Sie, dass dies im Wesentlichen der Vorgang ist, der bei einer Briefwahl benutzt wird, um Anonymität zu gewährleisten.

Es ist klar, dass dieser Dienst etwas kostet: Man muss den Service des MIXes bezahlen, und man braucht das doppelte Porto.

Wie kann man diesen anschaulichen Vorgang umsetzen in ein elektronisches System? Die Idee besteht darin, das Einpacken eines Briefes in einen Umschlag durch *Verschlüsseln* nachzubilden. Beachten Sie, dass beim Einste-

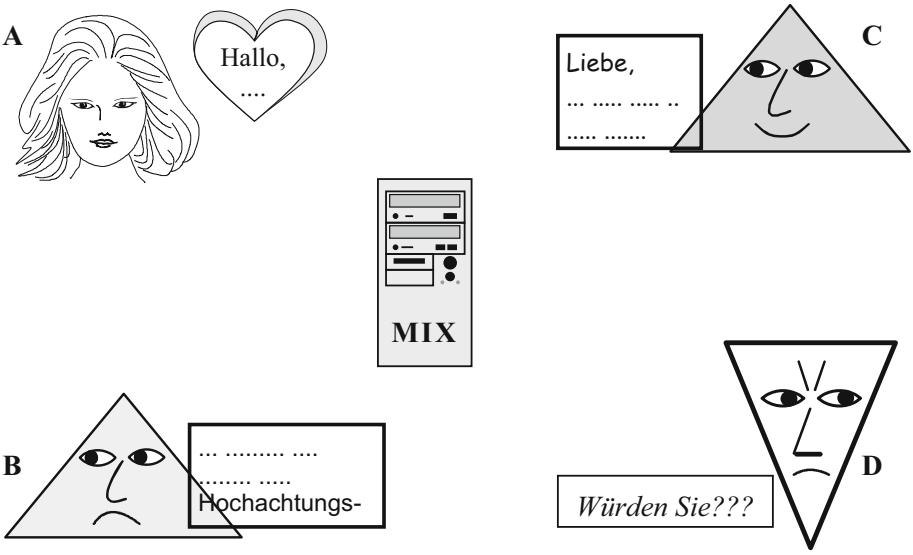


Abb. 6.1 Erster Akt: Jeder schreibt einen Brief

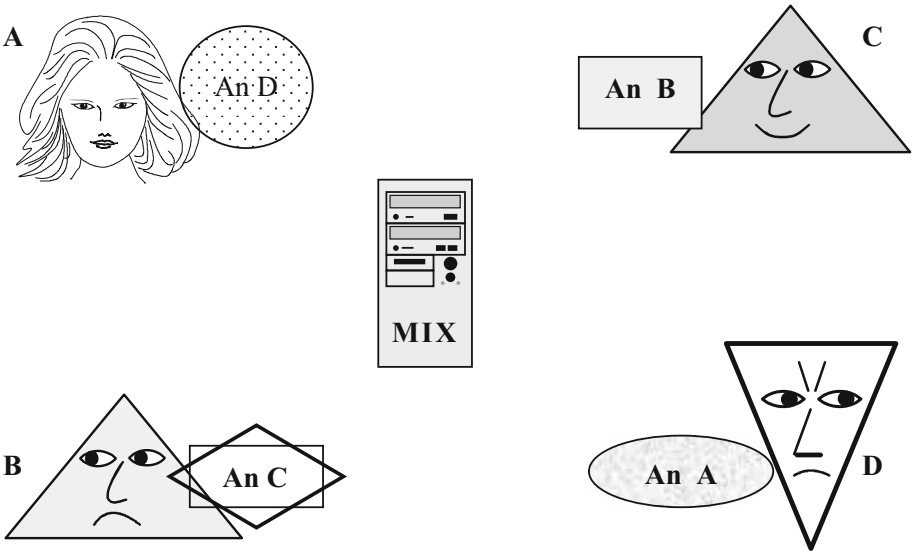


Abb. 6.2 Zweiter Akt: Jeder steckt seinen Brief in einen kleinen Umschlag und adressiert ihn

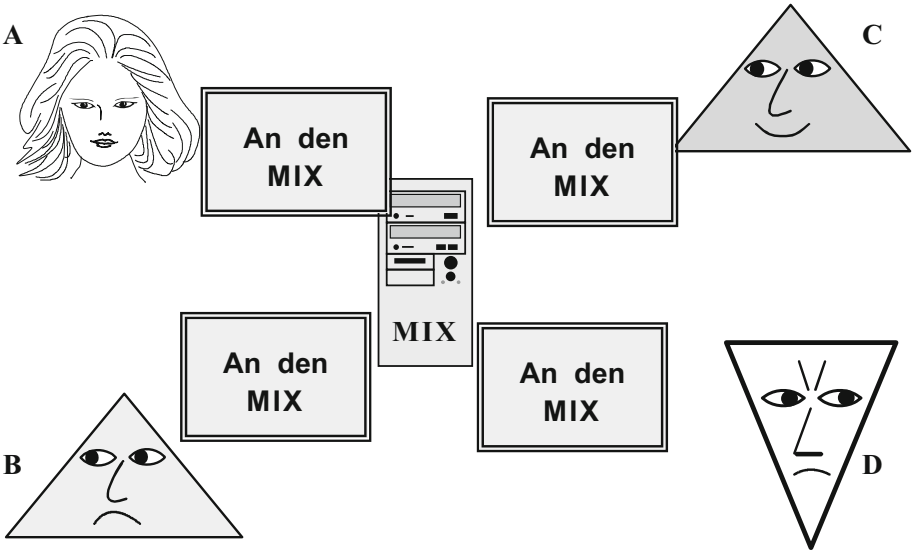


Abb. 6.3 Dritter Akt: Jeder steckt den Umschlag in einen großen Umschlag und schickt ihn an den MIX

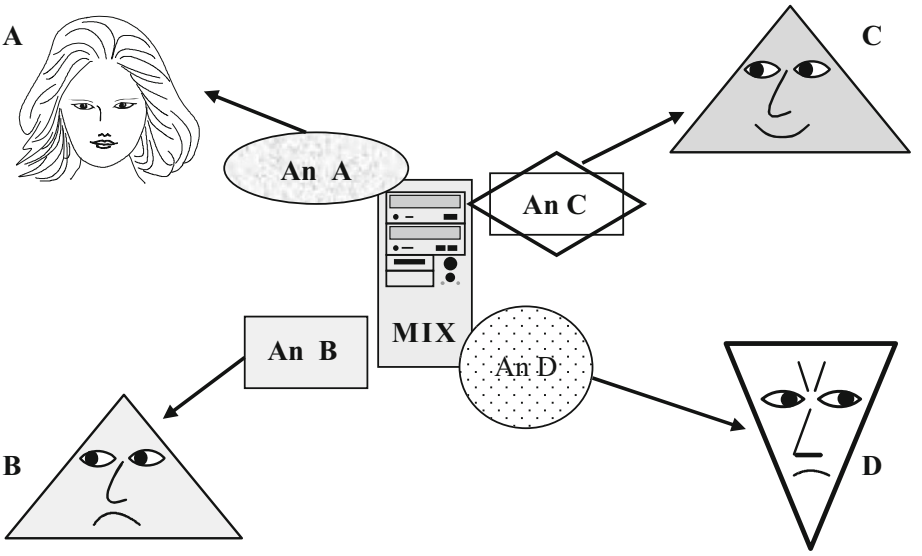


Abb. 6.4 Vierter Akt: Der MIX öffnet die großen Briefe und schickt jeden Brief an den „eigentlichen" Empfänger

cken eines kleinen Umschlags in den großen die Adresse unsichtbar wird; das bedeutet, dass beim entsprechenden Verschlüsselungsvorgang die Adresse des Empfängers mitverschlüsselt werden muss.

Dem Protokoll liegt ein asymmetrisches Verschlüsselungsschema zugrunde. Stellen wir uns nun vor, ein Sender A möchte eine Nachricht m an den Empfänger B schicken. Dazu verschlüsselt er m zusammen mit Bs Adresse a_B unter dem öffentlichen Schlüssel E des MIXes. Also sendet er $E(m, a_B)$ an den MIX. Dieser kann den erhaltenen Geheimtext mit seinem geheimen Schlüssel D entschlüsseln und erhält m und a_B.

Danach permutiert der MIX alle bis zu einem bestimmten Zeitpunkt eingelaufenen Nachrichten und schickt diese dann an den richtigen Empfänger; beispielsweise schickt er die Nachricht m an B.

Da kein offensichtlicher Zusammenhang ersichtlich ist zwischen den Nachrichten, die an den MIX geschickt werden, und den Nachrichten, die der MIX weiterschickt, kann niemand die Nachrichten vom Empfänger zum Sender zurückverfolgen.

Ein solches System kann nur dann seine Funktion richtig erfüllen, wenn auch gewisse nichtkryptologische Voraussetzungen gegeben sind; die zwei wichtigsten seien hier genannt.

1. Das ganze System nützt nichts, wenn insgesamt nur sehr wenige Nachrichten gesendet werden. Es ist daher äußerst wichtig, dass kontinuierlich „viele“ Nachrichten ausgetauscht werden.
2. Ein ernstes Problem: Muss bei einem solchen System nicht jeder dem MIX vertrauen? Mit anderen Worten: Ist der MIX jetzt der „Big Brother“? Ist er die Krankheit, deren Therapie zu sein, er behauptet? Diese Gefahr besteht – aber genau aus diesem Grund wird vorgeschlagen, mehrere MIXe zu verwenden; diese könnten auch von verschiedenen, voneinander unabhängigen gesellschaftlichen Gruppen gestellt werden: von der Regierung, von den Gewerkschaften, den Verbraucherverbänden, vom TÜV, von den Kirchen, … Man könnte sich sogar den Chaos Computer Club als Anbieter eines MIXes vorstellen. Dann könnte jeder Teilnehmer die MIXe wählen, denen er zutraut, dass sie insgesamt die Anonymität liefern, die er sich wünscht.

Nun ist es leider so, dass beim Einsatz vieler MIXe nicht nur die Protokolle sehr kompliziert werden, sondern die MIXe werden auch zu empfindlichen Flaschenhälsen.

Wir stellen hier nur den Fall dar, in dem ein Benutzer A zwei MIXe, MIX_1 und MIX_2 wählt. Zunächst wird er die Nachricht m zusammen mit der Adresse a_B von B verschlüsseln, und zwar unter dem öffentlichen Schlüssel E_2 von MIX_2; er erhält $E_2(m, a_B)$.

Aber damit ist A noch nicht fertig. Er muss zu dem berechneten Wert die Adresse a_2 von MIX_2 hinzufügen und dann alles mit dem öffentlichen Schlüssel E_1 von MIX_1 verschlüsseln: dadurch erhält er

$$E_1(E_2(m, a_B), a_2)$$

und schickt nun dieses Konglomerat an MIX_1. Dieser MIX beginnt mit der Entschlüsselungsprozedur: Durch Anwenden seines geheimen Schlüssels erhält er $E_2(m, a_B)$ und a_2. Damit weiß er, dass es seine Pflicht und Schuldigkeit ist, den Datensatz $E_2(m, a_B)$ an den MIX mit Adresse a_2, also an MIX_2 zu schicken. Dieser kann den Rest dechiffrieren und schließlich die eigentliche Nachricht m an den Teilnehmer B schicken.

Noch Fragen?

Der Nachteil einer solchen Prozedur ist sofort klar: Jede Nachricht muss mehrere Male ver- und entschlüsselt werden. Ferner werden die Nachrichten immer länger, so dass – bei heutiger Technologie – ein System mit mehreren MIXen sich bald selbst lahm legen würde. Das bedeutet: Wenn man Anonymität mit dieser Methode realisieren wollte, müsste man einen ganz erheblichen technischen Aufwand treiben. Dies ist der Preis, den man (bei der Verwendung von MIXen) für Kommunikationsanonymität bezahlen muss!

Man muss feststellen, dass die MIX-Technologie, die in den 80er und 90er Jahren des vergangenen Jahrhunderts von vielen als bloße akademische Spielerei angesehen wurde, mit der realen Gefahr der universellen Überwachung durch staatliche und andere Organisationen eine neue Bedeutung bekommen hat.

Zusatzinformation: Kann Kryptologie unsere Banknoten sicherer machen?

Die Notenbanken aller Länder sind über die technologische Entwicklung nicht immer erfreut: Maschinen, wie etwa Farbkopierer und Scanner, die auch zum Fälschen von Banknoten dienen können, werden immer besser und gleichzeitig immer billiger. Das heißt, dass der zeitliche technologische Vorsprung der Notenbanken vor potentiellen Fälschern immer geringer wird.

Bevor elektronische Münzen oder Banknoten flächendeckend eingeführt sind, muss man das real existierende Geld sicherer machen. Hier könnte Kryptologie, insbesondere asymmetrische Signaturschemata, eine neue Dimension von Sicherheit ermöglichen. Das könnte etwa wie folgt geschehen (vergleiche [Omu90]). Bei der Herstellung einer Banknote wird ein zufälliges physikalisches Muster im Papier der Banknote erzeugt. Dafür sind viele Möglichkeiten denkbar, zum Beispiel könnte man kleine Partikel in das Papier einfügen. Wichtig ist, dass dieser Prozess in dem Sinne zufällig ist, dass ein Fälscher nicht ein bestimmtes, vorgegebenes Muster erzeugen kann. Dann wird dieses Muster digitalisiert; die entstehende Zahl nennen wir MUS. Dieses Muster variiert also von Banknote zu Banknote. Bei der Herstellung signiert die Notenbank die Zahl MUS, d. h. sie wendet ihren geheimen Schlüssel auf MUS an. Die so erhaltenen Signatur SIG wird (beispielsweise) in eine Folge SIG* von Zahlen codiert, die deutlich lesbar auf die Banknote gedruckt werden.

Um die Echtheit der Banknote zu überprüfen, muss die Signatur verifiziert werden. Dazu liest man SIG*, übersetzt dies zurück in SIG, wendet den öffentlichen Schlüssel auf SIG an und erhält MUS. Dieser Datensatz wird dann mit dem Zufallsmuster verglichen, das über einen Sensor direkt von der Banknote gelesen wird. Da Banknoten im Laufe ihres Lebens immer mehr Altersflecken, Runzeln und Risse erhalten, wird der Sensor in der Regel nicht das jungfräuliche Muster MUS erkennen, sondern ein leicht verändertes Muster MUS′. Das Protokoll sagt nun, dass die Banknote als echt akzeptiert wird, wenn MUS und MUS′ in mindestens x% ihrer Bits übereinstimmen. Ein typischer Wert ist x = 80, wobei MUS aus etwa 3000 Bits besteht.

Beachten Sie, dass, dass man einen Algorithmus braucht, der aus SIG wieder MUS berechnet; unter den heutigen Algorithmen bietet sich dafür der RSA-Algorithmus an. Außerdem ist entscheidend, dass SIG* fehlerfrei gele-

sen wird (sonst würde man keinerlei Information erhalten); deswegen schützt man diese Daten so, dass keine Fehler auftauchen; man kann hierfür etwa fehlerkorrigierende Codes benutzen.

6.5 Übungsaufgaben

1 Diskutieren Sie ein weiteres Beispiel aus dem täglichen Leben, bei dem – Ihrer Meinung nach – *Anonymität* notwendig ist, sowie ein Beispiel, bei dem Sie Anonymität für schädlich halten.
Können Sie sich ein Beispiel vorstellen, bei dem eine wichtige gesellschaftliche Gruppe für Anonymität plädieren würde, eine andere, ebenso angesehene, dagegen?
2 Machen Sie sich die Vor- und Nachteile der Anonymität des Münzgeldes für die Gesellschaft klar. Vergleichen Sie dies mit einem Kreditkartensystem.
3 Wählen Sie die Werte von p, q, d und e von Übungsaufgabe 9 in Kap. 5 und berechnen Sie die Zahlen 111^d mod pq, 222^d mod pq, 333^d mod pq, …, 777^d mod pq.
4 Machen Sie sich die Analogie zwischen dem Schema für elektronische Münzen und dem Schema mit Papier, Kohlepapier und Umschlägen klar. Was entspricht C und was entspricht d?
5 Überlegen Sie sich, ob Chaums Schema für elektronische Münzen die folgenden Eigenschaften hat.

- Die Münze behält ihre Gültigkeit, wenn Prof. Assolldas ein kleines bisschen an ihr herummanipuliert.
- Der Händler kann die Münze benutzen, um damit selbst Waren einzukaufen.
- Der Händler kann eine Münze nur einmal einreichen.
- Hat echtes Münzgeld diese Eigenschaften?

6 Überlegen Sie, ob die Methode, mit der Emil in Erich Kästners *Emil und die Detektive* nachweist, dass der Herr mit dem steifen Hut ihm 140 Mark gestohlen hat, auch anwendbar gewesen wäre, wenn Emil elektronische Geldscheine benutzt hätte.
7 Schreiben Sie das Protokoll für drei MIXe.
8 Überlegen Sie, ob ein MIX (oder mehrere MIXe) dazu benutzt werden können, um das Problem der Anonymität bei der Schlüsselerzeugung zu lösen.

Ausklang

Bevor sich unsere Wege wieder trennen, möchte ich Ihnen, verehrte Leserin, lieber Leser, das folgende Wort des Dichter Novalis (1772–1801) in Erinnerung rufen.

In diesem Gedicht drückt sich die Sehnsucht nach einer Welt aus, in der Kryptologie überflüssig ist. Nach Meinung des Dichters wird diese Welt Wirklichkeit, wenn man nur ein Zauberwort weiß und es ausspricht:

Wenn nicht mehr Zahlen und Figuren
Sind Schlüssel aller Kreaturen,
wenn die, so singen oder küssen,
mehr als die Tiefgelehrten wissen,
wenn sich die Welt ins freie Leben
und in die Welt wird zurückbegeben,
wenn dann sich wieder Licht und Schatten
zu echter Klarheit werden gatten
und man in Märchen und Gedichten
erkennt die wahren Weltgeschichten,
dann fliegt von einem geheimen Wort
das ganze verkehrte Wesen fort.

A. Beutelspacher, *Kryptologie*, DOI 10.1007/978-3-658-05976-7,

Entschlüsselung der Geheimtexte

Im folgenden finden Sie Lösungshinweise zu einigen Übungsaufgaben. Die Hinweise sind so gestaltet, dass Sie noch die Chance haben, ein wenig nachzudenken.

Kapitel 1

Übungsaufgabe **1**: Ich wusste es ja!
Übungsaufgabe **15**: Moni, Moni!
Übungsaufgabe **23**: Das Konzert beginnt *vor* Seite 1.

Kapitel 2

Übungsaufgabe **5**: BXMFF, LEU, ZHUM
Übungsaufgabe **7**: Lesen Sie in [Sin02] den Abschnitt über Blaise de Vigenère.
Übungsaufgabe **17**: NEIN!

Kapitel 5

Übungsaufgabe **2**: Einer meiner amerikanischen Freunde, der diese Aufgabe lösen sollte, schrieb mir:

> You should have given a hint: The man is a second-rate mathematician! Then I would have gotten it right away. He's the French mathematician *Charles P. Tebeau*. Of course, it could also been the German *Albrecht E. Pause*, who is not so well known. My colleague, who is very clever at these things, argued that it must be *Beulah C. Streep*, the feminist author from the Bronx who ran unsuccessfully for congress; his second guess would be *Peaches Butler*, a porno star from Atlanta, Georgia, suspected of having an affair with the governor.

Anmerkungen zur Literatur

Der Klassiker von Kahn [Kah96] und das Buch von Franke [Fra82] sind sehr lesenswerte Darstellungen der Geschichte der Kryptologie, in denen insbesondere die Entwicklungen bis 1945 detailliert geschildert sind. Zur Vertiefung der in diesem Buch dargestellten Themen können [BSW], [BP82], [DP89], [FR94], [Ruh87] und [Schn00] dienen, während die Lektüre der empfehlenswerten kryptographischen Bücher [BNS], [Buc10], [Den83], [HKW85], [Hor85], [Kob87], [Koh81], [SP89] zum Teil größere mathematische Anforderungen an den Leser stellt.

Unter den regelmäßigen Veröffentlichungen zur Kryptologie müssen die folgenden Zeitschriften erwähnt werden:

- Journal of Cryptology (Springer Verlag),
- Designs, Codes and Cryptography (Kluwer) und
- Computer & Security (Elsevier),

die einen theoretisch-wissenschaftlichen Anspruch haben, während

- Cryptologia (Rose Hulman Institute)

sich schwerpunktmäßig mit der Geschichte der Kryptologie beschäftigt;

- KES Zeitschrift für Kommunikations- und Datensicherheit (Peter Hohl Verlag)

ist eine allgemeinverständliche Zeitschrift, in der man auch Beschreibungen von Produkten findet;

- Datenschutz und Datensicherung (Vieweg Verlag)

stellt insbesondere Zusammenhänge zwischen Technik und Recht dar.

Die Lecture Notes in Computer Science (Springer) veröffentlichen jährlich zwei Bände unter dem Titel *Advances in Cryptology*; dies sind die Proceedings der wichtigsten Tagungen über Kryptologie, nämlich der EUROCRYPT, ASIACRYPT und der CRYPTO. Zusammen mit dem Journal of

Cryptology sind diese Bände ein Muss für jeden, der sich einen Eindruck vom aktuellen Fortschritt der Kryptologie verschaffen will.

Ein Verzeichnis aller Listen mit frequently asked questions ist unter

http://www.faqs.org/faqs/cryptography-faq/

zu finden. Die Seiten

http://www.infoserversecurity.org
http://www.tcs.hut.fi/~helger/crypto/
http://www.bsi.de

sind sehr gute Startseiten mit vielen Kryptographie-Links.

Programme der meisten Algorithmen zum Selbst-Ausprobieren findet man unter

http://www.cryptool.de.

In den Newsgroups: *sci.crypt*, *sci.crypt.research* wird – wie in den meisten Newsgroups – alles mögliche diskutiert, manchmal auch etwas Wichtiges.

Literatur

Bau. Friedrich L. Bauer: Entzifferte Geheimnisse. Methoden und Maximen der Kryptologie. Springer-Verlag, Berlin, Heidelberg, 2000.

BDG95. J. L. Balcazar, J. Diaz, J. Gabarro: Structural Complexity I. Springer-Verlag, Heidelberg, 2. Auflage 1995.

BDPW90. M.V.D. Burmester, Y. Desmedt, F. Piper, M. Walker: A General Zero-Knowledge Scheme. Advances in Cryptology – EUROCRYPT '89. Springer Lecture Notes in Computer Science 434 (1990), 122-133.

Beu12. A. Beutelspacher. Geheimsprachen. C.H. Beck Verlag München, 5. Auflage 2012.

Beu86. A. Beutelspacher: Luftschlösser und Hirngespinste. Vieweg, Braunschweig und Wiesbaden 1986.

BFS92. Th. Beth, M. Frisch, G.J. Simmons: Public-Key Cryptography: State of the Art and Future Directions. Springer, Lecture Notes in Computer Science 578 (1992).

BKP91. A. Beutelspacher, A. Kersten, A. Pfau: Chipkarten als Sicherheitswerkzeug. Springer-Verlag, Heidelberg 1991.

BNS. A. Beutelspacher, H. B. Neumann, T. Schwarzpaul. Kryptografie in Theorie und Praxis. Vieweg, Braunschweig und Wiesbaden, 2. Auflage 2010.

BP82. H. Beker and F. Piper: Cipher Systems. The Protection of Communication. Northwood, London 1982.

BR89. A. Beutelspacher, U. Rosenbaum: Sicherer Zugang zu Betriebssystemen mit der Chip-Karte. 8. GI-Fachgesprächs über Rechenzentren, Informatik-Fachberichte 207 (Hg. J. Knop), 186-193 (1989).

BRK08. A. Bartholome, J. Rung, H. Kern: Zahlentheorie für Einsteiger. Vieweg, Braunschweig und Wiesbaden, 6. Auflage 2008.

BS91. E. Biham and A. Shamir: Differential Cryptanalysis of DES-like Cryptosystems. J. Cryptology 4 (1991), 3-72.

BS93. A. Beutelspacher, J. Schwenk: Was ist Zero-Knowledge? Math. Semesterber. 40, 73-85 (1993).

BS93. E. Biham and A. Shamir: Differential Cryptanalysis of the Data Encryption Standard. Springer-Verlag, Berlin, … 1993.

BSW. A. Beutelspacher, J. Schwenk, K.-D. Wolfenstetter: Moderne Verfahren

der Kryptographie. Von RSA bis Zero-Knowledge. Vieweg, Braunschweig und Wiesbaden, 8. Auflage 2014.

Buc10. J. Buchmann: Einführung in die Kryptographie. Springer-Verlag, Berlin, Heidelberg, 5. Auflage 2010.

CBZ99. C. Creutzig, A. Buhl, P. Zimmermann: PGP. Pretty Good Privacy. Der Briefumschlag für Ihre elektronische Post. FoeBuD e. V. Bielefeld 1999.

CCITT. CCITT Recommendation X.509: The Directory – Authentication Framework, 1988.

CFN90. D. Chaum, A. Fiat, M. Naor: Untraceable Electronic Cash. Advances in Cryptology – CRYPTO '88. Springer Lecture Notes in Computer Science 403, 1990, 319-327.

Cha81. D. Chaum: Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. Comm. ACM 24 (1981), 84-88.

Cha85. D. Chaum: Security without Identification: Transaction systems to Make Big Brother Obsolete. Comm. ACM 28 (1985), 1030-1044.

Den83. D. Denning: Cryptography and Data Security. Addison Wesley, Reading, Mass. 1983.

DH76. W. Diffie, M.E. Hellman: New directions in cryptography. Trans. IEEE Inform. Theory, IT-22, 6 (1976), 644-654.

Dif88. W. Diffie: The First Ten Years of Public-Key Cryptography. Proceedings of the IEEE 76 (5) (1988), 560-577.

DP89. D.W. Davies, W.L. Price: Security for Computer Networks. John Wiley & Sons, Chichester 1984, 2nd edition 1989.

DR01. J. Daemen and V. Rijmen: The Design of Rijndael. Springer, 2001.

ElG85. T. ElGamal: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. on Inform. Theory, Vol. IT-31 (1985), 469-472.

FFKK93. O. Fries, A. Fritsch, V. Kessler, B. Klein (Hrsg.): Sicherheitsmechanismen. Bausteine zur Entwicklung sicherer Systeme. Oldenbourg Verlag, München, Wien 1993.

FP90. W. Fumy, A. Pfau: Asymmetric Authentication Schemes for Smart Cards – Dream or Reality? Proc. IFIP SEC '90. Espoo, Finnland.

FR94. W. Fumy, H.P. Rieß: Kryptographie – Entwurf, Einsatz und Analyse symmetrischer Kryptosysteme. Oldenbourg, München 21994.

Fra82. H.W. Franke: Die geheime Nachricht. Umschau-Verlag, Frankfurt/Main 1982.

Fra84. O.I. Franksen: Mr. Babbage's Secret. The Tale of a Cypher – and APL. Prentice Hall, Englewood Cliffs, 1984.

FS87. A. Fiat, A. Shamir: How To Prove Yourself: Practical Solutions to Identification and Signature Problems. Advances in Cryptology – CRYPTO '86. Springer Lecture Notes in Computer Science 263 (1987), 186-194.

GMR89. S. Goldwasser, S. Micali, C. Rackoff: The Knowledge Complexity of Interactive Proof-Systems. SIAM J. Comput. 8(1) (1989), 186-208.

Gor85. J. Gordon: Strong Primes are Easy to Find. Advances in Cryptology – EUROCRYPT '84. Springer Lecture Notes in Computer Science 209 (1985), 216-223.

Hen13. N. Henze: Stochastik für Einsteiger. Eine Einführung in die faszinierende Welt des Zufalls. Springer-Spektrum, 10. Auflage 2013.

HKW85. F.-P. Heider, D. Kraus, M. Welschenbach: Mathematische Methoden der Kryptoanalyse. Vieweg, Braunschweig, Wiesbaden 1985.

Hon73. R. Honsberger: Mathematical Gems I. MAA 1973.

Hor85. P. Horster: Kryptologie. B.I.-Wissenschaftsverlag, Mannheim, Wien, Zürich 1985.

Hor96. P. Horster (Hrsg.): Digitale Signaturen. Grundlagen, Realisierungen, Rechtliche Aspekte, Anwendungen. DuD-Fachbeiträge. Verlag Vieweg, Braunschweig/Wiesbaden 1996.

ISO. ISO IS 7498/2: Open Systems Interconnection Reference Model – Part 2: Security Architecture.

Jö01. D. Jörgensen: Der Rechenmeister, Aufbau TB, Berlin 2001.

Kah96. D. Kahn: The Codebreakers. Macmillan, New York, revised edition 1996.

Kip. R. Kippenhahn: Verschlüsselte Botschaften. Geheimschrift, Enigma und Chipkarte. rororo-Taschenbuch, 4. Auflage 2005.

Kob87. N. Koblitz: A Course in Number Theory and Cryptography. Springer-Verlag, New York 1987.

Kob98. N. Koblitz: Algebraic Aspects of Cryptography. Springer-Verlag, New York 1998.

LM90. A.K. Lenstra and M.S. Manasse: Factoring by electronic mail. Advances in Cryptology – EUROCRYPT '89. Springer Lecture Notes in Computer Science 434 (1990), 355-371.

Mas69. J.L.Massey: Shift-register Synthesis and BCH Decoding. IEEE Inform. Theory, IT-15, 1 (1969), 122-127.

Mas83. J.L. Massey: Logarithms in finite cyclic groups – cryptopgraphic issues. Proceedings of the 4th Benelux Symposium on Information Theory (1983), 17-25.

Mau90. U. Maurer: Fast generation of secure RSA-moduli with almost maximal diversity. Advances in Cryptology – EUROCRYPT '89. Springer Lecture Notes in Computer Science 434 (1990), 636-647.

MBeu86. M. Beutelspacher: Kultivierung bei lebendigem Leib. Drumlin Verlag, Weingarten 1986.

McE78. R.J. McEliece: A public-key cryptosystem based on algebraic coding theory. JPL DSN Progress Report 42-44, pp. 114-116, Jan.-Feb. 1978.

MH78. R.C. Merkle, M.E. Hellman: Hiding information and signatures in trapdoor knapsacks. IEEE Trans. Inf. Theory, IT-24 (1978), 525-530.

MOV96. A. J. Menezes, P. C. Van Oorschot, S. A. Vanstone: Handbook of Applied Cryptography, CRC Press, 1996.

NIST91. National Institute of Standards and Technology (NIST): A Proposed Digital Signature Algorithm (4. Sept. 1991).

Omu90. J.K. Omura: Novel Applications of Cryptography in Digital Communications. IEEE Communications Magazine, May 1990, 21-29.

Per86. G. Perec: Anton Voyls Fortgang. Zweitausendeins, Frankfurt 1986.

QG90. J.-J., M., M., M. Quisquater, L., M., G., A., G., S. Guillou: How to explain Zero-Knowledge Protocols to Your Children. Advances in Cryptology – CRYPTO '89. Springer Lecture Notes in Computer Science 435 (1990), 628-631.

RE99. W. Rankl, W. Effing: Handbuch der Chipkarten. Aufbau – Funktiosnweise- Einsatz von Smart Cards. Hanser-Elektronik, 3. Auflage 1999.

RSA78. R. Rivest, A. Shamir, L. Adleman: A method for obtaining digital signatures and public key cryptosystems. Comm. ACM 21 (1978), 120-126.

Rue86. R. Rueppel: Analysis and design of Stream Ciphers. Springer-Verlag 1986.

Ruh87. Ch. Ruhland: Datenschutz in Kommunikationsnetzen. Datacom-Verlag Pulheim 1987.

Sal90. A. Salomaa: Public-Key Cryptography. Springer-Verlag, EATCS Monographs on Theoretical Computer Science Vol. 23, 1990.

Schn00. B. Schneier: Angewandte Kryptographie. Protokolle, Algorithmen und Sourcecode in C. Addison-Wesley, München, 2000.

Sha49. C.E. Shannon: Communication theory of secrecy systems. Bell. Sys. Tech. J. 30 (1949), 657-715.

Sha82. A. Shamir: A polynomial time algorithm for breaking the basic Merkle-Hellman cryptosystem. Proc. 23rd IEEE Symp. Found. Computer Sci. 142-152 (1982).

Sim79. G.J. Simmons: Cryptology: The mathematics of secure communication. The Mathematical Intelligencer 1 (1979), 233-246.

Sim92. G.J. Simmons: Contemporary Cryptology. The Science of Information Integrity. IEEE Press, New York 1992.

Sin00. S. Singh: Geheime Botschaften. dtv 2001.

Sin02. S: Singh: Codes. Hanser 2002.

Smi71. L. D. Smith: Cryptography. The Science of Secret Writing. Dover Publications, New York 1971.

SP89. J. Seberry, J. Pieprzyk: Cryptography. An Introduction to Computer Security. Prentice Hall 1989.

Sti95. D. R. Stinson: Cryptography. Theory and Practice. CRC Press, Boca Raton London Tokyo 1995

Str56. D.J. Struik: A concise History of Mathematics. Dover Publications, Inc. 1956.

WA75. H. Wußing, W. Arnold: Biographien bedeutender Mathematiker. Aulis Verlag Deubner & Co., Köln 1975.

Wel01. M. Welschenbach: Kryptographie in C und C++. Zahlentheoretische Grundlagen, Computer-Arithmetik mit großen Zahlen, kryptographische Tools. Springer-Verlag, Berlin, Heidelberg, 2. Auflage 2001.

Sachverzeichnis