Ein Verfahren zur Verschlüsselung von Nachrichten durch Ausnutzung der dynamischen Balancierung von Bäumen

Christian Martini

21. Januar 1998

Diplomarbeit nach einem Thema von Prof. Dr. Günter Hotz
in der Technischen Fakultät, Fachbereich 14 – Informatik,
der Universität des Saarlandes, Saarbrücken

Hiermit versichere ich an Eides Statt, daß ich diese Arbeit nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Saarbrücken, im Dezember 1997

Inhaltsverzeichnis

Abbildungsverzeichnis

Einleitung

Sicherheit im Sinne von Kommunikationssicherheit ist in unserer heutigen Zeit unabdingbar. Durch die immer weiter reichenden Ausbreitungen von Internet und WWW ist fast schon jeder von einer Kommunikation über öffentliche Netze betroffen.

Sollen dabei sicherheitsrelevante Daten oder einfach nur private Nachrichten, die sonst niemand erfahren soll, übertragen werden, muß man für eine entsprechende Geheimhaltung sorgen.

Eine gute Möglichkeit hierzu ist der Einsatz von kryptographischen Verfahren, die Nachrichten in der Art verschlüsseln, daß sie von keinem ungebetenen Dritten gelesen werden können. Nur der rechtmäßige Empfänger der Nachricht kann den Geheimtext wieder entschlüsseln und gelangt so an die Information.

Das hier vorgestellte kryptographische Verfahren beruht auf dem Einsatz von dynamisch balancierten Bäumen als Schlüssel für die Anwendung des Verschlüsselungsalgorithmus. Es ist ein symmetrisches Verschlüsselungsverfahren, bei dem zwei Kommunikationspartner jeweils über die selben Schlüssel zum Ver- und Entschlüsseln verfügen.

Das Haupteinsatzgebiet ist die Verschlüsselung von natürtlicher Sprache bei der Übertragung von sogenannten E-Mails. An dieser Vorstellung ist auch die Implementierung des C-Programms, das dieser Arbeit zugrunde liegt, ausgerichtet. Eine Anbindung des Verfahrens an ein gängiges Mailprogramm oder an Netscape ist denkbar und auch wünschenswert.

In Kapitel 1 werden die Grundlagen der Kryptographie und der Anwendung kryptographischer Verfahren aufgezeigt. Es wird beschrieben, wie Ver- und Entschlüsselung funktionieren. Auf Basis einer Unterscheidung in symmetrische und asymmetrische Verfahren werden Kryptosysteme vorgestellt. Anschließend erfolgt eine Beschreibung von Kryptoanalyse, die sich mit der Sicherheit bzw. Unsicherheit von kryptographischen Algorithmen und Verfahren beschäftigt.

Kapitel 2 stellt die informationstheoretischen Voraussetzungen für das Verfahren dar und beschreibt die Funktionsweise von diesem Ansatz aus.

Kapitel 3 beruht in erster Linie auf der Implementierung des Verfahrens, d. h. auf der Umsetzung des Algorithmus in ein Kryptosystem. Die Kommunikationsabläufe werden im Einzelnen beschrieben und es wird ein konkretes Kommunikationsprotokoll für dieses Verfahren aufgestellt. Aus der Beschreibung der Implementierung der dynamisch balancierten Suchbäume lassen sich dann die abschliessenden Betrachtungen über die Sicherheit und die Einsetzbarkeit des Verfahrens direkt ableiten.

Danksagung

An dieser Stelle möchte ich mich bei Herrn Professor Dr. Hotz für die Vergabe der Diplomarbeit und die für mich sehr interessante Themenstellung herzlich bedanken.

Allen Kommilitonen und Mitarbeitern an diesem Lehrstuhl danke ich für das hervorragende Arbeitsklima.

Kapitel 1

Die Kryptographie

1.1 Aufgaben der Kryptographie

Der grundlegende Gedanke der Kryptographie besteht darin, die Kommunikation zwischen einem Sender und einem Empfänger sicher zu gestalten.

Unter Sicherheit versteht man dabei die Vermeidung oder Verhinderung von Störungen, denen jede Kommunikation unterliegt. Der Nachrichtenaustausch findet i. a. über ein Übertragungsmedium, einen sogenannten Kanal, statt. Die Beeinträchtigungen lassen sich einteilen in

- zufällige Störungen
- systematische physikalische Störungen
- passive Störungen
- aktive Störungen

Physikalische Beeinträchtigungen, wie sie die beiden ersten Punkte nennen, werden in der Regel durch die Art des Übertragungsmediums bedingt.

Passiv kann ein Nachrichtenaustausch dadurch beeinflußt werden, daß ein weder vom Sender noch vom Empfänger bemerkter Zugriff auf die Daten erfolgt. Wird die Nachricht von einem Dritten verändert oder verfälscht, liegt eine aktive Störung der Kommunikation vor.

Aktiven Schutz und Sicherheit gegen ein solches ungewolltes Eingreifen Dritter in einen Kommunikationsablauf soll die Kryptographie leisten.

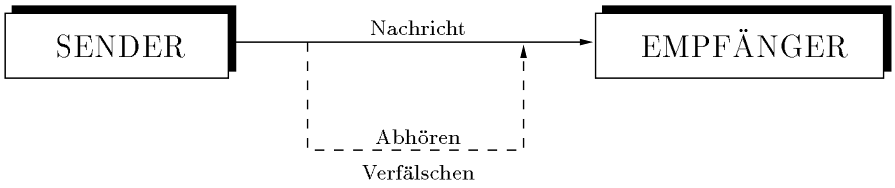


Abbildung 1.1: Beeinflussung eines Nachrichtenaustauschs

Eine einfache Methode hierzu ist das Chiffrieren einer Nachricht. Dabei wird vom Sender die Nachricht, die übertragen werden soll, verschlüsselt und so an den Empfänger gesandt. Dieser muß die Nachricht wieder entschlüsseln, um auf deren Inhalt Zugriff zu haben. Ein ungebetener Dritter kann somit höchstens mehr oder weniger verständliche, verschlüsselte Texte abhören oder abfangen.

Unter Chiffrieren versteht man also die Umwandlung eines Klartextes in einen sogenannten Geheimtext unter Zuhilfenahme eines Verschlüsselungsalgorithmus.

Sei nun M die Menge aller Klartexte und C die Menge aller Geheimtexte. Der Verschlüsselungsalgorithmus wird mit f bezeichnet. Der aktuell von Sender und Empfänger benutzte Schlüssel k führt zu einer Spezialisierung f_k vom allgemeinen Algorithmus f. Die Abbildung

$$f_k : M \longrightarrow C$$

wird als Transformation bezeichnet. Sei F die Menge aller Transformationen. Die Abbildung ist umkehrbar, d. h. es gilt

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

Es kann also jeder Geheimtext $f_k(m)$ durch Anwendung der Umkehrung f_k^{-1} in den entsprechende Klartext entschlüsselt werden.

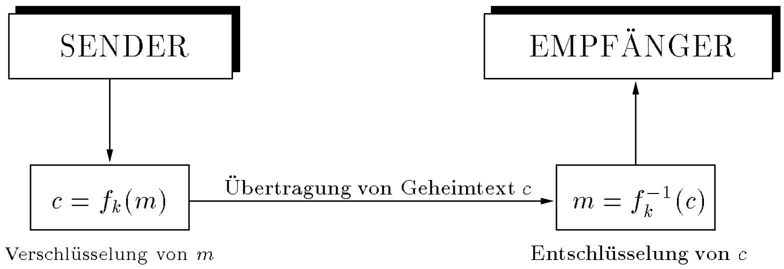


Abbildung 1.2: Der in einen Geheimtext c transformierte Klartext m wird übertragen

Hieraus ergibt sich nach Shannon die Definition eines Chiffriersystems [SHA]:
Ein symmetrisches Chriffriersystem S besteht aus einer Menge M von Klartexten, einer Menge C von Geheimtexten und einer Menge F von umkehrbaren Transformationen von M in C.

$$S = (M, C, F)$$

Ein solches Chiffriersystem zu konstruieren sollte nicht schwer fallen. Doch was nützt das schönste Chiffriersystem, wenn es nicht unseren Ansprüchen genügt. Gibt man sich bescheiden und stellt nur einen einzigen Anspruch an ein solches System, so sollte man fordern, daß es perfekt ist.

Ein Chiffriersystem wird als perfekt bezeichnet, wenn es die folgenden zwei Kriterien erfüllt [BEU]:

Kriterium 1
Jeder beliebige Klartext $m \in M$ kann mit einem zu S gehörenden Schüssel in jeden beliebigen Geheimtext $c \in C$ verschlüsselt werden.

Kriterium 2
Sei S ein System, in dem alle Schlüssel mit der gleichen Wahrscheinlichkeit vorkommen. Ferner gilt:

$$\mid F \mid = \mid C \mid = \mid M \mid$$

Gibt es zu jedem Klartext $m \in M$ genau einen Schlüssel aus S, der m in einen Geheimtext $c \in C$ transformiert, so ist S perfekt.

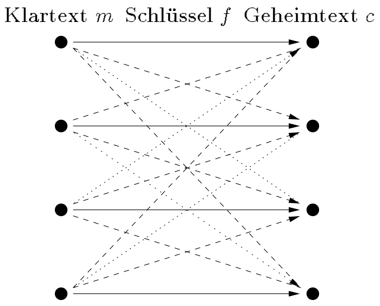


Abbildung 1.3: Jeder Geheimtext c läßt sich mit genau einem Schlüssel k aus dem Klartext m herleiten $\Longrightarrow \mid M \mid = \mid F \mid = \mid C \mid$

1.2 Bisherige kryptographische Verfahren

Kryptographische Verfahren lassen sich grob in zwei Klassen einteilen. Zum Einen Verfahren, bei denen zum Verschlüsseln auf Senderseite und zum Entschlüsseln auf Empfängerseite die gleichen Schlüssel verwendet werden. Beide Kommunikationspartner müssen also über einen identischen Schlüssel verfügen. Zum Zweiten die sogenannten asymetrischen Verfahren mit verschiedenen Schlüsseln auf Sender- und Empfängerseite.

1.2.1 Symmetrische Verfahren

Dieser Abschnitt soll einen kurzen Überblick über Verfahren der Kryptographie liefern, wie sie im geschichtlichen Verlauf entstanden sind. Auf eine genaue Beschreibung der einzelnen Algorithmen wird dabei verzichtet, da dies nicht Gegenstand dieser Arbeit sein soll. (Siehe zum Beispiel [BET], [BEU], [SCH])

Zuerst werden historische Verfahren vorgestellt. Im nächsten Unterabschnitt 1.2.2 wird auf moderne Kryptosysteme eingegangen, die auf Komplexitätsbetrachtungen und Abwägungen der Kryptoanalyse, die im Abschnitt 1.3 noch beschrieben wird, aufgebaut sind.

Ein Transpositionsalgorithmus beruht einzig und allein auf der Tatsache, die durch einen Klartext gegebenen Buchstaben zu permutieren, also ihre Reihenfolge zu ändern. Ein Beispiel für die Anwendung dieses Algorithmus ist der **Skytale von Sparta**. Aus heutiger Sicht läßt sich dieses Verfahren als äußerst unsicher bezeichnen.

Der Schritt zum Substitutionsalgorithmus wurde durch ein Verfahren, das nach **J. Caesar** benannt ist, vollzogen. Grundgedanke dieser Verschlüsselung ist die Ersetzung einzelner Buchstaben des Klartextes durch andere Buchstaben des verwendeten Alphabets. Man identifiziert die Buchstaben mit den Zahlen 0, ..., 25 und berechnet den verschlüsselten Buchstaben als $f(i) = i + s$, mit i entspricht dem zu verschlüsselnden Buchstaben, s gibt die Weite der Verschiebung im Alphabet an. s ist hier sozusagen der Schlüssel. Durch einfachste Häufigkeitsbetrachtungen (siehe Tabelle) läßt sich diese Art der Verschlüsselung bereits knacken.

Im Vergleich zu den beiden ersten Verfahren benutzt **de Vigenère** in seinem Algorithmus eine fest vereinbarte Schlüsselfolge, die zum Klartext addiert wird. Dabei werden die Buchstaben des Alphabets wieder mit den Zahlen 0, ..., 25 identifiziert. Durch die begrenzte Länge der Schlüsselfolge und das damit verbundene periodische Auftreten kann auch dieses Verfahren keiner Kryptoanalyse basierend auf Häufigkeitsbetrachtungen standhalten.

Vernam erweiterte das Vignère-Verfahren dahingehend, daß er keine feste Schlüsselfolge wählte, sondern eine zufällige, die mindestens solange war wie der eigentliche Klartext. Der Schlüssel muß nach den Erkenntnissen von Shanon jedoch

eine reine Zufallsfolge maximaler Entropie sein. Eine scheinbare Zufallsfolge, zum Beispiel ausgewählte Literaturstellen, wie sie früher benutzt wurden, reicht nicht aus.
Das Verfahren wird auch als **One-time-pad** bezeichnet, nach seiner Verwendung in der Telegraphie. Die binäre Schlüsselfolge wurde auf ein Papierstreifen geschrieben, der von Sender und Empfänger nur ein Mal benutzt wurde.

1.2.2 Asymmetrische Verfahren

Basis für die Kommunikation nach einem symmetrischen Verfahren ist das Vorhandensein eines identischen Schlüssels auf Sender- und Empfängerseite, d. h. der Sender verschlüsselt eine Nachricht m mit dem Schlüssel f und der Empfänger benutzt eben diesen ihm bekannten Schlüssel f um die Nachricht zu entschlüsseln. Da dieser Vorgang auch in der anderen Richtung ablaufen kann, kann also prinzipiel jeder der verschlüsseln kann auch entschlüsseln. Diese Tatsache wird allgemein als Vorteil anerkannt. Sie setzt jedoch voraus, daß beide Partner über einen identischen Schlüssel verfügen, der natürlich zu diesem Zweck erst ausgetauscht werden muß.

Asymmetrische Verfahren, auch **Public-Key-Verfahren** genannt, verzichten auf einen identischen Schlüssel auf Sender- und Empfängerseite und ersparen sich somit einen aufwendigen Schlüsselaustausch zwischen den Kommunikationspartnern. Der Schlüsselverwaltungsproblem der symmetrischen Verfahren scheint damit behoben zu sein, wie es auch Diffie und Hellman in ihrer Arbeit erörtern. Desweiteren lösen diese Verfahren auch das Signatureproblem, das bei den bisherigen Verfahren bestand.

In einem Public-Key-Kryptosystem verfügt jeder Kommunikationspartner sowohl über einen öffentlichen Schlüssel, auf den jeder zugreifen kann, als auch über einen privaten Schlüssel, den nur er kennt. Beim Kommunikationsablauf dient dabei der öffentliche Schlüssel E_T dem Verschlüsseln der Nachricht und der geheime Schlüssel D_T des Teilnehmers T dem Dekodieren des Geheimtextes. Die Schlüssel erfüllen die Forderung, daß von E_T nicht auf D_T geschlossen werden kann, d. h. wenn ein Sender eine Nachricht mit Hilfe des Schlüssels E_T verschlüsseln kann, so kann er sie noch lange nicht entschlüsseln. Dadurch ergibt sich der Vorteil, daß alle Schlüssel zum Verschlüsseln öffentlich bekannt sein können.

Will nun ein Sender S eine Nachricht an einen Empfänger E senden und diese geheim halten, so wählt er den öffentlich bekannten Schlüssel E_E des Empfängers, kodiert damit die Nachricht m und sendet den Geheimtext an E. Der Empfänger muß jetzt nicht lange darauf achten, von wem die Nachricht kommt und, wie bei symmetrischen Verfahren üblich, mit welchem Schlüssel sie verschlüsselt wurde.

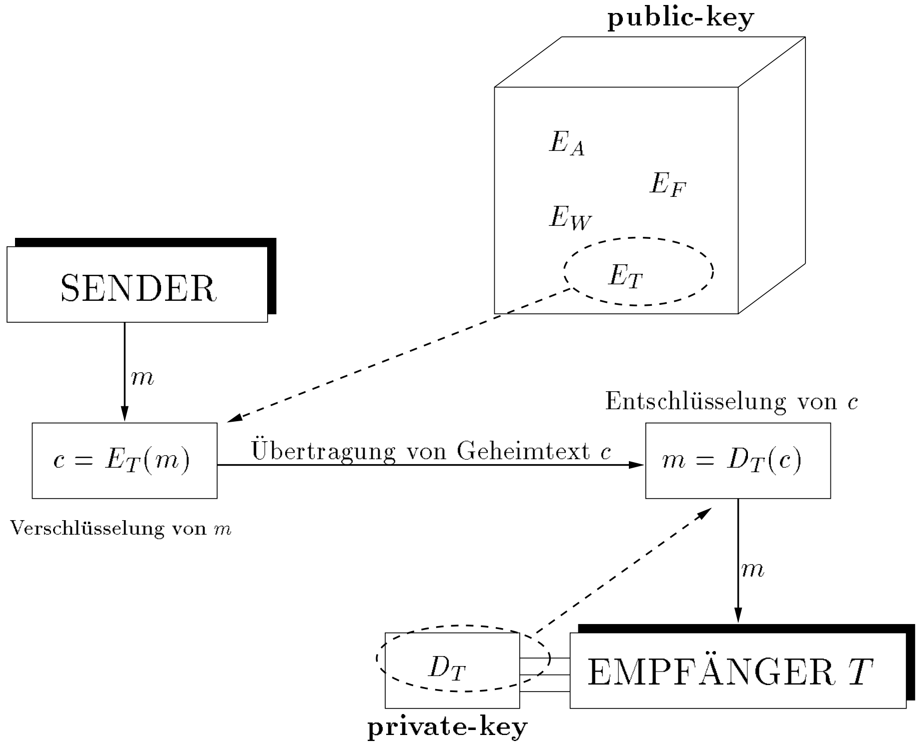


Abbildung 1.4: Asymmetrische Chiffrierung

Stattdessen weiß er, daß die Nachricht, da sie ja für ihn bestimmt war, mit seinem öffentlichen Schlüssel E_E kodiert wurde. Daher benutzt er einfach seinen privaten Schlüssel D_E, um den Geheimtext zu entschlüsseln. Der Klartext der Nachricht ergibt sich aus

$$D_E(E_E(m)) \quad = \quad m.$$

Ein Kryptosystem, das nach diesem Funktionsprinzip gestaltet ist, heißt asymmetrisches Verschlüsselungssystem. Eine Nachricht, die mit Hilfe des Schlüssels E_E kodiert wurde, kann mit D_E einfach wieder entschlüsselt werden und zwar nur vom rechtmäßigen Empfänger E und von keinem anderen, da kein anderer den privaten und somit geheimen Schlüssel D_E kennt.

Der Hauptvorteil eines asymmetrischen Verschlüsselungssystems besteht im Wegfall des Schlüsselaustauschs, da ja beide Kommunikationspartner nicht über identische Schlüssel verfügen müssen.Es kann also eine sogenannte spontane oder offene Kommunikaiton stattfinden, ohne daß diese von beiden Teilnehmern vorher vereinbart wurde, da jeder, der einen Public-Key festlegt für viele Partner zur Kommunikation bereitsteht, ohne einen einzelnen speziell auszuwählen.

Aus der hier beschriebenen Tatsache rührt der nächste Vorteil: Jeder Teilnehmer muß nur über zwei Schlüssel, einen öffentlichen und einen privaten, verfügen, wohingegen bei einem symmetrischen Kryptosystem die Teilnehmer paarweise über einen identischen Schlüssel verfügen müssen, d. h. für jeden Teilnehmer, mit dem man kommunizieren will, braucht man einen extra Schlüssel. Mit jedem neuen Teilnehmer steigt die Gesamtzahl der verwendeten Schlüssel somit quadratisch an. Hinzu kommt der Verwaltungsaufwand, den der neue Teilnehmer betreiben muß, da er sich ja an jeden potentiellen Kommunikationspartner wenden muß, um mit ihm die Schlüssel auszutauschen, die er außerdem auch alle verwalten muß.

Nachteilig am Public-Key-Verfahren ist die mangelnde Schnelligkeit, die ein realistisches, sicheres Verfahren mit sich bringt. Exemplarisch soll hier der RSA-Algorithmus vorgestellt werden. Zu beachten bleibt auch, daß ein Vortäuschen eines Schlüssels durch eine entsprechende Schlüsselauthentifikation gewährt sein muß, d.h. daß kein Teilnehmer A vorgeben kann, Teilnehmer B zu sein und einen Schlüssel E_B veröffentlicht. Geschehen könnte dies relativ leicht, da die Public-Key, wie der Name schon sagt, öffentlich abgelegt sein und für jeden Kommunikationspartner frei zugänglich sein müssen. Eine Möglichkeit der Sicherung bildet in diesem Fall das Unterschreiben des öffentlichen Schlüssels mit dem privaten Schlüssel, um so ein Austauschen durch Dritte zu erschweren. Unmöglich wird dieses Unterfangen dadurch natürlich nicht.

In der Regel der Fälle dürften Public-Key-Verfahren wohl nicht zur Verschlüsselung von Nachrichten eingesetzt werden, da dies die allgemeinen Anforderungen an die Geschwindigkeit von Kryptographieverfahren nicht erlauben. Häufiger trifft man Public-Key-Verfahren in Kombination mit symmetrischen Verfahren in **hybriden Kryptosystem** an. Der asymmetrische Algorithmus wird ausschließlich dazu verwandt, einen Sitzungsschlüssel für ein symmetrisches Verfahren zwischen zwei Kommunikationspartnern zu übertragen. Sitzungsschlüssel bezeichnet hier den Schlüssel, der nur für eine Kommunikationssitzung benutzt wird. Für die nächste wird wiederum ein neuer Schlüssel verwendet.

Benutzen die Partner S als Sender und E als Empfänger ein hybrides Kryptosystem, so läuft die Kommunikation in den folgenden Schritten ab:

1. S verschlüsselt den Sitzungsschlüssel K mit Hilfe des öffentlichen Schlüssels von E
$$E_E(K) = c$$
und sendet diesen Geheimtext an E.

2. E entschlüsselt die Nachricht einfach mit seinem privaten Schlüssel und erhält so den gewünschten Schlüssel
$$D_E(c) = K.$$

3. Beide Kommunikationspartner können jetzt den Schlüssel K auf ein symmetrisches Verfahren anwenden, um ihre Nachrichten geheim auszutauschen.

1.2.3 Data Encryption Standard

Der Data Encryption Standard oder kurz **DES** soll hier als heute gängig eingesetztes symmetrisches Verfahren vorgestellt werden. Entwickelt wurde das Verfahren bereits in den siebziger Jahren von der Firma IBM nach einer Ausschreibung des National Bureau of Standards (NBS, heute NIST) zur Schaffung eines einheitlichen, standardisierten Kryptoverfahrens zur sicheren Speicherung und Übermittlung von Daten.

DES als symmetrisches Verfahren benutzt zum Ver- und Entschlüsseln den selben 56 Bit Schlüssel. Dabei werden durch eine Blockchiffrierung jeweils 64 Bit Klartext in 64 Bit Geheimtext und umgekehrt chiffriert.

Im Grunde genommen beruht DES nur auf den beiden Methoden der Konfusion und Diffusion, die in einer mehrstufigen Folge auf den Eingabeblock angewandt werden. Konfusion, z. B. durch eine einfache Substitution realisiert, soll den Zusammenhang zwischen Klartext und Geheimtext verbergen, um statistische Analyseansätze zu vereiteln. Diffusion stellt ein Mittel dar, im Klartext vorhandene Redundanzen nicht auf den Geheimtext zu übertragen. Durch Permutationen, wie sie von Transpositionsalgorithmen umgesetzt werden, wird genau dieser Effekt erziehlt. DES bezeichnet eine solche Kombination, d. h. eine Permutation nach einer Substitution, die auf eine Eingabe angewandt wird als Runde. Sechzehn dieser Runden werden auf den ursprünglichen Klartextblock angewandt.

Der 64 Bit Block als Klartexteingabe wird einer Eingangspermutation unterzogen und dann in zwei 32 Bit Hälften unterteilt. Nun folgen die oben bereits erwähnten sechzehn Runden. In jeder werden dabei die Bits des Schlüssels verschoben und ein 48 Bit Schlüssel ausgewählt und rotiert, der dann mittels XOR mit der auf 48 Bit verbreiterten rechten Seite der Eingabe kombiniert wird. Die Erweiterung von 32 auf 48 Bit erfolgt mit einer Expansionspermutation. Durch acht sogenannte S-Boxen werden aus dem Ergebnis 32 neue Bits gewonnen, die wiederum permutiert werden. Diese vier Operationen werden unter der Bezeichnung $Funktion f$ zusammengefaßt. Die 32 Bit der Ausgabe von $Funktion f$ werden mittels XOR mit der linken Seite der Eingabe kombiniert und bilden die rechte Hälfte für die nächste Runde. Die alte rechte Hälfte wird zur neuen linken Hälfte.

Seien also L_{i-1} und R_{i-1} die linke und rechte Hälfte der Eingabe der i-ten Runde. K_i stellt den anzuwendenden 48 Bit Schlüssel dar. Das Ergebnis der i-ten Runde, also die beiden Hälften, erhält man wie folgt:

$$L_i = R_{i-1}, \quad R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$$

Die beiden Hälften der sechzehnten Runde werden zusammengefügt und einer Schlußpermutation unterzogen, die invers zur Eingangspermutation ist.

Für die genauen Funktionsweisen von Eingangspermutation, Schlüsseltransformation, Expansionspermutation, S-Box-Substitution, P-Box-Permutation und Schlußpermutation sei an dieser Stelle auf die Literatur verwiesen, da sie nicht zum grundsätzlichen Verständnis des Verfahrens beitragen, sondern nur die Kryptanalyse erschweren (siehe [SCH]).

In Bezug auf die Kryptanalyse kann man wohl sagen, daß DES einen hohen Sicherheitsstandard erfüllt. So konnte es sich z. B. gegen die differentielle Kryptanalyse von Biham und Shamir aus dem Jahr 1990 erfolgreich zur Wehr setzen, wobei die S-Boxen in dem in den siebziger Jahren entwickelten Verfahren auf diesen recht neuen Angriff abgestimmt zu sein schienen. Bei diesem Verfahren untersucht man Klartexte mit bestimmten Differenzen und die Differenzen die dadurch bei den Geheimtexten resultieren (siehe [SCH]).

Erfolgreicher war die lineare Kryptanalyse von Matsui. Durch lineare Approximation, hier durch XOR-Verknüpfung von Klartext- und Geheimtextbits, erhält man Informationen über den Schlüssel, die mit einer gewissen Wahrscheinlichkeit eintreffen. Dieses Verfahren kann mit durchschnittlich 2^{43} Klartexten den Schlüssel zu DES mit sechzehn Runden bestimmen (siehe [SCH]).

1.2.4 RSA

Rivest, Shamir und Adleman stellten mit dem nach ihnen benannten RSA-Verfahren das erste vollständige Public-Key-System auf, mit dem man Verschlüsseln und Signaturen erstellen konnte. Bei dem wohl bekanntesten Vertreter der asymmetrischen Verfahren handelt es sich um ein Kryptosystem, dessen Sicherheit auf der Schwierigkeit der Faktorisierung großer Zahlen beruht.

Die Bestimmung des öffentlichen und privaten Schlüssels ergibt sich wie folgt: Man wählt zufällig zwei große Primzahlen p und q und bildet das Produkt

$$n = pq.$$

Man bestimmt nun den Chiffrierschlüssel e so, daß e teilerfremd ist zu $(p-1)(q-1)$ (die Anzahl der zu n teilerfremden natürlichen Zahlen nach Euler). Den zweiten Teil des Schlüssels erhält man, indem man die Gleichung

$$ed \bmod ((p-1)(q-1)) \equiv 1$$

löst und nach d auflöst.

Die beiden Zahlen e und n bilden den öffentlichen Teil des Schlüssels, d den privaten Teil. p und q werden nicht benötigt und sollten zur Sicherheit vernichtet werden, da mit ihrer Hilfe die Schlüssel berechnet werden können.

Zur Verschlüsselung der Nachricht m wird diese so in numerische Blöcke unterteilt, daß jeder Block $m_i \leq n$ ist. Ist n etwa eine 512-Bit Zahl, wie es heute üblich ist, so wird der Text einer Nachricht in Blöcke zu 64 Zeichen aufgeteilt, vorausgesetzt ein Zeichen benötigt zur Darstellung 8 Bit. Der Geheimtext jedes einzelnen Blocks ergibt sich dann als

$$c_i \quad \equiv \quad m_i^e \bmod n.$$

Die Geheimnachricht setzt sich aus den einzelnen c_i zusammen. c_i läßt sich vom Empfänger wieder entschlüsseln durch die Anwendung des privaten Schlüssels d auf den einzelnen Chiffrierblöcke

$$m_i \quad \equiv \quad c_i^d \bmod n.$$

Die Sicherheit bzw. Unsicherheit von RSA besteht in der Veröffentlichung von e und n und der Frage, ob man von diesen beiden Größen auf den privaten Schlüssel d schließen kann. Will man d berechnen, so muß man die Zahl $\phi(n) = (p-1)(q-1)$ kennen. Diese kennt man natürlich, wenn man $n = pq$ in die beiden Primfaktoren zerlegen kann. Das Problem d zu bestimmen besteht also darin, n zu faktorisieren oder $\phi(n)$ zu berechnen. Mit der Schwierigkeit, große Zahlen zu faktorisieren hängt also auch direkt die Sicherheit von RSA ab.

Um die Sicherheit von RSA zu erhöhen kann man natürlich n vergrößern und noch weit über die heute üblichen 200 Dezimalstellen hinausgehen, doch leidet darunter auch die Geschwindigkeit des Verfahrens. So sind Hardware-Implementierungen von RSA etwa um Faktor 1000 langsamer als Hardware-Implementierungen des im vorangegangenen Abschnitt vorgestellten DES-Verfahrens. Bei Software-Implementierungen beider Verfahren geht dieser Vorteil von DES etwas verloren, doch besteht immer noch ein Geschwindigkeitsunterschied von Faktor 100 (siehe [SCH]).

1.3 Die Kryptoanalyse

Das Gebiet der Kryptoanalyse ist der Versuch, aus einem Geheimtext ohne den entsprechenden Schlüssel den zugehörigen Klartext zu erstellen. Der Erfolg besteht also in der Ermittlung des Klartextes oder des Schlüssels. Die Kryptoanalyse versucht bestehende Verschlüsselungsverfahren zu brechen und dabei ihre

Schwachstellen aufzuweisen. Wobei die Philosophie der moderenen Kryptoanalyse im Grunde dem Prinzip von Kerckhoffs folgt, das dieser bereits im Jahr 1883 aufstellte. Nach diesem Prinzip darf die Sicherheit eines Kryptosystems nicht von der Geheimhaltung des Algorithmus abhängen. Die Sicherheit gründet sich nur auf die Geheimhaltung des Schlüssels. Selbst nach Bekanntgabe eines Verschlüsselungsalgorithmus und Diskussion durch Kryptoanalytiker sollte ein Verfahren immer noch sicher sein.

Geht man davon aus, daß dem Kryptoanalytiker auf jeden Fall der Verschlüsselungsalgorithmus bekannt ist, so unterscheidet man, jenachdem, wieviele Fakten zusätzlich bekannt sind, die Attacken:

Ciphertext-only attack
Der Kryptoanalytiker ist im Besitz mehrerer beliebig großer Stücke Geheimtext die mit dem selben Verfahren chiffriert wurden, was auch sehr realistisch erscheint, und versucht daraus den entsprechenden Klartext zu gewinnen bzw. sogar Rückschlüsse auf den Schlüssel zu ziehen, um damit weitere Geheimtexte entschlüsseln zu können.

Gegeben: $c_1 = E_T(m_1), \quad c_2 = E_T(m_2), \quad ..., \quad c_n = E_T(m_n)$
Gesucht: $m_1, m_2, \quad ..., m_n$ oder ein Algorithmus, der von c_i auf m_i schließen läßt, mit $1 \leq i \leq n$

Known plaintext attack
Der Kryptoanalytiker verfügt über zusammengehörige Klar- und Geheimtextteile. Diese Annahme scheint realistisch, da ein Gegener oft weiß, um was es in der entsprechenden Nachricht gehen könnte oder ihm sind zumindest Schlagworte bekannt. Der Kryptoanalytiker versucht den Schlüssel zu finden bzw. einen Algorithmus, mit Hilfe dessen sich aus weiteren Geheimtexten die entsprechenden Klartexte bestimmen lassen.

Gegeben: $m_1, c_1 = E_T(m_1), \quad m_2, c_2 = E_T(m_2), \quad ..., \quad m_n, c_n = E_T(m_n)$
Gesucht: Schlüssel oder ein Algorithmus, der von c_i auf m_i schließen läßt, mit $1 \leq i \leq n$

Chosen plaintext attack
Der Kryptoanalytiker verfügt nicht nur über Geheimtexte und die dazugehörenden Klartexte wie beim oben beschriebenen Angriff. Zusätzlich ist er in der Lage, selbstgewählte Textstücke zu verschlüsseln, um dadurch Rückschlüsse auf den verwendeten Schlüssel zu ziehen oder einen Algorithmus zu bestimmen, mit dem

weitere Nachrichten entschlüsselt werden könnten. Der Gegner könnte darüber hinaus auch Nachrichten erzeugen, einen anderen Sender vortäuschen und so beim Empfänger eine von ihm gewollte Reaktion hervorrufen.

Gegeben: $m_1, c_1 = E_T(m_1), \quad m_2, c_2 = E_T(m_2), \quad ..., \quad m_n, c_n = E_T(m_n)$
mit $m_1, m_2, ..., m_n$ vom Kryptoanalytiker selbst gewählt
Gesucht: Schlüssel oder ein Algorithmus, der von c_i auf m_i schließen läßt, mit $1 \leq i \leq n$

Als Fortführung oder spezielle Ausprägung dieses Angriffs sei die **Adaptive-chosen-plaintext attack** genannt, bei der davon ausgegangen wird, daß der Kryptoanalytiker aus seinen bisherigen Verschlüsselungsversuchen lernt, d. h. er verschlüsselt zuerst kleine Textteile und baut auf seinen Erkenntnissen hieraus seine weiteren Angriffe auf.

Eine monoalphabetische Chiffrierung läßt sich so zum Beispiel recht leicht mit der ersten Angriffsvariante knacken. Betrachtet man sich die Struktur der deutschen Sprache, so wird man feststellen, daß die am häufigsten benutzten Buchstaben bereits drei Viertel eines Textes ausmachen. Hat der Gegener also den Geheimtext komplett oder in großen Stücken abgefangen, läßt dies einen sehr guten Rückschluß auf den Inhalt zu.

Buchstabe	Häufigkeit in %	Buchstabe	Häufigkeit in %
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

Abbildung 1.5: Häufigkeit der Buchstaben der deutschen Sprache

Eine Gruppierung der Buchstaben nach ihren Häufigkeiten führt zur folgenden Viereinteilung:

Gruppe	Pronzentualer Anteil
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

Abbildung 1.6: Gruppierung von Buchstaben nach ihren Häufigkeiten

Diese Betrachtung eines Geheimtextes setzt natürlich voraus, daß der Gegner weiß, daß ein monoalphabetisches Verfahren zur Verschlüsselung verwendet wurde.

Um eine Kryptoanalyse erfolgreich zu nennen, braucht man nicht unbedingt den Schlüssel herauszufinden. Man muß sozusagen nur Teile der Geheimhaltung lüften. Knudsen unterscheit deshalb vier Grade von gelungenen Attacken [KNU]:

1. **Vollständiges Aufbrechen** heißt, daß ein Schlüssel K gefunden wird, für den gilt: $D_K(c) = m$. Der Geheimtext kann also vollkommen entschlüsselt werden und man erhält einen Schlüssel, um weitere Geheimtexte, die nach gleichem Verfahren und mit gleichem Schlüssel kodiert wurden zu entschlüsseln.

2. **Globale Deduktion** bedeutet, daß der Kryptoanalytiker zwar nicht den eigentlichen Schlüssel K findet, aber einen Algorithmus A, der die gleiche Arbeit leistet, also ebenfalls den Klartext m liefert, wenn man ihn auf c anwendet: $A(c) = m$.

3. **Punktuelle oder lokale Deduktion** beschränkt das gelingen der Kryptoanalyse auf den einen vorliegenden Geheimtext. Nur der entsprechende Klartext kann gefunden werden, nicht aber der verwendete Schlüssel K.

4. **Informationsdeduktion** verlangt nur, daß Teile des Schlüssels oder Teile des Klartextes gefunden werden können.

Der betriebene Aufwand, um die verschiedenen Angriffe durchzuführen, wird in Form von drei unterschiedlichen Komplexitätskriterien gemessen:

1. Die **Datenkomplexität** macht eine Angabe über die Menge an Eingabedaten, die beim Angriff benötigt wird.

2. Die **Berechnungskomplexität** ist ein Maß für den Rechenaufwand, d. h. für die zur Durchführung des Angriffs erforderliche Zeit.

3. Die **Speicheranforderung** gibt an, wieviel Speicher für den Angriff notwendig ist.

Als uneingeschränkt sicher kann man laut [SCH] einen Algorithmus erst dann bezeichnen, wenn er selbst bei unbegrenzten Ressourcen, d. h. bei beliebig viel Geheimtext nicht zu knacken ist. Nach dieser Voraussetztung wäre nur das Verfahren des One-time-pad sicher, bei allen anderen Verfahren könnte man theoretisch einfach alle möglichen Schlüssel ausprobieren, bis ein Known-plaintext-Angriff zum Erfolg führt. Der hier beschriebene Angriff wird auch als Brute-Force-Angriff bezeichnet. Dabei rückt jedoch die Berechenbarkeit bei entsprechender Schlüsselgröße und heute gegebenen Rechnerkapazitäten in weite Ferne. So sind zum Beispiel zur Brechnung eines 128-bit-Schlüssels auch 2^{128} Rechenoperationen notwendig. Die Anwendbarkeit des entsprechenden Verfahrens läßt natürlich mit wachsendem Schlüssel nach. Man kann jedoch an dem in dieser Arbeit beschriebenen Verfahren sehen, daß selbst bei im Verhältnis zu 128 Bit riesigen Schlüsseln das Verfahren ohne weiteres leicht anwendbar bleibt.

1.4 Authentifikation

Die Sicherheit einer Nachrichtenübertragung besteht nicht nur darin, sich gegen passive Störungen zu schützen. Über das reine Mitlesen hinaus gilt es vielmehr, wie bereits am Anfang dieses Kapitels erwähnt, die Möglichkeit des aktiven Beeinflussens, d. h. eine Nachricht wird aktiv verändert oder gar komplett durch eine andere ersetzt, zu verhindern.

Aus dieser Tatsache rührt die Forderung der Kryptologie nach Authentizität der Daten und nicht nur nach ihrer Geheimhaltung. Die Frage nach der Datensicherheit unterscheidet drei Formen:

1. Die **Nachrichtenintegrität** bezeichnet die Korrektheit der Nachrichtenübermittlung in Hinblick auf Veränderungen oder Verfälschungen.

2. Die **Nachrichtenauthentifikation** sorgt für eine Überprüfung, ob die eingegangene Nachricht wirklich vom angegebenen Absender stammt.

3. Die **Benutzerauthentifikation** liefert dem Empfänger die Möglichkeit zu prüfen, ob er wirklich mit dem Sender kommuniziert, mit dem er glaubt zu kommunizieren.

Nachrichtenintegrität und -authentifikation verlangen danach, daß eine Nachricht zusätzliche Informationen zum eigentlichen Nachrichtentext beinhaltet, um zu erkennen, daß eine Mitteilung unverändert und unverfälscht den Empfänger erreicht hat. Der eigentlichen Nachricht wird aus diesem Grund ein sogenannter **Message-Authentification-Code** hinzugefügt.

Der Sender schickt dem Empfänger die Nachricht m und zusätzlich einen Message-Authentification-Code, der mit Hilfe eines Schlüssels k und einem Algorithmus A berechnet wurde.

$$MAC \quad = \quad A_k(m)$$

Zur reinen Sicherung der Nachricht kann diese zusammen mit dem Message-Authentification-Code sogar unverschlüsselt gesendet werden, was jedoch nicht die Regel sein dürfte. In diesem Fall dient der MAC als elektronische Unterschrift. Die Informationen, die in der Nachricht erhalten sind, können jedoch von einem beliebigen Dritten gelesen werden.

Der Empfänger vollzieht zur Prüfung der Nachrichtenintegrität die gleichen Schritte, die zuvor der Sender durchlaufen hat. Er setzt also den ihm bekannten Algorithmus A unter Verwendung des Schlüssels k auf die empfangene Nachricht m an und überprüft einfach, ob er den gesendeten Message-Authentification-Code erhält.

$$A_k(m) \quad = \quad MAC \quad ?$$

Ist dies nicht der Fall, so weiß der Empfänger, daß die Nachricht entweder nicht vom angegebenen Sender stammt, oder daß sie verändert wurde. Hat der Empfänger einen solchen Sachverhalt festgestellt, so muß der Sender die Nachricht erneut verschicken. Eine Rekonstruktion der ursprünglichen Nachricht ist nicht möglich.

Wurde die Nachrichtenintegrität erfolgreich geprüft, geht man gleichzeitig von der Authenzität der Nachricht aus, da nur der entsprechende Sender den empfangenen Message-Authentification-Code erzeugen konnte.

Der Algorithmus, der den Message-Authentification-Code erzeugt, sollte dabei nach Diffie und Hellman die folgenden Bedingungen erfüllen [DIF]:

1. Der Message-Authentification-Code sollte „leicht“ zu berechnen sein.

2. Es sollte „praktisch unmöglich“ sein, also mit heute gegebenen Mitteln in annehmbarer Zeit, zu einem gegebenen Message-Authentification-Code den ursprünglichen Text zu finden. Ein Algorithmus, der einen solchen Message-Authentification-Code erzeugt, wird als **Einweg-Hashfunktion** bezeichnet.

3. Es sollte ebenso unmöglich sein, zwei Ursprungstexte zu finden, die den selben Message-Authentification-Code erzeugen. In diesem Fall heißt die Einweg-Hashfunktion **kollisionsfrei**.

Das hier vorgestellte Verfahren wird zur Authentifikation ebenfalls einen Message-Authentification-Code verwenden, der jedoch auf andere Weise erzeugt wird (siehe 3.6.1).

Auf die oben beschriebene Benutzerauthentifikation soll hier nicht näher eingegangen werden, da sich das hier vorgestellte Verfahren auf den Austausch von Nachrichten per E-Mail beschränkt, so daß die Nachrichtenintegrität und -authentifikation die wichtigeren Gesichtspunkte darstellen.

1.5 Lösungsansatz der Algorithmischen Informationstheorie

Das Verfahren, das hier vorgestellt werden soll, ist, vielleicht entgegen mancher Erwartung, die sich im Laufe dieses Kapitels gebildet haben könnte, ein symmetrisches Verfahren. Doch im Gegensatz zu den in Abschnitt 1.2.1 beschriebenenen Algorithmenen, wird kein Schlüssel fester Länge verwendet, sondern der Schlüssel liegt in Form eines binären Baumes vor, in dem alle zum Nachrichtenaustausch benutzten Wörter abgelegt sind. Die Verschlüsselung stellt sich dann als Pfad durch den Baum zum entsprechenden Eintrag dar. Die verwendeten Bäume sind dynamisch, d. h. ihre Form hängt direkt von den versendeten Nachrichten ab und ändert sich regelmäßig mit diesen. Dadurch entfällt auch der ständige Austausch neuer Sitzungsschlüssel, da diese ja nebenbei automatisch generiert werden und so für die weitere Kommunikation zur Verfügung stehen.

Im nächsten Kapitel werden die informationstheoretischen Voraussetzungen für die Umsetzung des Verfahrens erörtert. In Kapitel 3 wird auf die konkrete Gestaltung des Algorithmus innerhalb des Programmes eingegangen, das den praktischen Teil dieser Arbeit bildet.

Kapitel 2

Informationstheoretische Grundlagen des Verfahrens

2.1 Die Idee des Verfahrens

Ziel der Verschlüsselung eines Nachrichtenaustauschs zwischen zwei Kommunikationspartnern ist es, einen Geheimtext in Form einer binären Kodierung zu erzeugen, so daß für einen außenstehenden Dritten kein Rückschluß vom Geheimtext auf den Klartext möglich ist. Andererseits muß ein Empfänger einem Geheimtext wieder eindeutig einen Klartext zuordnen können.

Betrachtet man eine binäre Kodierung, so läßt sich jeder präfixfreie Kode durch einen binären Baum darstellen. Alle Wege im Baum von der Wurzel zu den Blättern definieren über ihre Beschriftung einen Kode C, so daß es zu jedem Wort $u \in \{0,1\}^k$ genau ein Präfix in C gibt, wenn die Länge des Wortes größer oder gleich der Tiefe des Baumes ist. Geht man zu einem reduzierten Baum, wie er bei dem hier beschriebenen Verfahren verwendet werden soll, über, so geht diese Eigenschaft verloren. Stattdessen muß ein Mittel geschaffen werden, die Präfixfreiheit der Kodierung zu erhalten. Dies erreicht man durch ein Bitpaar als Beschriftung der Kanten und einem Zusatzbit für einen positiven Vergleich. Zur näheren Erläuterung beachte man Abbildung 2.3 und Abschnitt 3.5.2. Jede Folge läßt sich in ein Produkt von Faktoren aus C zerlegen, es läßt sich dabei aber für eine hinreichend große Folge aufgrund von Faktorzerlegungen nicht sehen, aus welchem Kode C sie erzeugt wurde.

Identifiziert man die im Suchbaum abgelegte Information mit einer Quelle (A, p), so scheint der Kode angreifbar zu sein, wenn man A und p approximativ durch Beobachtung bestimmen kann. Dieser Tatsache kann man sich entziehen, indem man jedes Wort nach seiner Verwendung in der Nachricht im Suchbaum unter Erhaltung der Ordnung zur Wurzel nach oben rotieren läßt. Empfänger und Sender der Nachricht führen diese Rotationen parallel aus, so daß ihre beiden Suchbäume immer übereinstimmen. Dadurch wird ein Schritt vom konstanten Kode zu einem sich dynamisch anpassenden Kode gegangen, der abhängig ist vom gesamten Kommunikationsverlauf zwischen zwei Partnern. Wird auf diese Weise eine binäre Folge als Geheimtext erzeugt, so wird durch die Theoreme der Informationstherie garantiert, daß dieser Kode nahezu optimal ist, da durch die Transformationen der Suchbäume der Kode stets in der Nähe des Optimums liegt. Die Kommunikationsfolge ist approximativ zufällig.

Für die spätere praktische Umsetzung soll es genügen, die Rotationen erst nach der kompletten Übermittlung der Nachricht durchzuführen, um einen entsprechenden Verwaltungsaufwand während der Kodierung zu sparen. Der Kode wird dadurch nicht angreifbarer, da selbst für sehr große Nachrichtentexte eine statistische Auswertung aufgrund von Beobachtungen nicht möglich ist (vergleiche auch 3.8.1).

2.2 Ablauf der Kommunikation

Die beiden Partner A und B wollen über Geheimtexte miteinander kommunizieren. Aus diesem Grund legt jeder von ihnen einen binären Suchbaum fest, der die zur Kommunikation notwendigen Zeichen und Wörter einer natürlichen Sprache enthält. Die verwendete Sprache könnte zum Beispiel die deutsche Sprache sein. Die beiden Suchbäume müssen natürlich übereinstimmen.

Will nun A eine Nachricht an B senden, so überträgt er für jedes Wort des Nachrichtentextes den ihm entsprechenden Weg von der Wurzel zum Auftreten des Wortes im Suchbaum. Durch die Präfixfreiheit des verwendeten Kodes kann der Empfänger B dem empfangenen Geheimtext wieder eindeutig Wörter aus der natürlichen Sprache zuordnen. Dabei folgt B in seinem Suchbaum dem Weg, den die binäre Folge des Geheimtextes darstellt und findet so die verschlüsselten Wörter. Diese Wörter können dann zur ursprünglichen Nachricht zusammengesetzt werden.

2.3 Die Knotenform

Das Verschlüsselungsverfahren beruht auf der Anordnung aller für eine Kommunikation in natürlicher Sprache benutzten Wörter in einem binären Suchbaum.

Die Knoten mit den verschiedenen Einträgen müssen dabei so angeordnet sein, daß eine lexikographische Ordnung entsteht.

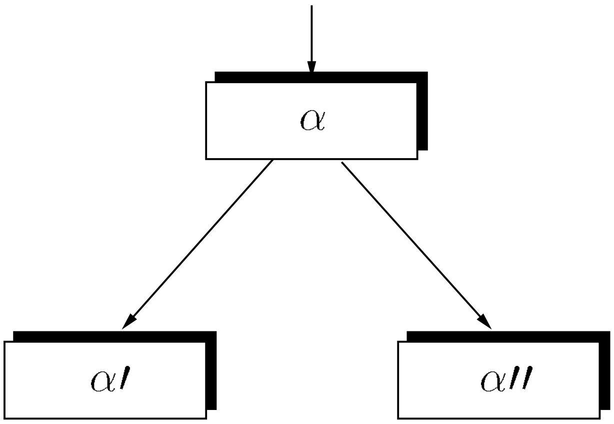


Abbildung 2.1: Knotenstruktur

Für jeden Knoten im Baum gilt also $\alpha' < \alpha < \alpha''$, wenn $' < '$ die gegebene Ordnung ist.

Bei jedem Suchdurchlauf nach einem Wort des Nachrichtentextes werden die folgenden Schritte vollzogen:

1. Starte an der Wurzel des Suchbaumes.
2. Vergleiche den aktuellen Eintrag mit dem zu suchenden Wort. Bei Gleichheit beende die Suche.
3. Bei Ungleichheit prüfe, ob das Suchwort lexikographisch kleiner ist als der aktuelle Eintrag. Ist dies der Fall, suche im linken Teilbaum weiter, sonst suche im rechten Teilbaum.
4. Wiederhole die Schritte 2. und 3. bis das Wort gefunden wurde.
5. Endet die Suche in einem Blatt des Suchbaumes und ist der Eintrag dieses Blattes ungleich dem Suchwort, so ist das Suchwort nicht im Baum enthalten.

Dadurch ergibt sich an die Knoten im Suchbaum folgende Anforderung:

- Ein Eintrag mit einem Wort aus der natürlichen Sprache.
- Ein Verweis auf den linken Sohn.

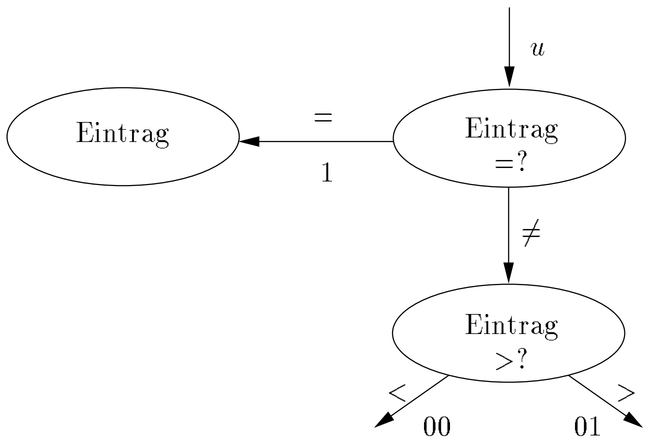


Abbildung 2.2: Die durchzuführenden Vergleiche an jedem Knoten des Suchbaumes.

- Ein Verweis auf den rechten Sohn.

Die Beschriftung der Kanten des Suchbaumes dient dazu während des Suchdurchlaufs die binäre Kodierung für das Suchwort zu erzeugen. Aus Praktikabilitätsgründen wird jede Verzweigung mit einem Bitpaar 00 bzw. 01 für links und rechts beschriftet. Bei erfolgreichem Vergleich ergibt sich als Kodierung eine 1. Eine Begründung hierfür findet man in 3.5.2.

Für jedes Wort einer Nachricht wird ein Algorithmus der Art

```
while Anfrage ≠ Eintrag
    do  if Anfrage < Eintrag
            then follow 0
            else follow 1
        fi
    od
```

durchlaufen.

2.4 Der Aufbau des Suchbaumes

Natürlich kann man die Knotenform aus 2.3 zu einem Knoten zusammenfassen, indem man direkt die Vergleiche durchführt. Durch diese Darstellung erhält man folgende Baumstruktur für den Suchbaum:

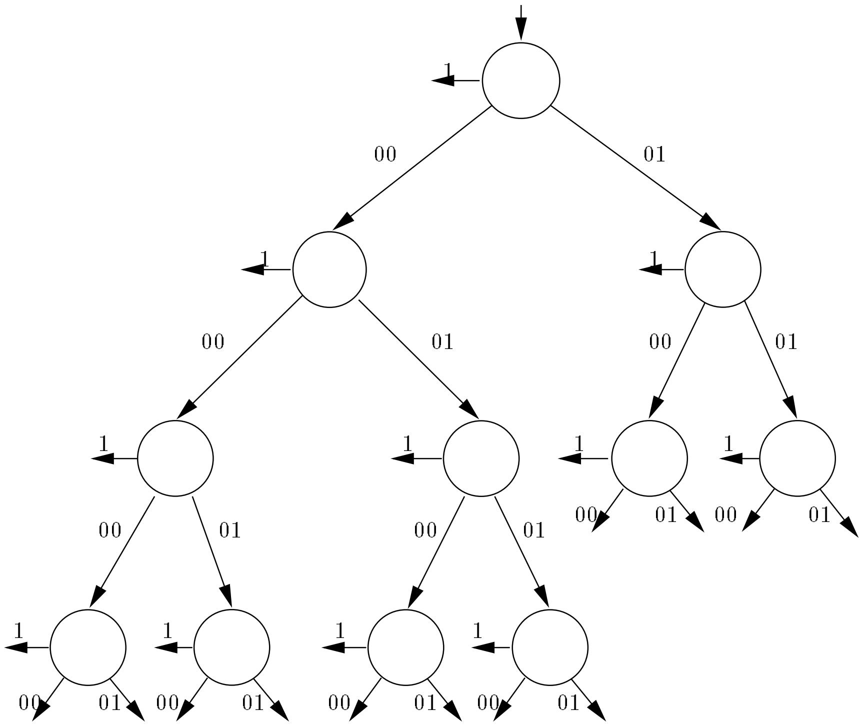


Abbildung 2.3: Die Struktur der verwendeten Suchbäume

Im Suchbaum unterliegen die Einträge in den Knoten einer lexikographischen Ordnung. Die Kanten sind an den Linksverzweigungen mit 00 beschriftet, an den Rechtsverzweigungen mit 01. Wird ein Knoten gefunden, wird im Geheimtext ein zusätzliches Bit in Form einer 1 eingetragen. Dieser Tatsache wird in der Baumdarstellung mit der 1 direkt am Knoten Rechnung getragen.

2.5 Rotationen im Suchbaum

Der dynamische Charakter der Suchbäume basiert auf der Tatsache, daß Wörter einer Nachricht nach deren Übertragung zur Wurzel rotieren. Werden diese Rotationen für alle Nachrichten vollzogen, so wird dadurch erreicht, daß immer wieder benutzte Wörter sich ständig in Wurzelnähe befinden und ein entsprechend optimaler Kode erzeugt wird.

Die Rotationen, die für jedes Wort durchzuführen sind, stellen sich als Abfolge von Einzelrotationen zwischen je zwei Knoten dar. Dabei entspricht die Zahl der Rotationen pro Wort seiner Tiefe im Suchbaum.

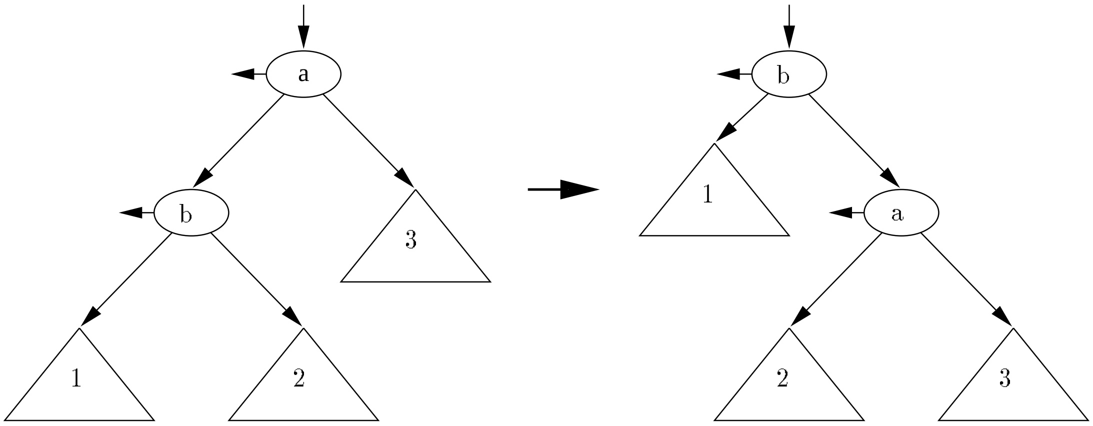


Abbildung 2.4: Schema für die Rotation eines Wortes zur Wurzel.

2.6 Ein Beispiel

Ein Beispiel soll die Darstellungen der vorangegangen Abschnitte verdeutlichen. Grundlage bildet der Suchbaum in Abbildung 2.6.

Will ein Sender nun den Klartext „Eine einfache Methode ist das Chiffrieren einer Nachricht“ verschlüsseln, beginnt er mit der Suche nach dem Wort an der Wurzel des Suchbaumes. Da der Eintrag an der Wurzel und das Suchwort nicht übereinstimmen, verzweigt die Suche in den linken Teilbaum der Wurzel, da das Wort „Eine“ lexikographisch kleiner ist als das Wort „einfache“. Von einer Unterscheidung zwischen Groß- und Kleinschreibung soll hier abgesehen werden. In Realität muß sie natürlich beachtet werden, doch dient eine solche Unterscheidung nicht dem grundsätzlichen Verständnis des Beispiels. Aufgrund der Linksverzweigung der Suche wird dem Geheimtext ein Bitpaar 00 angehängt. Verfolg man die Such im Suchbaum weiter, so gelangt man irgendwann zu dem Knoten mit dem Eintrag „Eine“. Der Geheimtext bis zu dieser stelle lautet 0001. Beim Vergleich an diesem Knoten wird jetzt Gleichheit festgestellt. Dem Geheimtext wird eine 1 angefügt. Man erhält somit für das Wort „Eine“ die verschlüsselte Darstellung 00011.

Folgt man dieser Vorgehensweise für den gesamten Text, so erhält man ohne Berücksichtigung von Rotationen

$$00011101101001000100100100010110101 1.$$

Entschlüsseln kann man diesen Geheimtext, indem man für jedes Bitpaar 00 im Suchbaum die Suche nach links verzweigt, für 01 nach rechts. Wird eine 1 gelesen, weiß man, daß der Eintrag des aktuellen Knoten das gesuchte Wort enthält.

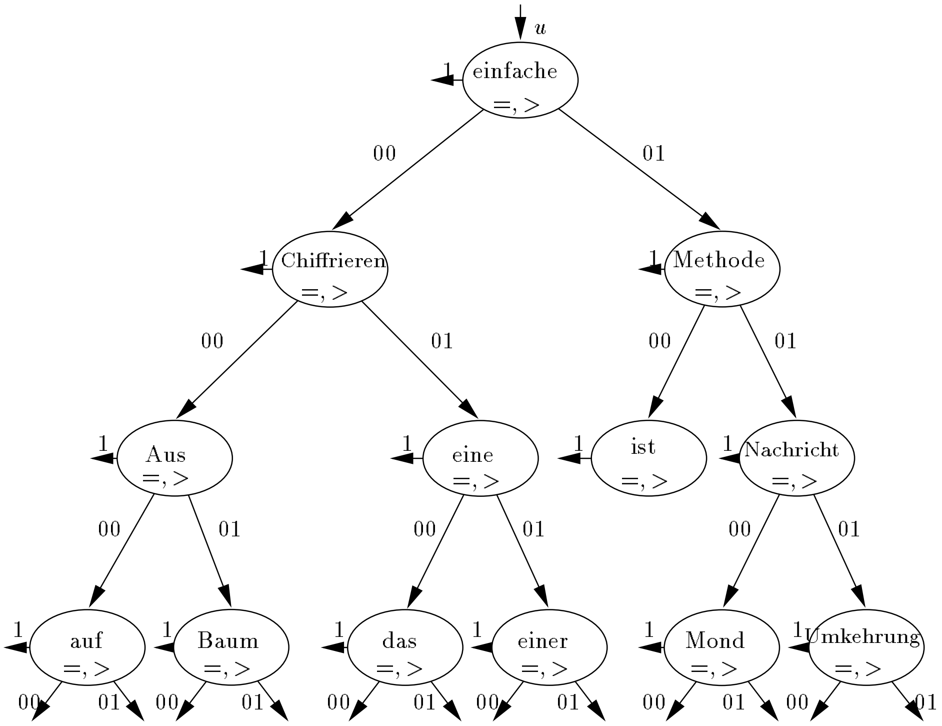


Abbildung 2.5: Beispiel für die Kodierung eines Satzes anhand eines Suchbaumes.

Kapitel 3

Die praktische Umsetzung des Verfahrens

Es soll ein Programm entstehen, das eine Nachricht in natürlicher Sprache aus einer Datei ausliest, diese automatisch unter Beachtung gewisser Protokollbedingungen auf Basis eines binären Suchbaumes verschlüsselt und an einen Empfänger versendet. Wie bei symmetrischen Verfahren üblich, wird der Empfänger zum Lesen der Nachricht das Verfahren genau in der umgekehrten Reihenfolge unter Zuhilfenahme des gleichen Schlüsselbaumes anwenden.

3.1 Grundlagen der Implementierung

Für die praktische Umsetzung des Verfahrens sollen nun einige grundlegende Voraussetzungen vereinbart werden, die die genaue Gestaltung der Implementierung festlegen. Diese Anforderungen sollen keinesfalls als Einschränkung angesehen werden; sie dienen lediglich der genaueren Definition.

- Die praktische Umsetzung soll sich auf den reinen Nachrichtenaustausch in Form natürlicher Sprache beschränken, d. h. es soll keine Übertragung von Programmdateien oder ähnlichem stattfinden. Dies hat primär den Vorteil, daß der Suchbaum nur darstellbare Zeichen bzw. Wörter über diesen Zeichen enthält. Es müssen keine Kontrollsequenzen oder Steuerzeichen beachtet werden. Sollte die Implementierung über diesen Umfang hinaus erweitert werden, so muß nur der Mechanismus, mit dem die Einträge für den Suchbaum aus einer Datei gelesen werden, geändert werden. Ein Wort ist hierbei eine beliebige Folge darstellbarer Zeichen, die von Leerzeichen, Tabulatorzeichen, Satzzeichen oder Zeilenendemarkierungen begrenzt ist.

- Es dürfte klar sein, daß bei Kommunikationsbeginn noch nicht alle Wörter, die für eine Verständigung in natürlicher Sprache gebraucht werden, im Suchbaum vorhanden sein können. Deshalb werden durch die fortlaufende Kommunikation zwischen zwei Partnern neu auftretende Wörter dem Wortschatz im Suchbaum hinzugefügt und können für die spätere Kommunikation genutzt werden.

- Bei Kommunikationsbeginn zwischen zwei Partnern wird daher von einem Basisbaum ausgegangen, der zumindest das Alphabet in Groß- und Kleinschreibung, sowie Satz- und Sonderzeichen enthalten muß. Es muß ein Mechanismus geschaffen werden, der es ermöglicht, unbekannte Wörter, d. h. Wörter, die noch nicht im Suchbaum enthalten sind, durch einzelne Zeichen zu beschreiben.

- Wie man aus Kapitel 1 weiß, ist es nicht nur wichtig, die Nachricht geheim zu übertragen, ohne daß ein Dritter sie lesen kann, sondern der Empfänger muß auch sicher sein, daß die Nachricht wirklich vom angegebenen Absender stammt. Neben der reinen Verschlüsselung der Nachricht muß eine Möglichkeit der Authentifikation des Senders für den Empfänger implementiert werden.

- Bei dem hier beschriebenen Verfahren stellt der binäre Suchbaum den Schlüssel dar. Da dieser nach jeder Nachricht dynamisch umgeordnet wird, muß der Empfänger einer eintreffenden Nachricht eindeutig einen Schlüssel, also einen Suchbaum in einer bestimmten Form, zuordnen können. Für den Empfänger muß eine eindeutige Reihenfolge der Nachrichten erkennbar sein, da von ihr die Form des aktuell benutzten Suchbaums direkt abhängt. Auf diese Art kann eine reibungslose Kommunikationsabfolge gewährleisetet werden.

- Der Empfänger muß eindeutig das Ende einer Nachricht erkennen können. Er sollte auch feststellen können, ob eine Nachricht komplett oder nur in Teilen übertragen wurde, d. h. ob ein Gegener eventuell ganze Abschnitte der Nachricht gelöscht, ausgetauscht oder verändert hat.

- Sollte eine Nachricht auf die oben beschriebene Art und Weise nicht korrekt beim Empfänger angekommen sein, so muß der Sender die Nachricht nochmals versenden, da eine Rekunstruktion der Nachricht auf Empfängerseite in der Regel nicht möglich ist. Der Sender bewahrt daher den zu einer Nachricht gehörenden Suchbaum solange auf, bis er weiß, ob die Nachricht korrekt übermittelt wurde. Der Empfänger muß folglich den Empfang einer Nachricht bestätigen, so daß der Sender die für das Verfahren notwendigen Rotationen an seinem Baum endgültig durchführen kann.

3.2 Kommunikationsablauf

3.2.1 Senden

1. Der Sender verfaßt wie gewohnt seine Nachricht und speichert diese ab. Dann ruft er das Verschlüsselungsprogramm auf. Diese Verfahrensweise kann dadurch erleichtert werden, daß man das Verschlüsselungsprogramm an ein Mailprogramm wie Xmail oder auch an Netscape anbindet. Die Verschlüsselung könnte dann sogar automatisch gestartet werden, wenn ein bestimmter Adressat als Empfänger eingetragen wird.

2. Zur Initalisierung liest das Verschlüsselungsprogramm den abgespeicherten Baum in der Grundform oder, wenn bereits eine Kommunikation stattgefunden hat, in der Version mit bereits abgespeicherten neuen Wörtern.

3. Dannach werden mit Hilfe einer Zufallsfunktion die Authentifikationsdaten bestimmt, die in den Kopf der Nachricht geschrieben werden. Diese werden vom Empfänger genutzt, um den Sender eindeutig zu indentifizieren. Um eine Änderung der Daten zu verhindern, wird dieser MAC selbstverständlich ebenfalls verschlüsselt.

4. Jetzt wird die Nachricht nach dem hier beschriebenen Verfahren verschlüsselt, d. h. das Programm sucht für jedes Wort der Nachricht einen entsprechenden Eintrag im Suchbaum und benutzt die Beschriftung aus $\{0,1\}^k$ des Pfades zu dem Auftreten des gesuchten Wortes zur Kodierung. Tritt ein Wort zum ersten Mal auf, wird es in Form von einzelnen Zeichen übertragen. Die Beschriftung gibt dabei an, ob bei der Suche im Baum nach links oder rechts verzweigt wurde.

5. Um die Reihenfolge der Nachrichten, die vom Sender zum Empfänger gehen eindeutig bestimmen zu können, wird dem binären Code eine Nummerierung hinzugefügt.

6. Der Empfänger sollte die Vollständigkeit der Nachricht überprüfen können. Aus diesem Grund wird ein Nachrichtenendezeichen angefügt. Darüber hinaus wird eine Prüfsumme der zu übermittelnden Bits bestimmt und der Nachricht hinzugefügt. So kann der Empfänger den korrekten Umfang der Nachricht überprüfen.

7. Die Nachricht kann nun auf gewohntem Weg per E-Mail versendet werden.

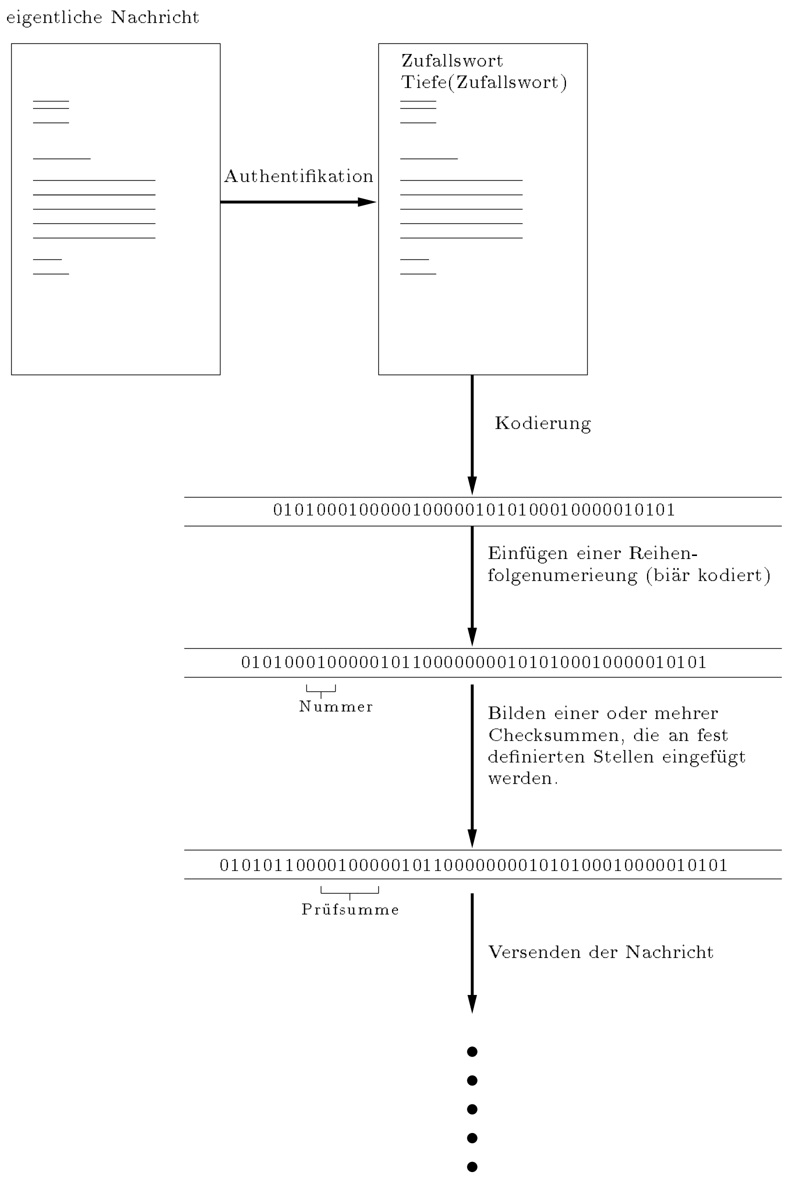


Abbildung 3.1: Gestaltung der Kommunikation auf Senderseite

3.2.2 Empfangen

1. Nach dem Eintreffen der Nachricht überprüft der Empfänger zuerst die Vollständigkeit der Nachrichtenübertragung anhand der übermittelten Prüfsummen. Nur wenn die Nachricht korrekt gesendet wurde, wird versucht, sie zu entschlüsseln.

2. Um die Nachricht korrekt zu entschlüsseln, muß man wissen, ob der aktuelle

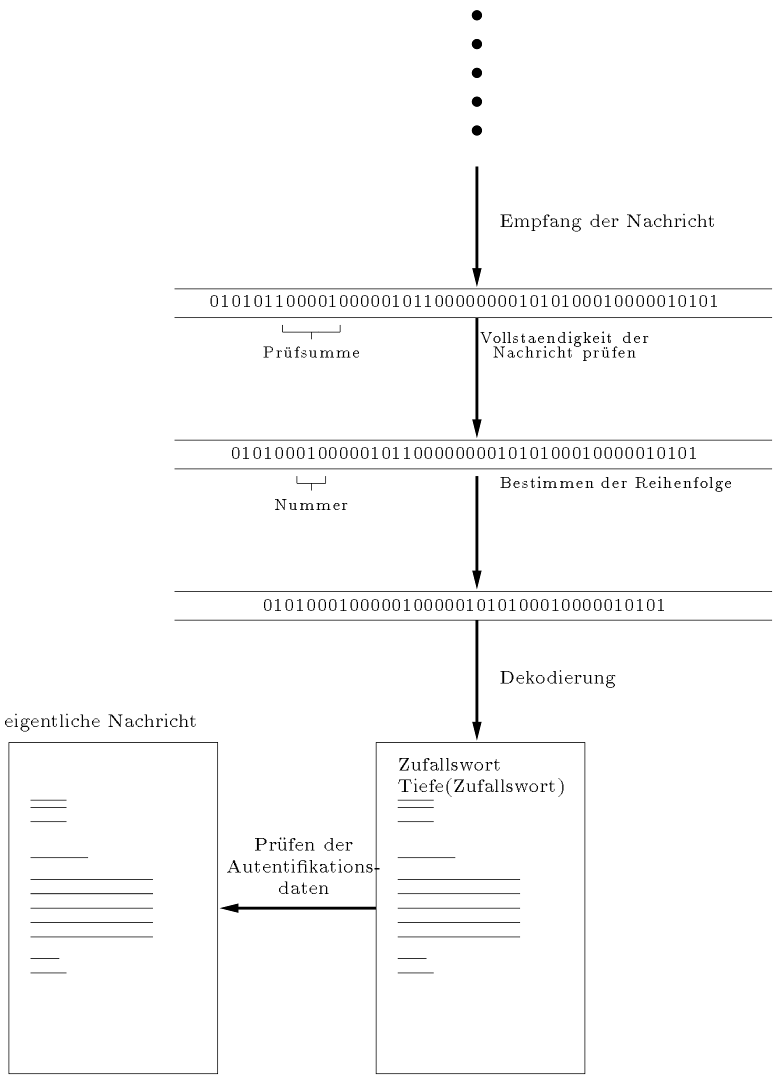


Abbildung 3.2: Gestaltung der Kommunikation auf Empfängerseite

Suchbaum zur empfangenen Nachricht korrespondiert. Man muß daher die Reihenfolge der eingegangenen Nachrichten bestimmen. Dies geschieht über die der Nachricht mitgegebene Numerierung.

3. Konnte der Nachricht ein Suchbaum eindeutig zugewiesen werden, so kann die Nachricht unter Zuhilfenahme des entsprechenden Binärbaumes entschlüsselt werden. Neue Wörter, die in Form von einzelnen Zeichen übermittelt werden, werden nach dem Zusammensetzen sofort in den Suchbaum eingefügt.

4. Um zu wissen, ob der reale und der angegebene Sener übereinstimmen,

werden die Authentifikationsdaten, die am Anfang des Nachrichtentextes stehen überprüft.

5. Wurde die Nachricht für korrekt, so können die notwendigen Rotationen im Suchbaum auf Basis der gerade entschlüsselten Nachricht durchgeführt werden.

3.3 Kommunikationsprotokolle

3.3.1 Allgemeine Anforderungen

Um einen kryptographischen Algorithmus anwenden zu können, d. h. um ein neues Verfahren in die Realität umzusetzen, bedarf es einer genauen Definiton der Kommunikationsabläufe. Dieser Schritt wird von einem Protokoll vollzogen, das eine Folge von Aktionen festlegt, die zur Verfahrensanwendung notwendig sind. Ein Protokoll erfüllt die Grundvoraussetzungen:

- Ziel des Protokolls ist die Durchführung einer bestimmten Aufgabe.
- Am Protokoll sind mindestens zwei Teilnehmer beteiligt.
- Die Teilnehmer müssen die Aktionen, die das Protokoll, vorgibt im voraus kennen und sich an diese Anweisungen halten.
- Das Protokoll muß eindeutig sein.
- Das Protokoll muß vollständig sein, d. h. alle möglichen Fälle müssen definiert sein.

Ein kryptographisches Protokoll formalisiert die Anwendung eines Verfahrens und abstrahiert von der Umgebung, in der es eingesetzt wird. So kann etwa von einer konkreten Implementierung abgesehen werden, wenn die Sicherheit des Protokolls bereits außer Frage steht.

In der Kommunikationsabfolge unterscheidet man drei Typen von Protokollen, jenachdem, ob in den Nachrichtenaustausch zwischen zwei Teilnehmern noch weitere Instanzen eingreifen oder nicht.

- **Eigenständiges Protokoll**, d. h. nur die Kommunikationspartner nehmen am Ablauf des Nachrichtenaustauschs teil.

- **Protokoll mit Schiedsrichter**, bei dem die Kommunikation im Grunde genommen direkt, d. h. als eigenständiges Protokoll, abläuft. Es kann jedoch in bestimmten Fällen ein Schiedsrichter hinzugezogen werden, der die Korrektheit der Abläufe im nachhinein prüft.

- **Protokoll mit Vermittler**. Die gesamte Kommunikation läuft hier über einen unvoreingenommen Dritten, der von allen Kommunikationspartnern akzeptiert wird, ab. Dieser prüft ständig den korrekten Ablauf des Nachrichtenaustauschs.

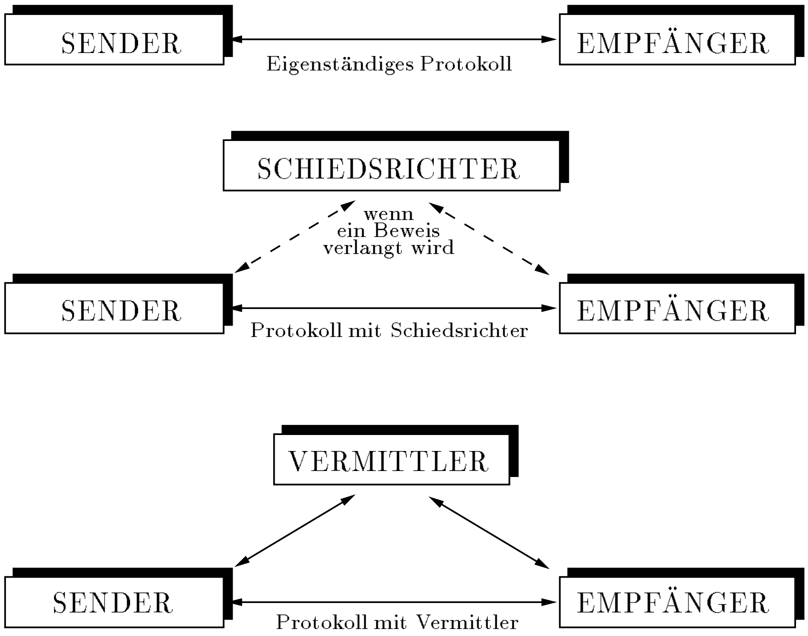


Abbildung 3.3: Eigenständiges Protokoll, Protokoll mit Schiedsrichter bzw. Vermittler

3.3.2 Ein konkretes Protokoll

Das hier vorgestellte Verfahren soll durch ein Protokoll umgesetzt werden, das bei der Kommunikation auf das Einwirken von Dritten in Form eines Schiedsrichters oder Vermittlers verzichtet. Bei einem solchen eigenständigen Protokoll oder self-enforcing protocol geht man davon aus, daß sich die kommunizierenden Parteien vertrauen und eventuelle Betrügereien oder Beeinflussungen des Nachrichtenaustauschs von selbst entdecken können.

Kern des Verfahrens ist die Anwendung eines symmetrischen Kryptosystems, das zum Schlüsselaustausch zwischen den Teilnehmern zu einem hybriden Kryptosystem erweitert wird. Zwei Partner verfügen jeweils über einen identischen Schlüssel in Form eines binären Baumes, der die Grundlage für den symmetrischen Teil des Kryptosystems bildet. Eine Mindestkonfiguration (siehe 3.4), die als Ausgangspunkt für die Kommunikation dient, wird zu Beginn des Nachrichtenaustauschs mit Hilfe eines asymmetrischen Verfahrens übertragen. Durch die dynamische Umordnung der Bäume zweier Kommunikationspartner wird gewährt, daß immer wieder neue Sitzungsschlüssel automatisch erzeugt werden.

Im Folgenden soll ein Protokoll aufgestellt werden, das die Anwendung des in dieser Arbeit beschriebenen Verfahrens umsetzt. Der Nachrichtenaustausch findet zwischen den Kommunikationspartnern A und B statt. Die eigentliche Übermittlung der Nachricht wird mit Hilfe des symmetrischen Algorithmus BV chiffriert und dechiffriert. Grundlage bildet der Schlüssel K_n, der in Form eines binären Baumes vorliegt und sich individuell an die Kommunikation der beiden Teilnehmer anpaßt und dynamisch nach jeder Nachrichtenübermittlung ändert. Die Grundkonfiguration K_0 dieses Schlüssels wird zu Beginn der Kommunikation über ein Public-Key-Verfahren ausgetauscht. Die Teilnehmer verfügen zu diesem Zweck jeweils über einen öffentlichen Schlüssel E_A bzw. E_B und über einen privaten Schlüssel D_A bzw. D_B.

1. Einigung der Kommunikationspartner A und B auf die Anwendung des in dieser Arbeit beschriebenen kryptographischen Verfahrens.

2. Austausch eines Mindestschlüssels K_0, der zumindest alle darstellbaren Zeichen enthält, mit Hilfe eines asymmetrischen Verfahrens: Teilnehmer A legt einen Schlüssel K_0 fest und verschlüsselt diesen mit dem öffentlichen Schlüssel von Teilnehmer B.

$$E_B(K_0) \quad = \quad c_K$$

   A sendet den Geheimtext c_K, der den Schlüssel K_0 enthält an B. Dieser entschlüsselt ihn wieder mit seinem privaten Schlüssel und erhält so den eigentlichen Schlüssel zur Anwendung des Verfahrens.

$$K_0 \quad = \quad D_B(c_k)$$

   Der hier beschriebene Schritt kann natürlich auch von Teilnehmer B ausgehen und erfolgt dann analog.

3. Jedem Teilnehmer steht jetzt für die Kommunikation mit dem symmetrischen Verfahren der gleiche Schlüssel K_0 zur Verfügung. Will nun A die Nachricht m an B senden, so kann er sie folgendermaßen verschlüsseln: Der Teilnehmer A bestimmt einen MAC (siehe Abschnitt 3.6.1) fügt diesen mit

dem Klartext m zusammen. Dann wendet er den Algorithmus BV basierend auf dem binären Baum K_0 auf den neuen Klartext m an und erhält den Geheimtext c.

$$c = BV_{K_0}(m)$$

4. Er bildet eine Checksumme (siehe Abschnitt 3.6.3), die er dem eigentlichen Geheimtext hinzufügt. Diese soll dazu dienen, den vollständigen Empfang einer Nachricht zu überprüfen.

5. Die ausgehenden Nachrichten werden in der Reihenfolge ihres Versendens durchnummiert, um ihnen bei der Entschlüsselung eindeutig einen Suchbaum zuordnen zu können.

6. Die gesamte Nachricht sendet A nun als Mail an Teilnehmer B.

7. Der Empfänder B prüft mit Hilfe der Checksumme den korrekten Empfang der Nachricht. Verläuft dieser Test positiv, dechiffriert er den erhaltenen Geheimtext c, indem er den Algorithmus BV^{-1} mit dem Schlüssel K_0 anwendet

$$m = BV_{K_0}^{-1}(c)$$

und erhält so eine Nachricht m, die aus dem eigentlichen Klartext und dem MAC besteht. Kann die Authentifikation des Senders anhand des MAC durchgeführt werden, kann die Nachricht verwendet werden.

8. Teilnehmer B bestätigt A den korrekten Erhalt der Nachricht m. Diese Bestätigung muß verschlüsselt ablaufen, um ein Einwirken eines Gegners zu verhindern. Teilnehmer B übermittelt A zum Beispiel ein OK-Symbol, das als Sonderzeichen in K_0 enthalten war.

9. Beide Teilnehmer führen jetzt die notwendigen Umformungen, basierend auf Nachricht m, am Baum K_0 aus, der eventuell um neue Wörter erweitert wurde, und erhalten so einen neuen Sitzungsschlüssel K_1.

10. Für die weitere Kommunikation werden die Schritte 3. bis 9. wiederholt. Die Teilnehmer erhalten bei jedem Durchlauf einen neuen Schlüssel K_n. Ein Austausch wie in Schritt 2. beschrieben oder gar das Aufstellen eines neuen Schlüssels K ist nicht notwendig, da sich der Schlüssel mit jeder Nachrichtenübertragung ändert.

3.4 Die Problematik des Schlüsselaustauschs

Wie bei allen anderen symmetrischen Kryptosystemen stellt sich auch bei dem hier beschriebenen Verfahren das Problem des sicheren Schlüsselaustauschs zwischen zwei Kommunikationspartnern. So ist es unabdingbar, daß zwei Teilnehmer mit dem gleichen Baum bzw. dem gleichen Wortschatz die Kommunikation beginnen, da dieser den Schlüssel für die Verschlüsselungen und Entschlüsselungen darstellt.

Geht man davon aus, daß die Kommunikation mit einem Grundbaum startet, der bei jeder Installation des Kryptosystems gleich ist, so könnte man die Bäume beider Seiten durch eine Art Verrauschen mit Pseudonachrichten individualisieren. Man muß jedoch davon ausgehen, daß auch einem Gegener dieser Grundbaum bekannt wäre und daß jegliche Kommunikation von Anfang an, d. h. auch die Pseudonachrichten, abgefangen würde. So könnte ein Gegener jederzeit über die Schlüsselbäume verfügen und somit alle Geheimnachrichten entschlüsseln und lesen, ja sogar stattdessen selbst formulierte Nachrichten verschlüsseln und diese gegen die ursprüngliche Nachricht austauschen.

Da bei diesem Verfahren der Suchbaum den eigentlichen Schlüssel bildet, ist dieser der wichtigste Bestandteil der Kommunikation und muß immer geheim gehalten werden. Es bleibt daher nur die Möglichkeit, den Baum verschlüsselt zu übertragen. Wie im vorangegangenen Abschnitt über das verwendete Protokoll bereits angedeutet wurde, bietet sich hierzu ein Public-Key-Verfahren an, wodurch ein hybrides Kryptosystem entsteht. Man kann hier natürlich jedes geeignete asymmetrische Verfahren verwenden. Eine Möglichkeit wäre zum Beispiel das RSA-Verfahren aus Abschnitt 1.2.4.

Es stellt sich nun die Frage, welche Anforderungen an diesen Grundbaum zu stellen sind. Welche Informationen müssen im Grundbaum mindestens enthalten sein?

Die Informationen, die beim Nachrichtenaustausch übermittelt werden, bestehen bei einer natürlichen Sprache aus Buchstaben, Zahlen und einigen Sonderzeichen. Wie man in einem späteren Abschnitt noch sehen kann, läßt sich jedes Wort einer Sprache in seine Bestandteile zerlegen und diese können dann einzeln verschlüsselt übertragen werden. Soll die Kommunikation z. B. in deutscher oder englischer Sprache ablaufen, genügt es somit, wenn der Grundbaum das Alphabet in Groß- und Kleinbuchstaben $A - Z$, $a - z$, die Ziffern $0 - 9$, sowie die darstellbaren Sonderzeichen, wie , | ! | ? | . usw., enthält. Dies sind ca. 98 Zeichen, die gleichzeitig auch die Mindestanforderung an den Grundbaum darstellen, damit jedes Wort der Sprache erzeugt werden kann. Geht man von einer Zeichendarstellung mit 8 Bit aus, so müssen nur ca. 800 Bit an Information mit einem Public-Key-Verfahren übertragen werden.

Bei der Übertragung muß selbstverständlich gewährt sein, daß Teilnehmer B verifizieren kann, daß er einen Schlüssel von Teilnehmer A erhält und nicht von irgendjemand anderem. D. h. der Schlüsselaustausch muß mit einer Authentifikation einhergehen. An dieser Stelle sei jedoch auf die in der Literatur beschriebenen Protokolle zum Austausch von Schlüsseln verwiesen, da dies nicht Gegenstand dieser Arbeit ist.

Selbst wenn ein Grundbaum sicher übertragen werden kann, so stellt sich doch die Frage, ob ein Gegner eine solche Grundkonfiguration nicht erraten könnte; sie enthält ja schließlich nur ein paar Buchstaben und Zahlen. Der Gegener würde eine Art Brute-Force-Angriff auf die Grundkonfiguration durchführen, indem er alle möglichen Kombinationen der 98 Zeichen aufstellt und einfach die erste abgefangene Nachricht damit entschlüsselt, bis er eine brauchbare Nachricht gefunden hat, die einen Sinn ergibt. Die Komplexität dieses Problems ist jedoch bereits beachtlich. Bei einer Grundkonfiguration von 98 Zeichen gibt es C_{98} Kombinationsmöglichkeiten, was eine Größenordnung von 10^{55}, also etwa 2^{186}, darstellt. Ein solcher Brute-Force-Angriff ist aber in keiner annehmbaren Zeit mit heutigen Mitteln durchführbar. Die Zahl C_n wird in einem späteren Abschnitt über die Sicherheit und Komplexität dieses Verfahrens noch näher erläutert.

Zur Erinnerung sei nochmals darauf hingewiesen, daß ein Austausch eines Sitzungsschlüssels in dieser Grundkonfiguration nur bei der Neuaufnahme einer Kommunikation mit einem neuen Partner notwendig ist. Ansonsten wird der Sitzungsschlüssel für den nächsten Nachrichtenaustausch automatisch durch die dynamische Umbildung der Suchbäume auf Basis der gerade empfangenen Nachricht erzeugt.

Den größten Angriffspunkt des Verfahrens bildet der Austausch des Grundbaumes. Kann dieser Austausch geheim ablaufen, so kann ein Gegner auch alle folgenden Sitzungen abhören und es kann trotzdem nichts passieren.

3.5 Die Realisierung der binären Bäume

In Kapitel 2 wurden bereits die Grundlagen des Verfahrens beschrieben. Die eigentliche Methode der Verschlüsselung beruht dabei auf Schlüsseln, die in Form von binären Suchbäumen vorliegen. Bei jeder Verschlüsselung werden diese binären Bäume abgearbeitet und es entsteht für jedes Wort im Klartext eine bestimmte binäre Folge im Geheimtext, die nur von der aktuellen Form des Suchbaums abhängt.

3.5.1 Verwendete Datenstrukturen

Entsprechend der in 2.3 beschriebenen Knotenform, wurde zum Aufbau des Suchbaumes eine Datenstruktur *Tree* benutzt, die Angaben über den Vorgänger im Baum sowie den linken und rechten Sohn enthält. Darüber hinaus gibt es einen Eintrag, der das zu verschüsselnde Wort enthält. Die Angaben über den linken und rechten Sohn werden bei Suchzugriffen benutzt, um in die entsprechenden Unterbäume zu verzweigen. Auf den Vaterverweis wird bei der Durchführung der Rotationen zurückgegriffen.

```
typedef struct Tree
{
    char *info;             // Eintrag des Knotens
    struct Tree *left;      // Verweis auf den lexikographisch kleineren Teilbaum
    struct Tree *right;     // Verweis auf den lexikographisch größeren Teilbaum
    struct Tree *father;    // Verweis auf den Vorgängerknoten
} Tree;
```

In [HOT] wird davon ausgegangen, daß es genügt die Einträge der Knoten auf den präfixfreien Kode der Wörter der zugrundeliegenden Sprache zu beschränken. Dies erscheint für eine reale Umsetzung jedoch zu idealisiert, da sich Wörter einer natürlichen Sprache oft nur im letzten Buchstaben, z. B. bei einer Pluralbildung unterscheiden. Auch müssen die verschiedenen Kasusendungen im Suchbaum abgelegt werden, die sich ebenfalls nur in den letzten Zeichen unterscheiden. Aus diesem Grund enthalten die Einträge der Knoten in der Implementierung immer die kompletten Wörter.

3.5.2 Realisierung des Suchbaumes

Basierend auf dieser Datenstruktur aus 3.5.1 wird der gesamte Suchbaum als Verkettung der einzelnen Knoten aufgebaut.

Wird bei einer Verschlüsselung ein Wort im Binärbaum gesucht, so beginnt man an der Wurzel und vergleicht jeden Eintrag im Knoten mit dem aktuellen Suchwort. Wird Gleichheit festgestellt, ist die Suche beendet. Ist das Suchwort lexikographisch kleiner als der Knoteneintrag, verzweigt die Suche über den Linkersohn-Verweis zum linken Teilbaum, sonst zum rechten Teilbaum.

Durch diese Verzweigungen wird der erzeugte Geheimtext jedes einzelnen Wortes einer Nachricht bestimmt. Für den Fall einer Linksverzweigung wird dem Geheimtext in der aktuellen Programmversion die Bitfolge 00 angefügt, bei einer Rechtsverzweigung ist dies die Bitfolge 01. Wurde das Suchwort gefunden, wird eine 1 an den Geheimtext angefügt.

Man könnte eine Verzweigung auch nur mit 0 und 1 kenntlich machen, doch müßte man dann für das Ende der Suche ein anderes Kennzeichen einführen. Ohne diese Endekennzeichnung wäre der Algorithmus nicht in der Lage, festzustellen, ob er weiterlesen soll, oder ob bereits ein vollständiges Wort abgearbeitet wurde.

Die Kennzeichnung des Pfadendes für ein Suchwort läßt keinen direkten Rückschluß auf das Wort selbst zu. Ein Gegner muß in der Bitfolge aus 0 und 1 zuerst eine Folge eindeutig zuordnen können und dann kann er über die Länge der Folge höchstens feststellen, ob das entsprechende Wort bei der letzten Kommunikation eventuell benutzt wurde und daher aufgrund der durchgeführten Rotationen relativ nah bei der Wurzel zu finden ist.

Bei der Entschlüsselung eines Geheimtextes geht man genau in der umgekehrten Reihenfolge vor. Man liest den Geheimtext und verzweigt im Suchbaum nach links, falls die Bitfolge 00 gelesen wurde. Bei 01 verzweigt man nach rechts. Wurde das von diesen Bitfolgen präfixfreie Zeichen 1 gelesen, geht der Algorithmus davon aus, daß der aktuelle Knoten als Eintrag das Suchwort enthält. Dieser Eintrag wird dem Klartext hinzugefügt.

Endet die Verfolgung eines Pfades im Suchbaum an einem Blatt, ohne daß an diesem Blatt eine 1 im Geheimtext gelesen wird, geht das Programm davon aus, daß ein Wort übermittelt werden soll, daß noch nicht im Suchbaum vorhanden ist. Eine gesonderte Funktion liest nun die Informationen des Geheimtextes weiter, bis der entschlüsselte Text ein Leerzeichen, Satzzeichen oder einen Zeilenwechsel ergibt und setzt die so gelesenen Zeichen zu einem Wort zusammen. Endet in dieser Funktion wiederum die Suche an einem Blatt des Suchbaums, ohne das eine Endemarkierung gelesen wurde, so kann man aufgrund der Eindeutigkeit des Verfahrens davon ausgehen, daß der Geheimtext nicht korrekt übermittelt wurde oder daß eine Beeinflüssung durch einen Gegner vorliegt.

Ein auf diese Art und Weise übermitteltes neues Wort muß sofort in den Suchbaum eingefügt werden. Denn, sollte das neue Wort nochmals in der selben Nachricht verwendet werden, verweist der Sender nur auf die entsprechende Stelle im Suchbaum und überträgt das Wort nicht noch einmal Zeichenweise.

Zum Einfügen eines neuen Wortes wird dieses im Binärbaum gesucht. Die Suche endet an einem Knoten, dessen entsprechender linker oder rechter Sohnzeiger, jenachdem, ob das neue Wort lexikographisch kleiner oder größer als der Eintrag dieses Knoten ist, mit $NULL$ belegt ist. Genau an dieser Stelle wird das neue Wort als Eintrag eines zusätzlichen Knotens eingefügt. Der Sohnzeiger des

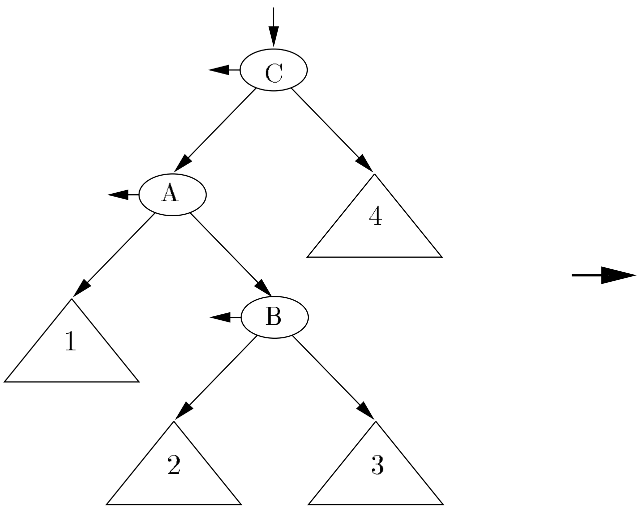
Knotens, in dem die Suche endete wird auf den neuen Knoten gesetzt, der Vaterverweis des neuen Knotens ist entsprechend entgegengerichtet. Damit ist das neue Wort dem Suchbaum hinzugefügt; eine weitere Bearbeitung ist primär nicht notwendig und wird erst bei der Umordnung des Suchbaumes nach kompletter Übermittlung bzw. Empfang der Nachricht notwendig (siehe 3.5.3).

3.5.3 Die Rotationen

Um die in Kapitel 2 beschriebenen Zugriffszeiten zu erreichen sowie eine Kompression des Klartextes im Geheimtext, ist es notwendig, daß häufiger benutzte Wörter und Zeichen sich relativ nah zur Wurzel befinden, d. h. diese Knoten haben im Suchbaum eine entsprechend geringe Tiefe.

Eine solche Plazierung von Wörtern in der Nähe der Wurzel kann erreicht werden, indem man nach jeder Nachrichtenübertragung bzw. nach jedem Nachrichtenempfang die zuletzt benutzten Wörter zur Wurzel rotiert. Häufig benutzte Wörter werden dadurch, daß in einer Großzahl der Mail vorkommen, auf Dauer in Wurzelnähe verbleiben.

Das Programm durchläuft nach der eigentlichen Nachrichtenverschlüsselung den Nachrichtentext noch einmal, sucht die entsprechenden Wörter inklusive der in dieser Nachricht neuen Wörter und rotiert Wort für Wort an die Wurzel. Dabei ist jede Rotation zur Wurzel eine Folge von Einzelrotationen zwischen einem Sohn- und einem Vaterknoten, bis der zu rotierende Knoten an der Wurzel des Suchbaumes angelangt ist. Die Anzahl der Rotationen pro Wort der Nachricht entspricht dessen Tiefe im Suchbaum.



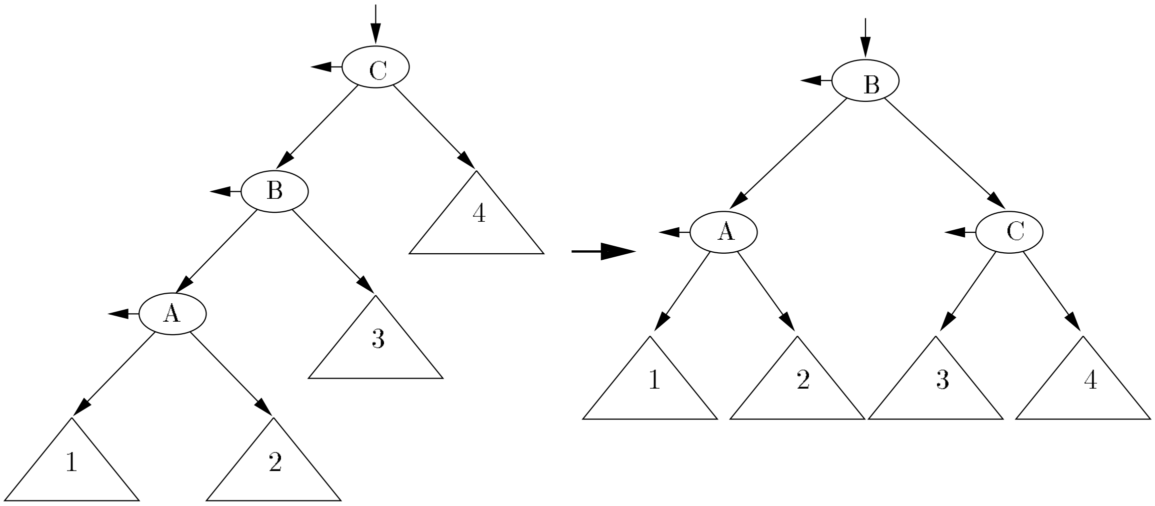


Abbildung 3.4: Beispiel einer Rotation: der Knoten mit dem Eintrag B soll an die Wurzel des Baumes rotieren.

3.5.4 Übertragung eines unbekannten Wortes

Natürlich kann es ohne weiteres vorkommen, daß ein Wort übertragen werden soll, das noch nicht in unserem Suchbaum enthalten ist. Liest das Programm ein solches Wort, so folgt es im Suchbaum dem Pfad, der dem Wort entsprechen würde. Die Suche endet jedoch an einem Knoten, dessen entsprechender Sohn leer ist. An dieser Stelle merkt das Programm, daß es sich um ein neues Wort handelt. Der bis hier hin gelesene Pfad wird der Kodierung der Nachricht hinzugefügt. Anschließend wird das zuletzt gelesene Wort in seine Bestandteile aus einzelnen Zeichen zerlegt und der Code für jedes Zeichen der Kodierung der Nachricht angehängt. Alle Zeichen, die in einem Wort vorkommen können, sind ja bereits im Grundbaum beim Kommunikationsstart enthalten und können deshalb ohne jegliche Probleme verschlüsselt werden.

Ist das gesamte neue Wort abgearbeitet, geht man zum nächsten Wort über und verfährt in gewohnter Weise. Ein Ende der Übertragung der einzelnen Zeichen muß nicht extra angegeben werden, da man bei natürlicher Sprache davon ausgehen kann, daß auf jedes Wort ein Zeilenwechsel, Leerzeichen oder ein sonstiges Satzzeichen folgt, was dann als solches erkannt wird. Konkret realisiert wird diese Art der Übertragung durch eine Funktion, die automatisch aufgerufen wird, sobald der Algorithmus mit seiner Suche an einem Blatt endet, ohne daß das zuletzt gelesene Wort gefunden wurde.

Ein neues Wort u stellt sich in der Kodierung also als Pfad zu einem leeren Blatt und einer Folge von kodierten Buchstaben dar:

Sei$A$ = Menge der darstellbaren Zeichen, $A^n \ni u = u_0 u_1 ... u_n, \quad u_i \in A$
$\Longrightarrow c_u = BV_K(u) = Path(u) + BV_K(u_0) + \cdots + BV_K(u_n)$
+bedeutet hier die Konkatenation der Binärkodierungen

Daraus folgt für die Empfängerseite die Vorgehensweise bei der Dekodierung: Liest das Programm beim Dekodieren einen Pfad, der zu einem leeren Blatt führt, so weiß man, daß anschließend in der kodierten Nachricht eine Buchstabenfolge beschrieben wird, die das neue Wort bilden soll. Das Programm setzt diese Buchstaben zu einem neuen Wort zusammen und fügt einen Knoten, der als Eintrag das neue Wort enthält, an die Stelle des leeren Blattes ein. Das Ende der Buchstabenfolge muß nicht extra gekennzeichnet werden, da es sich automatisch aus einem folgenden Leer- oder Satzzeichen ergibt.

Da ein neues Wort nach seiner Übermittlung direkt in den Baum eingefügt wird, kann, sollte das Wort ein weiteres Mal in der Nachricht vorkommen, der Sender auf die Stelle im Baum verweisen, an der das neue Wort steht; es muß also nicht ein zweites Mal als Buchstabenfolge übertragen werden. Für die nach der Nachrichtenübertragung durchzuführenden Rotationen gilt dabei, daß nicht die einzelnen Zeichen des neuen Wortes zur Wurzel rotiert werden, sondern nur das neu eingefügte Wort als Ganzes.

3.6 Die Sicherheit der Übertragung

Die Sicherheit eines Verfahrens ist nicht nur von der Sicherheit des Verschlüsselungsalgorithmus abhängig, sondern vielmehr auch davon, ob die Nachricht ungestört zum Empfänger gelangen kann. Die Störungen, die bei der Übertragung auftreten können, werden dabei entweder von Dritten absichtlich verursacht, um den Kommunikationsfuß zu durchkreuzen, oder sie sind rein physikalischer Natur. Physikalische Störungen sollen hier nicht betrachtet werden. Setzt man einen externen Kontrollmechanismus voraus, so kann man davon ausgegehen, daß die Nachricht rein physikalisch korrekt übermittelt wurde.

3.6.1 Authentifikation des Senders

Es gibt mehrere Möglichkeiten, wie ein Dritter die Kommunikation zwischen zwei Partnern behindern kann.

So kann er beispielsweise eine beliebige 0/1-Folge erzeugen, die zumindest von der äußeren Struktur (Folge von 00 und 01 Kombinationen mit abschließender

1) unserem Kodierungsverfahren entspricht, und zum Empfänger schicken. Dieser würde jedoch beim Versuch, die Nachricht zu dekodieren, merken, daß die angegebenen 0/1-Folgen keinem Pfad im Suchbaum entsprechen oder daß der Klartext keinen Sinn ergibt. Eine solche Störung führt also bei entsprechend hoher Wiederholungsrate höchstens zur Blockade des Empfangs, beeinflußt die Kommunikation zwischen den Partnern selbst jedoch nicht, d. h. es werden keine Nachrichten falsch verstanden.

Die größere Gefahr besteht darin, daß eine Nachricht korrekt entschlüsselt wird, aber nicht erkannt wird, daß sie nicht vom eigentlichen Kommunikationspartner stammt. D. h. auch diese unberechtigte Nachricht würde zu entsprechenden Veränderungen in unserem Suchbaum führen, so daß ein Lesen von weiteren Nachrichten unmöglich wäre. Aus diesem Grund sieht das hier beschriebene Verfahren eine Authentifikation des Senders vor. Dabei soll kein fester Schlüssel verwendet werden, den ein Dritter durch Beobachtung erfahren könnte. Vielmehr soll die Dynamik des Suchbaums ausgenutzt werden, um eine Authentifikation durchzuführen.

Auf der Senderseite sieht der Vorgang folgendermaßen aus: Der Sender bestimmt per Zufall ein Wort aus seinem Suchbaum und ermittelt dessen zugehörige Tiefe. Diese beiden Daten werden der zu übermittelnden Nachricht hinzugefügt und stellen den Message-Authentification-Code dar.

$$MAC \quad = \quad Zufallswort + Tiefe(Zufallswort)$$

Auf der Empfängerseite versucht man die Nachricht auf normalem Wege zu dekodieren. Ist dieser Vorgang erfolgreich, betrachtet man die Authentifikationsdaten, die in der Nachricht enthalten sind. Dazu bestimmt man im eigenen Suchbaum die Tiefe des Zufallswortes und vergleicht sie mit der Tiefe, die der Sender angegeben hat. Die beiden Werte sollten nur dann übereinstimmen, wenn Sender und Empfänger einen identischen Baum als Grundlage der Authentifikation benutzt haben.

3.6.2 Empfangsbestätigung

Bei den bisherigen Betrachtungen wurde stets davon ausgegangen, daß die Nachrichten störungsfrei zum Empfänger gelangen. Eine Störung der Übertragung hätte aber schwerwiegende Folgen für den gesamten Kommunikationsablauf, noch dazu wenn sie primär unerkannt bleiben würde, d. h. wenn die Dekodierung keinem nicht existenten Pfad verfolgen würde.

Daher ist es zwingend notwendig, die Korrektheit der Übertragung zu überprüfen und auch dieses Prüfergebnis dem Sender der Nachricht mitzuteilen, damit dieser weiß, ob er die gesendete Nachricht enventuell wiederholen muß. Auch muß der Empfänger feststellen können, ob die Nachricht komplett übertragen wurde. Dieses 'Die Nachricht ist zu Ende'-Signal kann man z. B. durch einen Verweis auf ein festdefiniertes Endezeichen im Suchbaum realieren, das selbstverständlich auch rotiert wird, so daß das Ende jeder Nachricht für einen externen Betrachter anders aussieht. Das Kommunikationsprotokoll sollte deshalb das allgemeine Aussehen einer Nachricht definieren.

Damit der Sender nicht jedesmal auf eine Empfangsbestätigung warten muß, muß er sich die Suchbäume nicht bestätigter Nachrichten zusammen mit den Nachrichtentexten aufbewahren, um gegebenenfalls eine oder mehrere Nachrichten wiederholen zu können.

3.6.3 Checksummen

Im vorletzten Abschnitt wurde zwar erklärt, daß es kein Problem darstellt, Störnachrichten durch einen einfachen Dekodierversuch zu erkennen. Es kann jedoch auch vorkommen, daß berechtigte Nachrichten falsch oder fehlerhaft übermittelt werden. Tritt dieser Fehler bei sehr großen Kodierfolgen erst relativ weit am Ende auf, so verschenkt man unnötig Zeit.

Um dieser Tatsache entgegenzuwirken, baut man in die kodierte 0/1-Folge Prffsummen ein, die Auskunft über die Anzahl und Art der übermittelten Bitfolgen geben. Sollten Werte durch die physikalische Übertragung verloren gegangen sein, so kann man dies durch einfaches Zählen feststellen, bevor man die komplette Nachricht dekodiert.

An dieser Stelle sei noch folgende Anmerkung gemacht:
Das ursprüngliche Vorhaben bestand darin, daß die kodierte Nachricht nur aus einer einzigen Zeile mit entsprechenden 0/1-Kombinationen bestehen soll. Tests, bei denen Mails unterschiedlicher Länge versandt wurden, haben jedoch ergeben, daß bei großer Zeilenlänge gehäuft Fehler auftreten, d. h. es gingen Nullen oder Einsen bei der Übertragung verloren. Reduziert man aber die Zeilenlänge der Nachricht, so treten diese Fehler nicht mehr auf. Aus diesem Grund wird die verschlüsselte Nachricht in Zeilen mit maximaler Länge von 80 Zeichen aufgeteilt. Ein Rückschluß auf Wortlänge oder die Gestatung des Suchbaumes ist dadurch trotzdem ausgeschlossen.

3.6.4 Numerierung der Nachrichten

Man kann sich leicht vorstellen, daß ein Dekodieren von Nachrichten entgegen der ursprünglichen Reihenfolge nicht sehr erfolgreich sein wird. Selbst wenn die angegebenen Pfade immer zu einem entsprechenden Eintrag im Suchbaum führen würden, so wäre der Gehalt der dekodierten Nachricht wohl eher fragwürdig.

Um einen solchen Effekt auszuschließen - es könnte ja schließlich durch eine reine Übertragungsverzögerung zu solch einem Fehler kommen - werden alle Nachrichten durchnumeriert und die Nummer dem Binärcode der Nachricht hinzugefügt.

Natürlich darf diese Numerierung nicht offensichtlich sein. Es wäre sonst ein leichtes für einen Gegener, andere Nummern einzufügen und so die komplette Kommunikation lahmzulegen, auch wenn er nichts über die Klartexte erfahren könnte. Man versteckt deshalb die binäre Darstellung der Nummer zwischen den Geheimtextsbits an einer vorher vereinbarten Stelle.

3.7 Speicherung des Suchbaumes

Man kann davon ausgehen, daß sich das Mailprogramm, mit dem das Kodierungsverfahren verbunden ist, nicht immer aktiv im Speicher eines Rechners befindet. Stattdessen müssen alle Einträge im Suchbaum beim Beenden des Mailprogrammes abgespeichert, bzw. nach dessen Start in den Suchbaum von neuem eingelesen werden.

Liegen die Einträge in alphabetischer Reihenfolge in der Datei vor, so entsteht beim Einlesen der Datei und Einfügen in den Suchbaum nach den Gesichtspunkten des Algorithmus ein degenerierter Baum, an dessen Wurzel z. B. nur ein 'a' steht und der nur aus einer Verzweigung oder vielmehr nur aus einem Zweig besteht. Neben einer schlechten Verschlüsselung hätte dies auch Suchzugriffe in linearer Zeit zur Folge.

Erst wenn mehrere Nachrichten basierend auf diesem Baum versendet wurden und die entsprechenden Rotationen durchgeführt wurden, entsteht ein Baum in der gewünschten Form.

Die Einträge in der Suchbaumdatei sollten daher nicht in alphabetischer Reihenfolge vorliegen, sondern vielmehr ungeordnet.

Man könnte diesen Effekt folgendermaßen erreichen:

- Neue Wörter werden einfach an die Datei in der Reihenfolge angehängt,

in der sie zum ersten Mal benutzt werden. D. h. die Einfügefunktion des Algorithmus würde nicht nur ein neues Blatt in den Suchbaum einfügen, sondern auch das neue Wort an die Datei anhängen.
Durch dieses einfache Anhängen ohne jegliche Vorrangbeziehung erhält man auf der einen Seite ein verbesserte Baumstruktur, jedoch basiert jede Kommunikation, die als ersten Schritt den Suchbaum aus der Datei einliest, auf einem fast identischen Baum. Der Baum würde sich höchstens in den zuletzt eingefügten Wörtern unterscheiden. Auch käme es zu Differenzen in den Suchbäumen zweier Kommunikationspartner, wenn ein Teilnehmer seinen Baum zwischenspeichert und der andere nicht.

- Eine weitere Lösung dieses Problems stellt folgendes Verfahren dar: Sollen die Bauminhalte gespeichert werden, so wird der Baum in einer gewissen Art und Weise (siehe Beispielabbildungen) durchmustert. Dabei soll nicht auf den Inhalt der einzelnen Einträge geachtet werden. Vielmehr wird nur der Struktur des Baumes gefolgt.
  Dadurch erreicht man, daß die Reihenfolge Einträge, die sich nach jeder Nachricht ändert, mit abgespeichert wird. Das Speichern des Baumes erhält also die Baumstruktur.

Beispiele für mögliche Durchmusterungen:

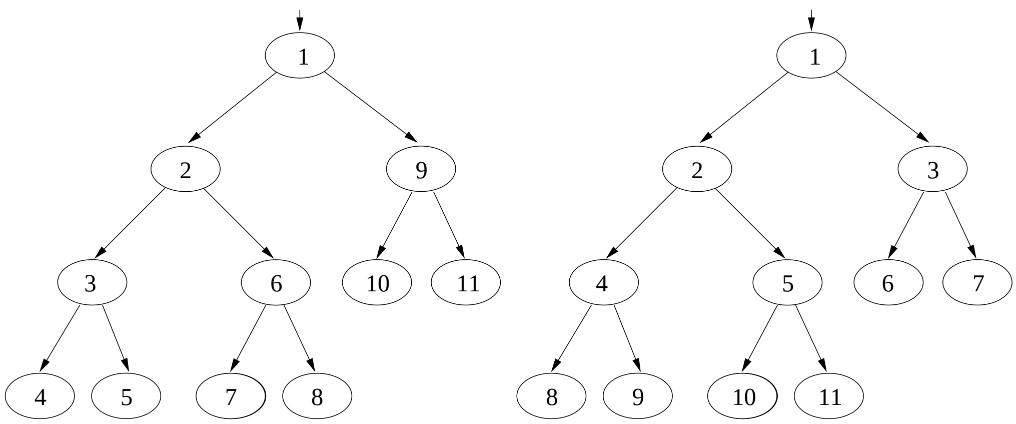


Abbildung 3.5: Beispiel für mögliche Durchmusterungen

Durchmustert man den Baum wie in der rechten Abbildung gezeigt und speichert die Einträge in dieser Reihenfolge ab, so bleibt gewährt, daß durch ein einfaches Einfügen der Wörter in der lexikographischen Ordnung, wie es das Verfahren vorschreibt, der gleiche Baum wie vorm Abspeichern aufgebaut wird, so daß die Kommunikation weiterlaufen kann, als wäre sie nie unterbrochen worden.

3.8 Analyse des Verfahrens

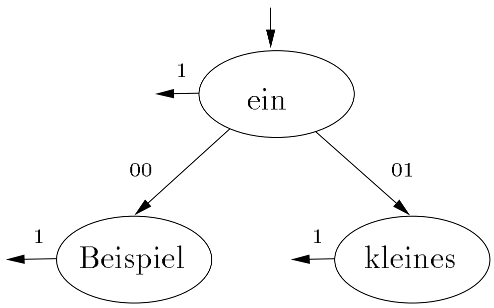
3.8.1 Sicherheit des Verfahrens

In diesem Abschnitt soll die Sicherheit des eigentlichen Algorithmus betrachtet werden, d. h. von Mängeln am in 3.3.2 vorgeschlagenen Protokoll oder ähnlichen Sachverhalten soll hier abgesehen werden.

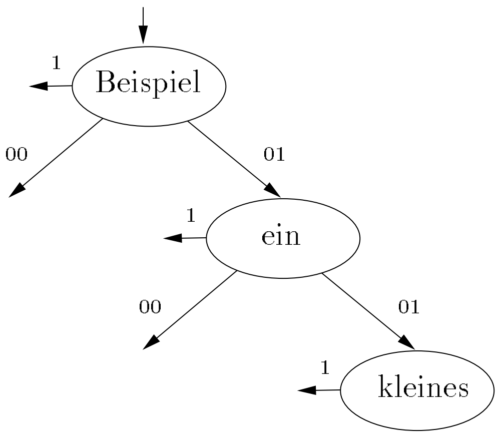
Der am weitesten reichende Angriff auf ein Kryptosystem stellt ein Brute-Force-Angriff dar. Bei dieser Art Angriff wird vom eigentlichen Verschlüsselungsverfahren im Grunde genommen komplett abgesehen und stattdessen werden einfach alle möglichen Schlüssel eines Algorithmus getestet, bis der entsprechende Klartext eines Known-Plaintext-Angriffs gefunden wurde. Die Schwierigkeit besteht darin, daß mit wachsender Schlüssellänge die Komplexität des Angriffs steigt. So gilt es z. B. bei einem für das DES-Verfahren gängigen Schlüssel von 56 Bit Länge 2^{56} verschiedene Schlüssel auszuprobieren.

Die Frage stellt sich nun, wie sicher das hier beschriebene Verfahren gegen Brute-Force-Angriffe gewappnet ist. Wieviele verschiedene Schlüssel müßte man testen, um zum richtigen Ergebnis zu gelangen?

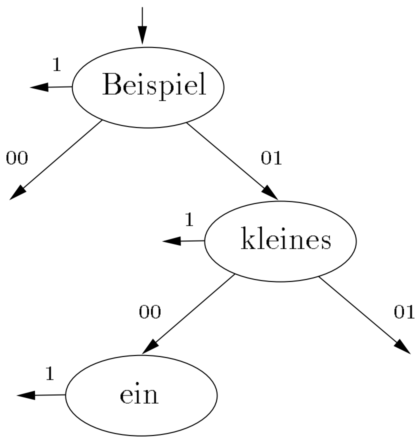
Den Schlüssel in diesem Verfahren stellt der Baum dar. Von seiner Anordnung, genauer gesagt von der Anordnung der Einträge im Suchbaum, hängt der erzeugte Geheimtext ab. Dies soll die folgende Abbildung anhand der möglichen Alternativen für den Klartext „ein kleines Beispiel“ verdeutlichen:



$\Longrightarrow$ *Geheimtext* $\quad c = 1011001$



$\Longrightarrow$ *Geheimtext* $\quad c = 011010111$



$\Longrightarrow$ *Geheimtext* $\quad c = 010010111$

Abbildung 3.6: Mögliche Geheimtexte bei alternativer Anordnung der Einträge im Baum. Die Möglichkeiten insgesamt betragen $C_3 = 5$.

Betrachtet man einen binären Baum als Schlüssel, so ergibt sich unter Beachtung der lexikographischen Ordnung als Anzahl der Anordnungsmöglichkeiten die Katalanische Zahl (siehe [CRO])

$$C_n \quad = \quad \frac{1}{n+1}\binom{2n}{n}, \text{ wobei n die Anzahl der Einträge ist.}$$

Für einen Angriff auf einen Grundbaum der Größe $n = 98$ müssen daher bereits etwa 2^{186} Möglichkeiten getestet werden, was zu den 2^{56} Schlüsselvarianten von DES schon recht beachtlich ist. Nimmt man nun einen Wortschatz, der für eine Kommunikation in natürlicher Sprache gedacht ist, so kommt man leicht auf eine Größenordnung von 250000 Einträgen. Das am Lehrstuhl installierte Wörterbuch von „ispell“ hat zum Beispiel einen Umfang von 236000 Wörtern. Hier werden natürlich auch Wortbeugungen und ähnliches mitgezählt, da diese einen eigenen Eintrag im Suchbaum darstellen. Man kann wohl ohne Einschränkung sagen, daß eine Anzahl von C_{250000} Anordnungsmöglichkeiten, was etwa 2^{499972} entspricht, einen Brute-Force-Angriff gänzlich ausschließt. Selbst wenn man davon ausgeht, daß ständig nur ein Teil der Wörter im Baum für die Kommunikation genutzt wird, bleibt ein solcher Angriff ausgeschlossen. In solch einem Fall würden nur die Blätter des oberen Suchbaumbereichs rotieren, während die Anordnung in den unteren Teilbäumen unverändert bliebe. Legt man eine so geringe Anzahl wie 10000 häufig benutzte Wörter zugrunde, so bleiben immer noch 2^{19979} verschiedene Schlüsselmöglichkeiten für einen Brute-Force-Angriff.

Durch die Art der Geheimtextgestaltung als Folgen von 00 und 01 Kombinationen mit einer abschließenden 1 könnte man bei sehr großen Nachrichten eine statistische Betrachtung über die Häufigkeit gewisser Bitfolgen und damit über die dargestellten Wörter machen. Jedoch gilt dies nur für sehr große Nachrichten und nur für häufig vorkommende Wörter wie zum Beispiel Artikel. Zum einen fragt man sich, ob der dadurch gewonnene Informationsgehalt sehr hoch ist, zum anderen fehlt jede Möglichkeit der Verifikation der Statistik, da die nächste Kommunikation auf einem geänderten Suchbaum beruht und sich somit sämtliche Verschlüsselungen, also auch der häufig benutzten Wörter, geändert haben. Die statistische Betrachtung müßte von vorne beginnen.

3.8.2 Zugriffsgeschwindigkeit

Die Verschlüsselung beim hier vorgestellten Verfahren läuft weder bitweise ab wie bei Stromchiffren noch blockweise wie bei Blockchiffren. Vielmehr wird Wort für Wort einer Nachricht verschlüsselt. Vorausgesetzt, das zu verschlüsselnde Wort ist im Baum enthalten, wird einfach in den Baum hinab gestiegen und an den jeweiligen Verzweigungen auf Gleichheit untersucht. Trifft dies zu, wird dem Geheimtext ein Bit in Form einer 1 hinzugefügt, andernfalls wird, falls der Suchbegriff lexikographisch kleiner als der Knoteneintrag ist, in den linken Unterbaum des aktuellen Knoten verzweigt, sonst nach rechts. Im ersten Fall wird der Geheimtext um die Beschriftung des linken Astes, also um 00 erweitert, im zweiten Fall um 01.

Der zugrundeliegende Suchbaum ist ein dynamischer binärer Baum. Dadurch ergibt sich nach Kapitel 2 im Mittel ein Zugriffszeit in Größenordnung der Entropie $H(A)$, wobei A für die Einträge im Baum steht. Problematisch bei der Berechnung der Zugriffszeit ist dabei, daß die Entropie $H(A) = -\sum p_i \log p_i$ abhängig ist von der Wahrscheinlichkeit des Auftretens der einzelnen Einträge. Eine Abschätzung dieser Wahrscheinlichkeiten hängt wiederum stark von der verwendeten Sprache, ja sogar vom Sprachgebrauch jedes einzelnen Kommunikationspartners ab. Geht man davon aus, daß jeder Eintrag im Baum gleich häufig verwendet wird, so ergibt eine Wahrscheinlichkeit von $\frac{1}{n}$, wobei n die Gesamtzahl der Einträge im

Suchbaum ist. In diesem Fall wäre die Entropie $H(\frac{1}{n}, ..., \frac{1}{n}) = \log n$ und die Suche im Baum wäre logarithmisch zur Anzahl der Einträge im Baum. Eine solche Gleichverteilung anzunehmen erscheint jedoch als wenig sinnvoll, da die Einträge auch sämtliche Sonderzeichen, Buchstaben und mit Sicherheit auch selten benutzte Wörter enthalten. Ein Teil aller Einträge im Baum wird also häufiger benutzt und ein Teil seltener. Wie groß dieser Anteil ist, müßte eine Untersuchung des Sprachgebrauchs ergeben. Dabei befindet sich der Teil der häufiger benutzten Einträge aufgrund der durchgeführten Rotationen näher zur Wurzel. Man erhält in diesem Fall eine mittlere Zugriffszeit der Größenordnung $H(A) \leq \log n$ oder besser.

Selbst das Einfügen neuer Einträge kann mit diesem Aufwand erfolgen, da das Einfügen auf die gleiche Art und Weise wie das Suchen funktioniert. Man sucht das nicht vorhandene Wort und fügt es einfach am Endpunkt der Suche ein. Soll ein solches Wort übermittelt werden, tritt ein Sonderfall auf. Jetzt müssen die einzelnen Buchstaben des neuen Wortes übertragen und daher zuerst im Baum gesucht werden. Sie treten jedoch genauso im Suchbaum auf, wie alle anderen Einträge, da sie bereits im Grundbaum enthalten sind (3.4). Der Suchaufwand für ein neues Wort u nimmt somit im Mittel $log|u|$ in Anspruch.

Als zusätzlicher Aufwand für das Verfahren kommt das Umordnen des Suchbaums auf Basis der zuletzt versendeten Nachricht hinzu, was aber nicht den eigentlichen Zeitaufwand für die Verschlüsselung darstellt. Die Rotation eines Knotens zur Wurzel kann dabei als Abfolge von Rotationen zwischen je zwei Knoten angesehen werden. Hierfür ist jedoch nur ein konstanter Aufwand notwendig, da nur eine konstante Zahl von Zeigern der einzelnen Knoten neu belegt wird. Die Anzahl der Rotationen wird bedingt durch die jeweilige Tiefe des zu rotierenden Knotens und ist somit in der Größenordnung eines Zugriffs möglich.

3.9 Kompression

Die Verwaltung der zu verschlüsselnden Wörter in einem binären Baum sorgt dafür, daß die Anwendung des hier beschriebenen Verfahrens als Nebeneffekt zu einer Kompression des Klartextes im Geheimtext führt. Die Angriffspunkte für eine Kryptoanalyse, die auf Redundanz der Sprache zurück geht, werden dadurch vermindert.

Jedes Wort der Nachricht wird im Geheimtext durch seinen Pfad im Suchbaum dargestellt. Die Pfadlängeentspricht der Anzahl der Vergleichoperationen bis der entsprechende Eintrag gefunden wurde und damit der Tiefe des Wortes. Für die Anzahl der Vergleichsoperationen zur Suche eines Wortes u und damit auch für die Länge gilt nach Abschnitt 3.8.2

$$|u| \leq \log n, \text{ wobei } n \text{ die Anzahl der Einträge im Suchaum ist.}$$

Damit ist die Darstellung eines Wortes u im Geheimtext unabhängig von seiner Länge $|u|$ im Klartext, was nebenbei auch zu einer Verschleierung führt. Kommt man auf das Beispiel mit 250000 Einträgen zurück und geht man von einem optimal balancierten Baum aus, so beträgt die mittlere Zahl der Vergleiche etwa 19. Durch die im Verfahren verwendete Doppelbezeichnung der Verzweigungen und durch das zusätzliche Kontrollbit liegt die durchschnittliche Darstellung für ein Wort bei 39 Bit. Legt man als Vergleichsgröße die ASCII-Darstellung von 8 Bit pro Zeichen zu Grunde, so entspricht dies einem Wort in natürlicher Sprache von ca. 5 Zeichen. Durch die Rotationen der häufig verwendeten Einträge im Suchbaum an dessen Wurzel ergibt sich für den gesammten oft genutzten Wortschatz eine noch bessere Kompression.

3.9.1 Einsatz des Verfahrens

Das hier beschriebene Verschlüsselungsverfahren beruht darauf, daß zwei Kommunikationspartner über identische binäre Bäume verfügen, auf deren Grundlage die Ver- und Entschlüsselungen durchgeführt werden.

Aus dieser Tatsache resu ltiert, daß ein einzelner Nutzer für jeden einzelnen Kommunikationspartner einen eigenen Baum mit einer zum Teil erheblichen Anzahl an Einträgen verwalten muß. Der Gedanke liegt daher nahe, das Verfahren insbesondere für die Kommunikation zwischen Gruppen einzusetzten, wobei jedes Gruppenmitglied auf den selben Baum zugreift.

Besonderheiten, die bei dem Einsatz des Verschlüsselungsverfahrens in dieser Art beachtet werden müssen, sollen im Folgenden betrachtet werden:

- Ein vom Benutzter unabhängiges System überwacht den Mailverkehr, d. h. der eigentliche Sender der Nachricht ist nicht für die Kodierung zuständig. Er schreibt vielmehr seine Nachricht wie gewohnt mit einem Mailprogramm und sendet diese ab.
- Ein Überwachungsmechanismus erkennt an der Empfängeradresse die Zugehörigkeit des Adressaten zu einer Gruppe und wählt den entsprechenden Suchbaum in der aktuell benötigten Form aus und verschlüsselt die Nachricht.
- Darüberhinaus erhält jede Nachricht an die selbe Gruppe eine Nummer, um die Reihenfolge der Versendung und damit auch der Entschlüsselung auf Empfängerseite festzuhalten.

- Die Korrektheit der Übertragung muß vom Empfängersystem automatisch überprüft werden können. Hierzu soll ein Verfahren mit Prüfsummen-Bildung genutzt werden, das schon in einem vorangegangenen Abschnitt beschrieben wurde.

- Die Rotationen der Einträge im Suchbaum anhand des Nachrichtentextes finden sofort nach dem Versenden der Nachricht statt, damit gewährt bleibt, daß die Nachricht des nächsten Senders mit einem neuen, d. h. neugeordneten Baum verschlüsselt wird. Der alte Suchbaum wird solange zusammen mit der entsprechenden Nachricht aufbewahrt, bis vom Empfänger ein endgültiges OK über den Empfang der Nachricht eingetroffen ist.

- Auf Empfängerseite mußsen nach erfolgter Authentifikation und Reihenfolgebestimmung sofort die Rotationen stattfinden, auch wenn der eigentliche Empfänger innerhalb der Gruppe die Nachricht in Klartext noch nicht gelesen hat. Voraussetzung hierfür ist natürlich, daß kein erkennbarer Übertragungsfehler vorlag.

Als mögliche praktische Anwendung kann man zum Beispiel die Kommunikation zwischen Autoherstellern und Zulieferern nennen. Die Produktionsabläufe auf diesem Gebiet folgen oft der Just-in-Time-Steuerung, die einen schnellen und papierlosen Dokumentenaustausch voraussetzt. Ein System, das auf diesem Gebiet eingesetzt wird ist EDIFACT (Electronic Data Interchange for Administration, Commerce and Transport). Dabei könnte jedes Dokument, das zwischen den beiden Partnern ausgetauscht würde, mit dem gleichen Suchbaum verschlüsselt werden, gleichgültig, von welcher Person es kommt oder an wen es gerichtet ist.

3.9.2 Die Problematik der dynamischen Schlüssel

Die Dynamik des Verfahrens wird hervorgerufen durch die automatische Anpassung der Suchbäume an die fortlaufende Kommunikation. Durch die Rotationen der Wörter einer Nachricht an die Wurzel des Suchbaumes entsteht nach jeder Nachrichtensendung bzw. nach jedem Nachrichtenempfang ein neuer Sitzungsschlüssel, der dann für die nächste Kommunikation genutzt wird.

Es wurde bereits erwähnt, daß nach jedem Versenden einer Nachricht der zu ihr gehörende Suchbaum zusammen mit dem Nachrichtentext aufbewahrt werden muß, bis eine Eingangsbestätigung vom Empfänger eintrifft, um bei einer eventuellen Kommunikationsstörung diese Nachricht und die darauf folgenden zu wiederholen. Die Folgenachrichten müssen ebenfalls wiederholt werden, da die notwendigen Rotationen nach der fehlgeschlagenen Kommunikation nicht durchgeführt werden konnten. Nach dem Versenden der Nachricht wird der Suchbaum auf Basis des Nachrichtentextes neu geordnet, um eine neue Nachricht versenden zu können.

Auch die folgenden Situationen könnten zu einer Störung der Kommunikationsabläufe führen: A und B seien zwei Kommunikationspartner.

1. Teilnehmer A sendet Teilnehmer B die Nachricht S_A^1. Diese wird während des Transports verzögert. B sendet A die Nachricht S_B^1, bevor Nachricht S_A^1 bei ihm angekommen ist.
   $\Longrightarrow$ A kann S_B^1 nicht entschlüsseln und B kann S_A^1 nicht entschlüsseln, da beide von Suchbäumen in einer falschen Anordnung ausgehen.

2. Teilnehmer A sendet Teilnehmer B n Nachrichten $S_A^1, \ldots, S_A^n$. B sendet A zum Zeitpunkt t eine Nachricht S_B^1. Zum Zeitpunkt t hat A gerade S_A^k, $k < n$, verschickt und S_B^1 sollte eintreffen, bevor S_A^{k+1} verschickt wird, doch S_B^1 geht verloren und kommt bei A nie an.
   $\Longrightarrow$ B kann die Nachrichten ab S_A^{k+1} nicht mehr entschlüsseln und wird dies A mitteilen. Die zwei möglichen Reaktionen von A sind:

   - A versendet alle Nachrichten von $k + 1$ bis n nocheinmal auf Basis der mit ihnen abgespeicherten Suchbäume. Dann mußs B mit seiner Nachricht S_B^1 warten, bis alle n Nachrichten von A angekommen sind.
   - B wiederholt die Sendung seiner Nachricht S_B^1 an A. A entschlüsselt S_B^1 mit Hilfe des Suchbaums, der auf S_A^k basiert. Danach versendet er seine Nachrichten $k+$ bis n. Hierzu kann er jedoch nicht die mit diesen Nachrichten abgelegten Suchbäume verwendeten, da die Rotationen auf Basis von S_B^1 in den Kommunikationsablauf eingeflossen sind.

3. Es ließen sich noch viele Beispiele dieser Art konstruieren, doch sollte die Problematik klar geworden sein.

Eine Lösung dieses Problem wäre eine Aufteilung des Suchbaums in einen Versendungssuchbaum und einen Empfangssuchbaum. Dadurch, daß Verschlüsseln und Versenden auf einem anderen Suchbaum basieren als Empfangen und Entschlüsseln, könnten solche Situationen wie in den obigen Beispielen nicht mehr auftreten.

Wenn man aber zwei Suchbäume pro Kommunikationspartner verwalten muß, stellt sich gleich die nächste Frage: Welchen Speicherplatz benötigt man zur Durchführung des Verfahrens?

Geht man von einem Suchbaum mit 300000 Einträgen aus und einer Wortlänge von 6 Zeichen als geschätztem Wert, so erhält man zusammen mit ein paar Organisationsbytes pro Eintrag einen Gesamtspeicherbedarf in einer Größenordnung von 3 MB im Arbeitsspeicher. Nimmt man die Neuerung aus dem ersten Teil dieses Abschnitts hinzu, daß man pro Teilnehmer je einen Suchbaum zum Versenden und einen Suchbaum zum Empfangen verwalten muß, so kommt man bei einer Zahl von 100 Kommunikationspartner auf einen Speicherbedarf von ca. 600 MB. Den Speicherbedarf könnte man etwas reduzieren, indem man in den Einträgen nicht mit den ganzen Wörten arbeitet, sondern nur mit Verweisen auf eine zentrale Haltung aller Einträge in lexikographischer Ordnung. Beim Abspeichern der Suchbäume könnte man analog verfahren.

3.10 Die X11-Umgebung

Das Programm wurde in eine X11-Umbebung eingebettet. Selbstverständlich ist die Implementierung des eigentlichen Algorithmus unabhängig davon. Man könnte den Algorithmus auch mit jedem anderen geläufigen Mailprogramm einsetzen. So wäre eine Anbindung an Xmail oder gar eine Integration in Netscape denkbar.

Die hier verwendete Form einer X11-Darstellung dient einzig und allein der besseren Handhabung des Programms. Sie soll daher auch nicht näher beschrieben werden.

Zur Funktion sei nur folgendes gesagt:

Nach dem Start des Programms wird automatisch eine Initialisierungsdatei eingelesen und deren Einträge in einem Suchbaum angeordnet, der dann für die Kommunikation genutzt wird.

Das Programm erlaubt das Öffnen, Editieren und Versenden von Nachrichten. Beim Versenden wird die Nachricht mit Hilfe des Suchbaumes kodiert und an den Empfänger gesandt.

Der Empfänger wählt einfach die eingegange Nachricht aus. Diese wird dann dekodiert.

Sowohl nach dem Senden als auch nach dem Empfang werden auf Sender- und Empfängerseite die Suchbäume entsprechend rotiert, so daß in der Kommunikation fortgefahren werden kann.

Beim Verlassen des Programms werden die Suchbäume automatisch abgespeichert und dienen beim nächsten Start der Initialisierung.

Zusammenfassende Betrachtungen

Das hier vorgestellte Verfahren scheint, soviel kann man am Ende dieser Arbeit sagen, sicher zu sein. Ob und in welcher Weise es doch angegriffen werden kann, wird wohl eine Kryptoanalyse zeigen. Zu bedenken ist dabei, daß ein Brute-Force-Angriff wegen der in 3.8.1 vorgestellten Komplexität auszuschließen ist. An dieser Stellt drängt sich einem sogar ein Vergleich mit dem One-Time-Pad-Verfahren auf, dessen Sicherheit auf der unendlichen Länge des verwendeten Zufallsschlüssels beruht.

Ein direkter Vergleich mit anderen Kryptosystemen fällt an dieser Stelle schwer, da die Art der Verschlüsselung vollkommen verschieden ist zu den bekannten Strom- und Blockchiffren. Ein Geschwindigkeitsvergleich zu Verfahren wie RSA oder DES müßte ein Praxistest ergeben. Betrachtet man nur die Zahl der Vergleiche in einem Suchbaum für das Auffinden eines Wortes und somit seiner Kodierung, so liefert das Verfahren gute Ergebnisse.

Der Aufwand für die Verwaltung der Suchbäume macht das Verfahren vorallem für den Einsatz bei der Kommunikation zwischen Gruppen interessant. An dieser Stelle wünsche ich mir, daß das Verfahren in solch einem Rahmen in einer praxisnahen Form umgesetzt wird, die noch weit über das Maß meiner Implementierung hinausreichen würde.

Literaturverzeichnis

[BET] Thomas Beth, Peter Heß, Klaus Wirl. Kryptographie. Teubner, 1983

[BEU] Albrecht Beutelspacher. Kryptologie. Vieweg, 1991

[CRO] T. Cormen, C. Leiserson, R. Rivest. Introduction to Algorithms. MIT Press, 1992

[DIF] W. Diffie, M. E. Hellman. New Directions in Cryptography. IEEE Transacitons of Information Theory, 1976

[HOT] Hotz Günter. Algorithmische Informationstheorie. Technical Report: Vorlesung WS 1996/97, 1997

[KNU] L. R. Knudsen. Block Ciphers - Analysis, Design, Applications. Aarhus University, 1994

[MEH] K. Mehlhorn. Datenstrukturen und effiziente Algorithmen. Teubner-Verlag, 1988

[SCH] Bruce Schneier. Angewandte Kryptographie. Addison-Wesley, 1996

[SHA] C. E. Shannon. Communication Theory of Secrecy Systems. Bell System Technical Journal, 1949