Tutoriumsblatt 8 mit Musterlösung

Aufgabe 8.1: TCP: Fluss- und Staukontrolle

Erläutern Sie die Gemeinsamkeiten/Unterschiede von Fluss- und Staukontrolle bei TCP.

a) Welches *Problem* versuchen beide zu beheben?

b) An welchen *Stellen* tritt das Problem jeweils auf?

c) Wie *erfährt* der Sender, dass er seine Datenrate durch die *Flusskontrolle* reduzieren muss?

d) Wie verhält es sich im Fall von *Staukontrolle*?

Lösung 8.1

1.a) Beide versuchen, Überlastungen zu vermeiden.

1.b) Der Unterschied liegt in der Stelle der Überlastung: bei der Flusskontrolle wird die Überlastung des Empfängers vermieden, bei der Staukontrolle die der Router im Netz.

1.c) Bei der Flusskontrolle teilt der Empfänger dem Sender explizit über das Empfangsfenster mit, wenn ein Verarbeitungsstau auftritt (Mitsenden der aktuellen Fenstergröße – d.h. der Größe des freien Bufferspeichers des Empfängers – in jedem Segment). Der Empfänger reduziert das Empfangsfenster (und damit die Anzahl der möglichen ohne Bestätigung ausstehenden Segmente) entsprechend seines freien Bufferspeichers, damit dieser nicht überläuft.

1.d) Bei der Staukontrolle muss der Sender anhand von fehlenden Bestätigungen erkennen, dass es zu Übertragungsproblemen kommt – es gibt keine explizite Benachrichtigung, da eine Überlastung der Router auf Schicht 3 stattfindet (IP) und aufgrund der Trennung der Schichten keine entsprechende Meldung an TCP (Schicht 4) auf den Endsystemen möglich ist. TCP geht daher einfach bei Verlust eines Segments davon aus, dass es durch einen Pufferüberlauf des Netzwerks verlorengegangen ist und Stau vorliegt. Die Stausituation wird behoben, indem der Sender einen Algorithmus verwendet (z.B. Slow Start), der die Anzahl der möglichen ausstehenden unbestätigten Pakete (das Congestion Window) und damit in Folge auch die Netzlast reduziert.

(Dieses Vorgehen ist eine Art Notlösung. Es ist nicht optimal, da eine fehlende Quittung nicht unbedigt auf eine Überlast im Netz hindeutet, sondern z.B. auch einfach auf einen Übertragungsfehler im Netz zurückzuführen sein kann. So etwas kommt zwar nur selten vor, aber in drahtlosen Netzen könnte durch Interferenz auf dem drahtlosen Link eine Übertragung für ein paar Sekunden unmöglich sein, was zu jeder Mege Timeouts und zu einer Fehlinterpretation der Situation führt.)

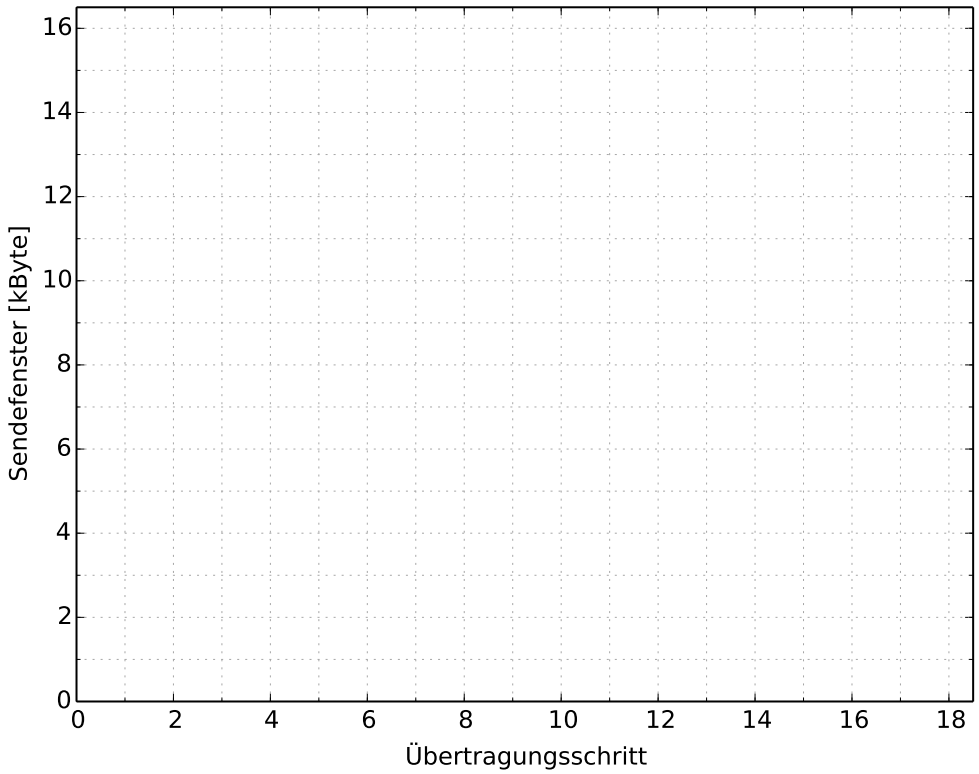
Was man auch noch erwähnen kann: Erst in letzter Zeit hat man eine Verletzung des Schichtenkonzepts eingeführt, um das TCP-Verhalten zu verbessern: eine Signalisierungsmöglichkeit, über die betroffene Router via IP eine Überlast an TCP melden können. Router verwerfen sowieso bereits seit längerem Pakete, auch wenn gar keine Überlastsituation vorliegt (Random Early Detection). Sobald die Queue des Routers einen bestimmten Schwellenwert überschreitet, fängt der Router an, mit gewisser Wahrscheinlichkeit zufällig Pakete zu verwerfen, um die Congestion Control bei den Sendern zu triggern, damit die Queue nicht noch weiter vollläuft. Die Neuerung in TCP (und IP) ist, dass ein Router nun auch Pakete weiterleiten kann, die er eigentlich verworfen hätte, aber ein Flag setzt, welches anzeigt, dass dieses Paket eigentlich gelöscht gehört hätte (Explicit Congestion Notification). Die empfangende IP-Instanz wird dieses Wissen an TCP weitergeben, welches daraufhin das Congestion Window reduziert. Aber in der Praxis ist dieser Mechanismus noch kaum implementiert...

Aufgabe 8.2: TCP Congestion Control

a) Beschreiben Sie knapp den *Slow-Start-Mechanismus* bei TCP. Warum wird ein *Schwellenwert* (Threshold) verwendet?

b) Eine TCP-Verbindung nutze den Slow-Start-Algorithmus mit einem Schwellenwert (Slow-Start Threshold, `ssthresh`) von anfangs 16 kByte. Die MSS sei 1 kByte, die Window Size des Empfängers 24 kByte.

Stellen Sie dar, wie sich die Datenrate in diesem Szenario ändert. *Zeichnen Sie für die Übertragungsschritte 1 bis 18 jeweils die erreichte Übertragungsrate (ausgedrückt über die Größe des Sendefensters `swnd`) sowie den Threshold in das folgende Diagramm ein.*



Als ein Übertragungsschritt werde hier die Versendung der möglichen Datenmenge samt Empfang der zugehörigen Quittungen bezeichnet; wurden alle Quittungen des aktuellen Übertragungsschrittes erhalten, wird im nächsten Übertragungsschritt wieder die gesamte nun mögliche Datenmenge versendet. Bei Übertragungsschritt 5 und 13 findet jeweils ein Timeout statt, der vom Sender als Netzüberlastung interpretiert wird.

c) In der Realität werden Slow-Start und Congestion Avoidance nicht alleine eingesetzt. Gängige Erweiterungen sind *Fast Retransmit* und *Fast Recovery. Welchen Zweck haben diese Erweiterungen*?

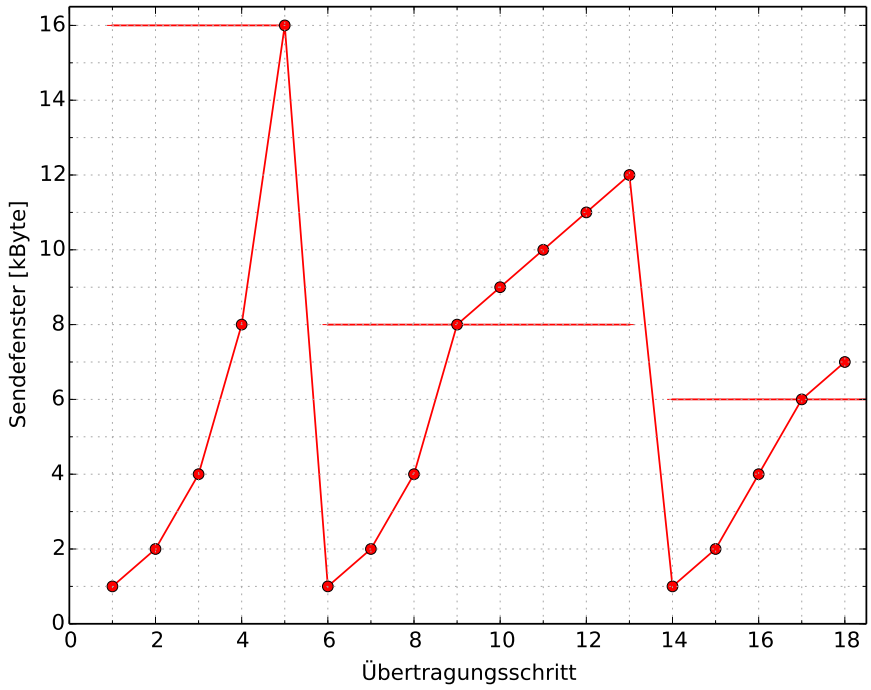
Lösung 8.2

2.a) Slowstart: übertrage zunächst nur ein Segment maximaler Größe. (Nebenbemerkung: ja, mittlerweile sind es abhängig vom Betriebssystem auch mehr. Diese Hochsetzung des initialen Werts verdoppelt in der Slow-Start-Phase den Durchsatz und bringt in der Praxis eine merkbare Effizienzsteigerung mit sich. Aber wir gehen der Einfachheit halber immer von einer initialen Windowsize von 1 MSS aus.)

Falls vor einem Timeout eine Quittung ankommt, übertrage zwei weitere Segmente. Übertrage nun für jede ankommende Quittung zwei weitere Segmente, d.h. in jedem 'Schritt' erfolgt eine Verdopplung der Zahl der übertragenen Segmente und so eine exponentielle Steigerung der Datenrate.

Threshold: soll verhindern, dass man irgendwann zu große Steigerungen vornimmt und damit das Netz spontan überflutet. Ab hier erfolgt nur noch eine lineare Steigerung: können aktuell laut `cwnd` n Segmente versendet werden, wird für jede empfangene Quittung ein neues Segment übertragen, und nach Empfang der n-ten Quittung noch ein weiteres. Dann wiederholt man das ganze für $n+1$, $n+2$, ...).

2.b)



Anmerkung: die 'Schritte' werden hier nur verwendet, um das Ganze übersichtlich darstellen zu können. In der Realität werden in der Slow-Start-Phase für jede Quittung direkt zwei neue Segmente versendet (siehe Folie V-55), man wartet nicht, bis man alle ausstehenden Quittungen empfangen hat und sendet erst dann die doppelte Menge Segmente los. Bei korrekter Darstellung wäre der Verlauf des Sendefensters zwar so ziemlich der gleiche, aber mit flatternden Linien.

Und hier kann man mal die Frage stellen, was in Schritt 17 passieren würde: in Schritt 14 sind wir wieder bei 1, in Schritt 15 bei 2, in Schritt 16 bei 4, in Schritt 17 bei 8, aber moment – `ssthresh` ist 6. Verdoppeln wir trotzdem bis 8 oder gehen wir nur bis 6???

Die Antwort ergibt sich aus der vorherigen Anmerkung: wir gehen nur bis 6. Denn hier haben wir idealisierte Sendeschritte. In der Realität würden wir, solange wir the Threshold noch nicht erreicht

haben, immer zwei Segmente pro empfangener Quittung losschicken (das Congestion Window also um eine MSS erhöhen), und sobald wir dadurch den Threshold erreichen, wird umgeschaltet und wir zählen nun ab, bis wir n Quittungen haben, bevor wir ein weiteres zusätzliches Segment verschicken.

2.c) Das Rückfallen auf ein Segment ist nicht effizient, falls nur ein oder wenige Segmente verloren gehen (was nicht unbedingt Überlast bedeuten muss). Die Erweiterungen sollen vermeiden, dass bei einem solchen Einzelverlust direkt die Datenrate in den Keller geht – man reduziert zwar die Datenrate, aber nicht so drastisch wie bei Slow-Start.

Fast Retransmit: der Sender wird über DUP-ACKs angewiesen, die Neuübertragung eines Segments schon vor dem Timeout durchzuführen und den Threshold auf die Hälfte des aktuell erreichten Werts (Anzahl momentan übertragener Segmente) zu setzen. Das cwnd wird auf den Threshold gesetzt. Kommt die zugehörige Quittung, bevor der Timeout uns zum Neubeginn des Slow Start zwingt, vermeidet man das drastische Einbrechen der Datenrate. Man hat das cwnd auf den Threshold zurückgesetzt, sendet also mit deutlich höherer Datenrate weiter als bei Rückfall auf ein Segment. Der Rückfall auf ein Segment erfolgt nur, falls doch ein Timeout auftritt.

Fast Recovery: das Problem von Fast Retransmit ist, dass durch die Reduktion des cwnd meistens bereits mehr unbestätigte Segmente unterwegs sind, als das reduzierte cwnd erlauben würde – der Sender wird gebremst und kann nichts mehr senden, während er auf die Quittung des wiederholten Segments wartet. Fast Recovery erhöht während des Wartens auf die Quittung das cwnd für jedes empfangene DUP-ACK um 1. Durch die kontinuierliche Vergrößerung ermöglicht man es dem Sender, doch weiter zu übertragen. Sobald das cwnd groß genug wird, kann der Sender für jedes erfolgreich übertragene Segment direkt ein neues versenden.

Aufgabe 8.3: Sicherheitsziele

a) Was ist der Unterschied zwischen *Authentifizierung* und *Autorisierung*?

b) Was sind die Unterschiede zwischen *Vertraulichkeit* und *Integrität*?

c) Ist *Vertraulichkeit auch ohne Integrität* möglich? Ist *Integrität auch ohne Vertraulichkeit* möglich? Begründen Sie Ihre Antwort.

d) Objekte im Internet (Switche, Router, Web-Server, Benutzer-Endsysteme, usw.) müssen häufig in der Lage sein, sicher miteinander zu kommunizieren. Geben Sie *zwei Beispiele* von Objekten an, die eventuell sicher miteinander kommunizieren wollen. Geben Sie auch jeweils an, *welche Sicherheitsmaßnahmen sinnvoll* sind.

Lösung 8.3

3.a) Bei der **Authentifizierung** weist z.B. ein Nutzer seine Identität nach. Dies geschieht häufig durch die Eingabe eines Nutzernamens und eines dazugehörigen Passwortes. Nach der Authentifizierung weiß das System, mit wem es kommuniziert, und prüft, ob derjenige **autorisiert** ist, einen bestimmten Dienst zu nutzen (z.B. Dateien auf einem FTP-Server löschen). Beides dient also der Zugriffskontrolle.

(Und an sich gibt es auch die *Authentisierung*. Wenn ein Client bei einem Server seine Identität nachweisen soll, dann *authentisiert* er sich, z.B. durch Angabe von Nutzername und Passwort. Durch die Überprüfung dieser Identitätsinformationen *authentifiziert* der Server den Client. Authentisierung und Authentifizierung sind also die zwei Seiten des gleichen Prozesses. Wir verwenden illegalerweise für beides den Begriff Authentifizierung, da es im Englischen für beides nur einen Begriff (authentication) gibt.

3.b) **Vertraulichkeit** bedeutet, dass der originale Klartext einer Nachricht durch einen Angreifer, der den verschlüsselten Ciphertext der originalen Nachricht abgefangen hat, nicht wiederhergestellt werden kann. Wir verhindern also das Lesen unserer vertraulichen Daten.

Integrität bedeutet, dass ein Angreifer eine Nachricht (verschüsselt oder unverschlüsselt) während der Übertragung nicht unbemerkt verändern kann. Wir lassen also das Lesen zu, verhindern aber die Modifikation der Daten.

3.c) Ja, beides ist möglich. Protokolle können die Eigenschaften unabhängig voneinander oder auch nur einzeln bieten (oft wird aber beides zusammen angeboten).

Eine verschlüsselte Nachricht kann während der Übertragung verändert werden, aber ist dennoch vertraulich, da der Angreifer den Klartext nicht wiederherstellen konnte. Beispiel: wird ein Datenstrom durch AES verschlüsselt, kann der Angreifer einzelne Blöcke modifizieren/ersetzen. Der Empfänger wird versuchen, sie mit dem verwendeten Schlüssel zu entschlüsseln. Heraus kommt höchstwahrscheinlich Müll, so dass wir zwar keine Daten verarbeiten können, aber nicht wissen, ob dies durch einen Übertragungsfehler oder einen Angriff verursacht wurde. Im schlimmsten Fall kann ein Angreifer die Daten sogar unbemerkt sinnvoll modifizieren: wird AES ohne CBC oder ähnliche Techniken eingesetzt, wird jeder Block völlig unabhängig von anderen übertragen. Wenn wir die Struktur der Nachricht kennen, können wir durch gezielte Modifikation den Sinn der Nachricht ändern, indem wir einzelne Blöcke austauschen. Z.B. wird in einem IP-Paket die Absenderadresse immer an der gleichen Stelle stehen und immer gleich kodiert. Gleiches gilt für die Zieladresse. Also können wir z.B. einfach beides tauschen und damit das Paket zurück an den Absender umleiten. Die Integrität wird also verletzt.

Heißt also: ja, Vertraulichkeit geht auch ohne Integrität, wenn man die richtige Verschlüsselung benutzt, aber man sollte sich lieber nie darauf verlassen, dass es keine Angriffe auf die Verschlüsselung gibt, und immer auch Integrität mit sicherstellen.

Andersherum kann eine Nachricht zwar integritätsgeschützt sein, aber da sie im Klartext gesendet wurde, ist sie nicht vertraulich – wir signieren z.B. nur einen Hash der Nachricht.

Also wichtig: die beiden Eigenschaften bedingen sich nicht gegenseitig. Wird Verschlüsselung eingesetzt, realisiert man aber meist auf Integrität mit.

3.d) Hier sind nur zwei Beispiele aufgeführt. Es gibt wohl noch sehr viele andere sinnvolle Beispiele.

i) Ein Browser, der auf einem Laptop läuft und mit einem Web-Server kommuniziert. Der Nutzer möchte z.B. nicht, dass Dritte Informationen darüber erlangen, was er bei `www.google.de` als Suchanfrage eingetippt hat.

   Dazu muss erstmal gewährleitet sein, dass der Nutzer auch wirklich mit dem Web-Server von Google kommuniziert. Ansonsten könnte jemand sich als Google ausgeben und die Suchabfragen abfangen. Daher ist es notwendig, dass der Server sich zunächst gegenüber dem Nutzer authentifiziert. Darüber hinaus ist natürlich Verschlüsselung nötig, damit niemand unsere Suchanfrage belauschen kann.

   Oder Online-Banking: es darf sich kein Dritter in die Sitzung schalten oder Nachrichten belauschen. Nötig ist gegenseitige Authentifizierung (wir wollen sicher sein, mit dem Online-Banking-Server zu kommunizieren und der Server muss sicher sein, dass wir wir sind, damit er überhaupt Kontostände rausgibt oder Überweisungen akzeptiert). Darüber hinaus sollte Autorisierung erfolgen, d.h. der Server muss unseren Zugriff auf genau unser Konto einschränken, wir dürfen nicht auf alle Konten der Bank Zugriff haben. Bei der Übertragung von Kontoständen und Überweisungen sollte alles verschlüsselt sein, damit niemand unsere Kontobewegungen nachverfolgen kann, und auch Integritätsschutz ist nötig (damit z.B. niemand unsere Überweisungen umleitet).

ii) Zwei Router. Router tauschen z.B. Informationen über ihre Nachbarn aus. Es wäre allerdings sinnvoll, dass nicht jeder beliebige Routing-Informationen an die Router schicken kann, so dass diese falsche Einträge in ihren Tabellen erzeugen (Authentifizierung, Autorisierung, Integrität wie beim Online-Banking). Vertraulichkeit ist hier nicht notwendig, da die ausgetauschten Informationen zur Netzwerktopologie jedem bekannt sein dürfen bzw. durch `traceroute` oder ähnliches auch durch jeden mühselig ermittelt werden können.

Aufgabe 8.4: Shift Cipher

a) Alice sendet folgende verschlüsselte Nachricht an Bob:

$$c = \texttt{GSVSREWGLYXDZIVSVHRYRK.}$$

Sie haben die verschlüsselte Nachricht c mitgehört und möchten diese nun entschlüsseln. Sie wissen, dass Alice eine Shift Cipher verwendet hat und dass nur Großbuchstaben verwendet werden. Sie kennen sogar die Kodierungstabelle:

A	0	E	4	I	8	M	12	Q	16	U	20	Y	24
B	1	F	5	J	9	N	13	R	17	V	21	Z	25
C	2	G	6	K	10	O	14	S	18	W	22		
D	3	H	7	L	11	P	15	T	19	X	23		

Geben Sie den Schlüssel k an, der zur Verschlüsselung verwendet wurde, und entschlüsseln Sie die Nachricht c. Dokumentieren Sie, wie Sie vorgegangen sind, um k zu ermitteln.

Hinweis: Die entschlüsselte Nachricht ist ein deutsches Wort.

b) Gegeben sei nun die folgende Nachricht:

$$c = \texttt{XCYMYL MUNT BUN EYCHY VYMIHXYLY VYXYONOHA.}$$

Geben Sie ein Verfahren an, mit dem Sie, unter der Annahme, einen normalen deutschen Text vorliegen zu haben, bei dem die Satz- und Leerzeichen nicht verschlüsselt worden sind, die Nachricht *ohne Brute-Force-Angriff* entschlüsseln können. Wie lautet der Schlüssel k?

Lösung 8.4

a) Hier soll nichts besonderes gemacht werden. Einfach per Brute Force den Schlüssel finden. Wir wissen, dass es 26 Zeichen gibt, also ist $n = 25$, da wir mit 0 beginnen. Die Identische Abbildung können wir aussschließen, also betrachten wir $k = 0$ nicht. Es gibt also maximal 25 mögliche Schlüssel. Wir iterieren einfach über alle Schlüssel beginnend mit $k = 1$ und prüfen ob der Text ein deutsches Wort ist. Bei $k = 4$ ergibt die Entschlüsselung `CORONASCHUTZVERORDNUNG`. Alle anderen ergeben kein deutsches Wort. Daher ist der Schlüssel $k = 4$.

b) Einfache Chiffren wie die *Caesar-* oder auch die *Vigenère-Chiffre* sind anfällig für *Wörterbuch-* oder *Rainbow-Table-Attacks*, bei denen bekannte Wörter (bis zu einer bestimmten Länge) und ihre Verschlüsselungen vorberechnet und mit dem verschlüsselten Text abgeglichen werden, um den Schlüssel herauszufinden.

Zusätzlich kann bei bekannter Sprache auch eine *Buchstabenhäufigkeitsanalyse* eingesetzt werden, um Brute-Force-Angriffe zu vereinfachen. Die Buchstabenverteilung im Deutschen ist in etwa die Folgende (Angaben in Prozent) [1]:

A	6.5	E	16.4	I	6.6	M	2.5	Q	0.02	U	4.2	Y	0.04
B	1.9	F	1.7	J	0.3	N	9.8	R	7	V	0.9	Z	1.1
C	2.7	G	3	K	1.4	O	2.6	S	7.3	W	1.9		
D	5.0	H	4.6	L	3.4	P	0.7	T	6.2	X	0.04		

Zählen wir nun den häufigst auftretenden Buchstaben in unserem verschlüsselten Text, das `Y`, so stellen wir fest, dass es 9 mal vorkommt. Unser Text ist insgesamt 36 Zeichen lang (ohne Satzzeichen und Leerzeichen), sodass das `Y` eine Häufigkeit von

$$\frac{9}{36} = 25\%$$

besitzt. Da laut obiger Tabelle das E mit Abstand am häufigsten vorkommt – wenn auch nicht so häufig – ist die Wahrscheinlichkeit, dass das `Y` ein `E` ist, sehr hoch. Mit einem Schlüssel von

$$k = Pos(Y) - Pos(E) = 25 - 5 = 20$$

können wir folgenden Text ohne Brute-Force dekodieren:

$$c = \texttt{DIESER SATZ HAT KEINE BESONDERE BEDEUTUNG.}$$

[1] Verteilung aus: Beutelspacher, Albrecht: *Kryptologie*, 7. Auflage, Vieweg, Wiesbaden, 2005