Skriptum gemäß schulinternem Lehrplan

Grundkurs Informatik – Q2

Stand: 28. August 2016

Inhaltsverzeichnis

Literatur **99**

Vorwort

Mit diesem Skriptum legen wir ein Dokument für unsere Schülerinnen und Schüler vor, das es ermöglichen soll, den Unterricht vor- und nachzubereiten.

Die Arbeitsmaterialien, die verwendet werden, wurden in den vergangenen Jahren von Informatikreferendarinnen und Informatikreferendaren der Fachseminare Informatik an den Zentren für schulpraktische Lehrerausbildung (ZfsL) Hamm, Arnsberg und Solingen entwickelt und im Unterricht erprobt.

> **!** Elemente dieses Skriptums wurden der Veranstaltung »Einführung in die Informatik – Teil 1« aus dem Wintersemester 2012/2013 von Prof. Dr. Till Tantau entnommen.

Die Materialien stehen unter einer freien Lizenz (
– Erläuterung siehe unten) und sind zum größten Teil über die Webseite `http://ddi.uni-wuppertal.de/material/` öffentlich zugänglich (vgl. (Pieper und Müller 2014)).

Vorbereitung

Was brauchen Sie?

In unserem Unterricht werden Sie viele schriftliche Notizen erstellen – Sie erhalten Informationsblätter und Arbeitsblätter. Für den Fließtext, den Sie schreiben werden, benötigen Sie regelmäßig Kugelschreiber oder Füller. Ihre Aufzeichnungen müssen Sie auf kariertem DIN-A4-Papier vornehmen, da häufig auch kleine Skizzen anzufertigen sind. Dazu benötigen Sie zwingend einen angespitzten Bleistift (HB) und zwei farbige Stifte (grün und rot). Damit Skizzen vernünftig aussehen, benötigen Sie ein Lineal und ein Geodreieck. Sammeln Sie Ihre Unterlagen in einem Hefter, den Sie im Unterricht immer dabei haben und den Sie jederzeit abgeben können. Wir sammeln Ihren Hefter gelegentlich ein und ziehen Ihre Aufzeichnungen zur Bewertung Ihrer sonstigen Mitarbeit heran.

! Versehen Sie jedes Arbeitsblatt, jedes Informationsblatt und jede Kompetenzüberprüfung, die wir Ihnen geben, unverzüglich mit Ihrem Namen und heften Sie diese Materialien in Ihren Hefter, der ebenfalls mit Ihrem Namen gekennzeichnet werden muss.

Gegebenenfalls erhalten Sie durch uns einen geschützten Zugang zu Informatiksystemen. Account und Passwort für solche Zugänge dürfen keinesfalls an andere (auch nicht im Kurs) weitergegeben werden.

	read	**w**rite	e**x**ecute
user	yes	yes	yes
group	no	no	no
other	no	no	no

Vorhaben 1

Formale Grammatiken, Endliche Automaten und Reguläre Ausdrücke[1]

1.1 Welche Kompetenzen sollen Sie in diesem Vorhaben erwerben?[2]

Die Schülerinnen und Schüler

- analysieren und erläutern die Eigenschaften endlicher Automaten einschließlich ihres Verhaltens bei bestimmten Eingaben (IF3, A),
- ermitteln die Sprache, die ein endlicher Automat ***oder Kellerautomat*** akzeptiert (IF3, D),
- entwickeln und modifizieren zu einer Problemstellung endliche Automaten ***oder Kellerautomaten*** (IF3, M),
- stellen endliche Automaten in Tabellen oder Graphen dar und überführen sie in die jeweils andere Darstellungsform (IF3, D),
- entwickeln zur Grammatik einer regulären Sprache ***oder kontextfreien Sprache*** einen zugehörigen endlichen Automaten ***oder Kellerautomaten***(IF3, M),
- analysieren und erläutern Grammatiken regulärer ***oder kontextfreier*** Sprachen (IF3, A),
- modifizieren Grammatiken regulärer ***oder kontextfreier*** Sprachen (IF3, M),
- ermitteln die formale Sprache, die durch eine Grammatik erzeugt wird (IF3, A),

[1] Teile dieses Abschnitts wurden den Planungsunterlagen der Veranstaltung »Einführung in die Informatik – Teil 1« aus dem Wintersemester 2012/2013 von Prof. Dr. Till Tantau zu den Themen **Formale Grammatiken**, **Endliche Automaten** und **Reguläre Ausdrücke** entnommen (dort: Kapitel 5) und an einigen Stellen geändert.

[2] Die in den Kompetenzen hervorgehobenen Elemente sind für den Leitungskurs vorgesehen.

- entwickeln zu einer regulären ***oder kontextfreien Sprache*** eine Grammatik, die die Sprache erzeugt (IF3, M),
- entwickeln zur akzeptierten Sprache eines Automaten eine zugehörige Grammatik (IF3, M),
- beschreiben an Beispielen den Zusammenhang zwischen Automaten und Grammatiken (IF3, D),
- ***modellieren und implementieren Scanner, Parser und Interpreter zu einer gegebenen regulären Sprache (IF3, I),***
- zeigen die Grenzen endlicher Automaten und regulärer Grammatiken im Anwendungszusammenhang auf (IF3, A).

Vielleicht erinnern Sie sich noch an eine der Fragen in einer der ersten Unterrichtsstunden in Informatik.

Die Frage war …

Wie weit waren Sie je in Ihrem Leben physikalisch von einer mit Text bedruckten Fläche entfernt?

Wenn Sie jemals zehn Meter geschafft haben, dann sind Sie vermutlich eine Reinkarnation eines Eremiten.

Texte sind allgegenwärtig und ihre effiziente Verarbeitung stellt eine der wichtigsten Beschäftigungen von Informatiksystemen dar. Aus diesem Grund ist auch viel Theorieforschung in die Analyse von Zeichenketten geflossen. Diese Forschung kann man als überaus erfolgreich bezeichnen: Es gibt kaum einen Begriff in der Theoretischen Informatik, der so gut verstanden ist wie der der *regulären Sprache*.

Wichtiger als der Umstand, dass man reguläre Sprachen theoretisch gut versteht, ist, dass man auch sehr effiziente Verfahren zur Verarbeitung solcher Sprachen kennt. Dazu werden wir den Begriff des *endlichen Automaten* kennen lernen, welche man sich als sehr einfaches Informatiksystem vorstellen kann.

Reguläre Sprachen lassen sich auch sehr elegant mittels gleich zweier Formalismen beschreiben: *reguläre Grammatiken* und *reguläre Ausdrücke*. Letztere werden für uns wichtiger sein, da sie in vielen Anwendungen vorkommen: Beim dem Ihnen hoffentlich noch bekannten Programm `grep` steht »re« für »regular expressions«, bei SQL kann man mittels `regex` einen Vergleich mit regulären Ausdrücken benutzen, und so weiter.

1.2 Formale Grammatiken

Detailkompetenzen

- Probleme als formale Sprachen formulieren können
- Syntax von kontextfreien und regulären Grammatiken kennen
- Ableitungen bilden können und die erzeugte Sprache angeben können
- Probleme mittels formalen Grammatiken beschreiben können

Aus Sicht von Hardcore-Theoretikern gibt es in der Informatik nur drei wichtige Arten, Dinge zu repräsentieren: als Zeichenketten, als Zeichenketten oder als Zeichenketten. Hierbei gilt: Was nicht passt, wird passend gemacht – wie wir in diesem und den nächsten Abschnitten sehen werden, kann man, wenn man denn unbedingt möchte (und Theoretiker möchten unbedingt), alles als Zeichenketten auffassen: Von Zahlen über Graphen bis zu komplexen Probleminstanzbeschreibungen.

Texte (also Zeichenketten) sind nur selten völlig zufällig. Vielmehr haben sie fast immer eine recht komplexe innere Struktur. Nehmen wir einen deutschsprachigen Zeitungsartikel als Beispiel. Es gibt endlos viele unterschiedliche Arten, einen solchen Artikel zu schreiben, trotzdem gehorcht jeder den (ausgesprochen komplexen) Regeln den deutschen Sprache. Ganz unabhängig vom Inhalt muss der Text aus Wörtern bestehen, die nur in ganz bestimmten Reihenfolgen stehen dürfen. Diese Regeln bezeichnet man auch als *syntaktische Regeln*, die Bedeutung der Wörter hingegen als *Semantik*. Ganz ähnlich verhalten sich die Dinge bei Programmen, die ja letztendlich auch nur Texte sind. Wieder kann man endlos unterschiedliche Programme »aufschreiben«, jedoch müssen syntaktische Regeln penibel eingehalten werden, will man einen »Syntax-Error« vermeiden.


Noam Chomsky

Copyright by Duncan Rawlinson, Creative Commons Attribution License

Die in diesem Abschnitt eingeführten (formalen) Grammatiken stammen in dieser Form von Noam Chomsky und sind eine Methode, die erlaubte Struktur eines Textes

zu beschreiben. Wichtig dabei ist, dass es *nicht* um den Inhalt und die Bedeutung geht, Grammatiken scheren sich nicht sonderlich um diese. Sie beschreiben lediglich den Aufbau von Texten.

1.2.1 Motivation

Grammatik von natürlichen Sprachen

Die Grammatik von natürlichen Sprachen.

- Bei natürlichen Sprachen beschreibt die Grammatik (unter anderem), wie Sätze gebildet werden können.
- Es gibt verschiedene Ansätze, aber ein gängiger ist, den Satzbau durch *Regeln* zu beschreiben.

Beispiel: Typische Grammatikregeln

1. Das Subjekt in einem Satz kann von der Bauart sein »Artikel Substantiv«.
2. Das Subjekt kann auch von der Bauart sein »Artikel Adjektivliste Substantiv«.
3. Die Adjektivliste besteht aus durch Kommata oder durch »und« getrennte Adjektive.
4. Adjektiven können ihrerseits Adjektivlisten vorangestellt werden.
5. ...

Ein Beispiel, wie man Grammatikregeln anwendet.

Anwenden von Regeln (Ableiten)
Subjekt
⇒ Artikel Adjektivliste Substantiv
⇒ Artikel Adjektivliste Informatik
⇒ Die Adjektivliste Informatik
⇒ Die Adjektiv Informatik
⇒ Die zuckersüße Informatik

Ein Beispiel, wie man Grammatikregeln rückwärts anwendet.

Herausfinden der Regelanwendungen (parsen)

Der gebildete Student
⇐ Der Adjektiv Student
⇐ Der Adjektiv Substantiv
⇐ Der Adjektivliste Substantiv
⇐ Artikel Adjektivliste Substantiv
⇐ Subjekt

Die blaue die Mensch
⇐ Die Adjektiv die Mensch
⇐ Artikel Adjektiv die Mensch
⇐ Artikel Adjektivliste die Mensch
⇐ ???

Ein paar Beobachtungen.

- Grammatikregeln sind *Ersetzungsregeln*.
- In einem noch nicht fertigen Satz wie »Der Adjektiv Student« gibt es Teile, die noch ersetzt werden, und Teile, die nicht mehr ersetzt werden.
- Die Teile, die nicht weiter ersetzt werden, nennt man *Terminale*.
- Die Teile, die noch weiter ersetzt werden, nennt man *Nonterminale*.

Beachte: Ersetzungsregeln reichen nicht aus, um die gesamte Komplexität natürlicher Sprache zu beschreiben.

Grammatik von Programmiersprachen

Die Grammatik von Programmiersprachen

- Bei Programmiersprachen beschreibt die Grammatik, wie Programmtexte gebildet werden können.
- Es gibt verschiedene Ansätze, aber ein gängiger ist, Programmtexte durch *Regeln* zu beschreiben.

Beispiel: Typische Grammatikregeln

1. Eine Methodenkopf kann von der Bauart sein »`def` Bezeichner(Parameterliste):«.
2. Ein Bezeichner ist eine Folge von Ziffern und Buchstaben, die mit einem Buchstaben (oder Unterstrich) beginnt.
3. Eine Parameterliste besteht aus vielen Parameterbezeichnern.
4. Eine Methode besteht aus einem Kopf und einem Rumpf.
5. ...

Ein Beispiel, wie man Grammatikregeln anwendet.

Anwenden von Regeln (Ableiten)

Methodenkopf
⇒ `def` Bezeichner `(` Parameterliste `):`
⇒ `def` Bezeichner `(self,` Parameterliste `):`
⇒ `def setzeWert (self,` Parameterliste `):`
⇒ `def setzeWert (self,` Parameter `):`
⇒ `def setzeWert (self, neuerWert ):`

Herausfinden der Regelanwendungen (Parsen)

`def setzeMolekuel (self, seltsam):`
⇐ `def setzeMolekuel (self,` Parameter `):`
⇐ `def setzeMolekuel (self,` Parameterliste `):`
⇐ `def setzeMolekuel (` Parameterliste `):`
⇐ `def` Bezeichner `(` Parameterliste `):`
⇐ Methodenkopf

Ein paar Beobachtungen

- Wieder sind Grammatikregeln *Ersetzungsregeln*.
- Wieder gibt es Teile, die noch ersetzt werden, und Teile, die nicht mehr ersetzt werden.
- Die Teile, die nicht weiter ersetzt werden, nennt man wieder *Terminale*.
- Die Teile, die noch weiter ersetzt werden, nennt man wieder *Nonterminale*.

Beachte: Wieder reichen (einfache) Ersetzungsregeln nicht aus, um die gesamte Komplexität von Programmiersprachen zu beschreiben.

Beschreiben und Parsen

Wieso sind Grammatiken wichtig?

1. Mit Hilfe von Grammatikregeln können wir eindeutig *beschreiben*, welche Programmtexte »korrekt« sind.

   Dies ist wichtig, damit klar ist, worüber man überhaupt redet.

2. Wir können auch die *Struktur* von Programmen beschreiben.

   Dies ist wichtig, damit man Programme übersetzen kann.

3. Wir können anhand der Grammatik die Struktur von Programmen herausfinden. Diesen Vorgang nennt man *parsen*.

   Dies ist eine der ersten und wichtigsten Aufgaben von Übersetzern.

Ein Versprechen

Die Theorie der Zeichenketten erlaubt es, Grammatiken *automatisch* in effiziente Parser zu verwandeln.

1.2.2 Formale Sprachen

Was ist eine Sprache?

Was ist eine formale Sprache?

- Natürlichen Sprachen sind komplexe Dinge, bestehend aus Wörtern, ihrer Aussprache, einer Grammatik, Ausnahmen, Dialekten, und vielen mehr.
- Bei *formalen Sprachen* vereinfacht man radikal.

Definition 1 (Alphabet)**.** Ein *Alphabet* ist eine nicht-leere, endliche Menge von *Symbolen* (auch *Buchstaben* genannt).

Definition 2 (Wort)**.** Ein *Wort* ist eine (endliche) Folge von Symbolen.

Definition 3 (Formale Sprache)**.** Eine *formale Sprache* ist eine (oft unendliche!) Menge von Wörtern für ein festes Alphabet.

Bemerkungen und Notationen

Alphabete

- Alphabete werden häufig mit griechischen Großbuchstaben bezeichnet, also Γ oder Σ. Manchmal auch mit lateinischen Großbuchstaben, also N oder T.
- Ein Symbol oder »Buchstabe« kann auch ein komplexes »Ding« sein, wie der »Buchstabe« »Adjektivliste«.

Beispiele

- Die Groß- und Kleinbuchstaben
- Die Menge $\{0, 1\}$ (bei Informatikern beliebt)
- Die Menge $\{A, C, G, T\}$ (bei Biologen beliebt)
- Die Zeichenmenge des UNICODE.

Bemerkungen und Notationen

Wörter

- »Wörter« sind im Prinzip dasselbe wie Strings. Insbesondere können in Wörtern Leerzeichen als Symbole auftauchen.
- Die Menge aller Wörter über einem Alphabet Σ hat einen besonderen Namen: Σ^*.
- Deshalb schreibt man oft: »Sei $w \in \Sigma^*, \ldots$ «
- Es gibt auch ein *leeres Wort*, abgekürzt ϵ oder λ, das dem String `""` entspricht.

Beispiele

- `Hallo`
- `TATAAAATATTA`
- ϵ
- `Hallo Welt.`
- `def` Bezeichner `(` Parameterliste `)`
  (Neun Buchstaben)

Bemerkungen und Notationen

Sprachen

- Statt »formale Sprache« sagt man einfach »Sprache«.
- Als Menge von Wörtern ist eine Sprache eine Teilmenge von Σ^*.
- Deshalb schreibt man oft: »Sei $L \subseteq \Sigma^*, \ldots$ «
- Formale Sprachen müssen weder sinnvoll noch interessant sein.

Beispiele

- Die Menge $\{AAA, AAC, AAT\}$ (endliche Sprache).
- Die Menge aller Python-Programmtexte (unendliche Sprache).
- Die Menge aller Basensequenzen, die `TATA` enthalten (unendliche Sprache).

1.2.3 Formale Grammatiken

Was ist eine Grammatik?

Was ist eine formale Grammatik?

- Grammatiken für natürliche Sprachen sind komplex und mit vielen Ausnahmen behaftet.
- Bei *formalen Grammatik* vereinfacht man wieder radikal.

Definition 4 (Formale Grammatik)**.** Eine *formale Grammatik* besteht aus vier Dingen:

1. Einer Menge N von *Nonterminalen* (Nonterminalalphabet).
2. Einer Menge T von *Terminalen* (Terminalalphabet).
3. Einem *Startsymbol* $S \in N$.
4. Einer Menge von *Ersetzungsregeln* der Form »linke Seite $\to$ rechte Seite«.

Beispiel einer Grammatik.

1. Die Nonterminale sind $N = \{S, X, Y\}$.
2. Die Terminale sind $T = \{a, c, g, t\}$.
3. Das Startsymbol ist S.
4. Die Regeln lauten

$$\begin{aligned} S &\to aS \\ S &\to cS \\ S &\to gS \\ S &\to tS \\ S &\to X \\ X &\to atgY \end{aligned} \qquad \begin{aligned} Y &\to aY \\ Y &\to cY \\ Y &\to gY \\ Y &\to tY \\ Y &\to \epsilon \end{aligned}$$

In verkürzender Schreibweise kann man die Regel auch so notieren, dass Regeln mit gleicher linker Seite zusammengefasst und die rechten Seiten durch einen senkrechten Strich | für alternative Auswahl getrennt werden.

$$\begin{aligned} S &\to aS \mid cS \mid gS \mid tS \mid X \\ X &\to atgY \\ Y &\to aY \mid cY \mid gY \mid tY \mid \epsilon \end{aligned}$$

Ableitungen

Hat man eine Grammatik, so kann man *Ableitungen* bilden. Dies funktioniert wie folgt:

- Man startet mit dem Startsymbol S.
- Dann ersetzt man das Startsymbol gemäß einer Regel $R_1 = S \to w_1$ durch die rechte Seite der Regel: $S \vdash_{R_1} w_1$
- Das resultierende Wort $w_1 = a_1 \dots a_k N_1 \dots N_l b_1 \dots b_h N_1' \dots$ (genannt *Satzform*) besteht aus Terminalen a_i, b_j und Nonterminalen $N_1, N_2, \dots, N_1'$
- Nun sucht man sich eine Regel $R_2 = u \to u'$ aus, so dass die linke Regelseite u irgendwo in der Satzform als Teilstring vorkommt: $w_1 = \alpha_1 u \beta_1$
- Dann ersetzt man genau dieses Vorkommen der linke Regelseite in der Satzform durch die rechte Seite u': $w_1 = \alpha_1 u \beta_1 \vdash_{R_2} \alpha_1 u' \beta_1 = w_2$.

- Dies wiederholt man, bis nur noch Terminale vorhanden sind: $S \vdash w_1 \vdash w_2 \vdash \ldots \vdash w_t$ *mit* $w_t \in T^*$.
- Falls eine Wort w_t über dem Terminalalphabet durch endlich viele Ableitungsschritte auf diese Weise aus dem Startsymbol erzeugt werden kann, schreiben wir $S \vdash^* w_t$ (transitiver Abschluss der Ableitungsrelation $\vdash$).

Beispiel von Ableitungen, erzeugte Sprachen

Welche Ableitungen sind in der oben betrachteten Grammatik G möglich? Eine Möglichkeit ist $S \vdash aS \vdash aX \vdash aatgY \vdash aatggY \vdash aatgg$.

Definition 5. Für eine Grammatik G nennen wir die Menge aller Wörter, die sich damit ableiten lassen, die *erzeugte Sprache*. Sie wird notiert als $L(G) := \{w \in T^* \mid S \vdash^* w\}$.

Welche Sprache erzeugt die obige Grammatik? (Was ist also die Menge aller Wörter, die sich erzeugen lassen?)

Reguläre Grammatiken

Reguläre Grammatiken sind die einfachsten Grammatiken.

Definition 6 (Reguläre Grammatik)**.** Bei einer *regulären Grammatik* wird immer ein Nonterminal durch eine Folge von Terminalen gefolgt von einem oder keinem Nonterminal ersetzt.

Bemerkungen

- In jeder Ableitung mittels einer regulären Grammatik haben alle Satzformen genau ein Nonterminal am Ende, bis auf das abgeleitete Wort.
- Mit regulären Grammatiken kann man Sprachen beschreiben wie »Enthält acc, aber nicht atg«.
- Mit regulären Grammatiken kann man Sprachen beschreiben wie die erlaubten Python-Bezeichner.
- Komplexe Sprachen wie Python lassen sich nicht mit regulären Grammatiken beschreiben.

Beispiele reguläre Grammatiken.

Beispiel
Folgende Grammatik erzeugt alle Wörter, deren Länge ein Vielfaches von 3 ist.

$$\begin{aligned} S &\to aX \mid cX \mid gX \mid tX \\ X &\to aY \mid cY \mid gY \mid tY \\ Y &\to aZ \mid cZ \mid gZ \mid tZ \\ Z &\to S \mid \epsilon \end{aligned}$$

Beispiel
Folgende Grammatik erzeugt die Sprache $\{aaa, atg, act\}$.

$$S \to aaa \mid atg \mid act$$

Konstruieren Sie eine Grammatik, die alle Worte über dem Alphabet $\{a, c, g, t\}$ erzeugt, die *nicht* das Teilwort at enthalten.

Kontextfreie Grammatiken

Kontextfreie Grammatiken können mehr.

Definition 7 (Kontextfreie Grammatik)**.** Bei einer *kontextfreien Grammatik* wird immer ein Nonterminal durch eine beliebige Folge von Terminalen und Nonterminalen ersetzt.

Bemerkungen

- Ableitungen für kontextfreie Grammatiken beschreibt man durch *Ableitungsbäume*, in denen innere Knoten den Nichtterminalen der verschiedenen Satzformen entsprechen und die Blätter durch Terminalsymbole beschrieben sind. Das erzeugte Wort ist der String, der durch die Blätter von links nach rechts gelesen gebildet wird.
- Kontextfreie Grammatiken können verschachtelte Strukturen beschreiben wie Klammerausdrücke oder HTML.
- Komplexe Sprachen wie Python lassen sich weitgehend, aber nicht vollständig mit kontextfreien Grammatiken beschreiben.

Beispiele kontextfreier Grammatiken.

1. Grammatik für die Sprache $\{\overbrace{a \cdots a}^{n}\overbrace{b \cdots b}^{n} \mid n \geq 1\}$.

   $$S \to aSb \mid ab$$

2. *Palindrome* sind Wörter, die von rechts nach links gelesen die gleiche Buchstabenreihenfolge ergeben: $w = a_1 a_2 \ldots a_k\ \zeta\ a_k \ldots a_1$, wobei ζ ein weiteres Alphabetsymbol sein kann oder das leere Zeichen.

   Beispiele: *otto*, *drehmagiezettelumamulettezeigamherd*

   einegüldnegutetugendlügenie

   Grammatik für diese Sprache:

   $$S \to aSa \mid bSb \mid \ldots \mid zSz \mid a \mid \ldots \mid z \mid \epsilon$$

3. Strukturell noch einfacher zu sein scheinen sogenannte *Quadrate*, die aus 2 gleichen Teilstrings bestehen: $w = uu$.

   Hierfür kann man jedoch keine kontextfreie Grammatik finden, sondern benötigt komplexere Regeln, die mehrere Symbole auch auf der linken Seite verwenden.

Beispiele kontextfreier Grammatiken.

Beispiel

Folgende Grammatik erzeugt arithmetische Ausdrücke.

$$\begin{aligned} S &\to (S) \mid S + S \mid S - S \mid S \cdot S \mid S / S \mid N \\ N &\to 0M \mid 1M \mid 2M \mid \cdots \mid 9M \\ M &\to 0M \mid 1M \mid 2M \mid \cdots \mid 9M \mid \epsilon \end{aligned}$$

Anspruchsvolle Aufgabe

Konstruieren Sie eine Grammatik für die Sprache $\{\overbrace{a \cdots a}^{n}\overbrace{b \cdots b}^{n}\overbrace{c \cdots c}^{n} \mid n \geq 1\}$.

Zusammenfassung – Formale Sprachen

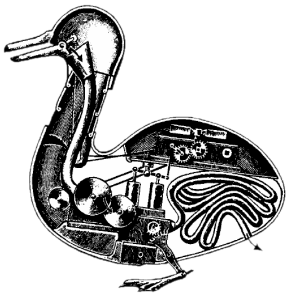
Formale Sprachen – Zusammenfassung

1. Formale Sprachen sind *beliebige Mengen von Wörtern*.
2. Formale Sprachen kann man als *Problemstellungen* ansehen.
3. Formale Grammatiken kann man benutzen, um formale Sprachen zu *erzeugen* und zu *beschreiben*.
4. Wichtige Grammatikarten sind *reguläre* und *kontextfreie* Grammatiken.

1.3 Endliche Automaten

Detailkompetenzen

- Funktionsweise von endlichen Automaten verstehen
- Formale Sprachen mittels endlicher Automaten entscheiden können
- Endliche Automaten implementieren können
- Einschätzen können, ob eine Sprache regulär ist



Public domain

Die Bezeichnung »endlicher Automat« ist vielleicht etwas irreführend. Die Theorie endlicher Automaten beschäftigt sich mit allem Möglichen, aber gerade nicht mit den Geräten, die man normalerweise als Automaten bezeichnet. Es wird in diesem Kapitel *nicht* darum gehen, Cola-Automaten zu programmieren oder auch nur zu modellieren. Ebensowenig geht es um die skurrilen Geräte, die in vergangenen Jahrhunderten als Automaten (wörtlich »Selbst-Bewegtes«) gefeiert wurden wie die im Jahr 1738 von Vaucanson vorgestellte mechanische Ente. Sie konnte sich laut Wikipedia »watschelnd fortbewegen, aber auch fressen, verdauen und ausscheiden«.

Unknown author, Creative Commons Attribution Sharealike License

Ebenso schön sind die Automaten von Jaquet-Droz, von denen Wikipedia zu berichten weiß: »Der Schreiber ist 70cm hoch, hat eine Gänsefeder in der Hand, sitzt vor einem kleinen Tisch und hat bewegliche Augen und Kopf. Er kann jeden beliebigen Text mit bis zu 40 Buchstaben Länge schreiben. Der Text wird auf einem Rad kodiert, wo die Buchstaben dann einer nach dem anderen abgearbeitet werden. Wenn er gestartet wird, taucht er zunächst die Feder in die Tinte und schüttelt sie leicht ab, dann schreibt er, wobei er wie ein echter Schreiber die Auf- und Abwärtsstriche richtig beachtet und auch absetzt. Er kann mehrzeilig schreiben und beachtet Leerzeichen.«

Die endlichen Automaten der Theoretischen Informatik sind ein Formalismus, der die Konzepte *Zustand* und *Zustandsänderungen* allgemein modelliert. Da wir *endliche* Automaten untersuchen werden, wird die Menge dieser möglichen Zustände stets endlich sein. Zustände versteht man vielleicht am besten anhand einer Analogie zur Biologie: Bei einer einfachen Zelle könnte der Zustand beispielsweise beschreiben, welche Proteine sich gerade in der Zelle befinden und welche Gene aktiviert sind. Durch Umwelteinflüsse und durch innere Prozesse wird sich der Zustand der Zelle langsam oder abrupt ändern, dies werden wir *Zustandsänderungen* nennen. Ziel der Automatentheorie ist nun, herauszufinden, welche Folge von äußeren Einflüssen welche Effekte in einem Automaten / einer Zelle haben wird. Die Analogie zu Zellen ist nicht herbeikonstruiert, vielmehr beschäftigt sich eine ganze Teildisziplin der Automatentheorie, nämlich die *Theorie der zellulären Automaten*, damit, wie sich ganze Felder solcher Automaten verhalten.

1.3.1 Motivation

Erste Motivation von endlichen Automaten

- *Endliche Automaten* sind ein *besonders einfaches* Modell von Informatiksystemen.
- Modelle sind nützlich, wenn man verstehen möchte, was Informatiksysteme *prinzipiell können* und *prinzipiell nicht können*.

Idee

Was sind die aller grundlegendsten Eigenschaften eines Informatiksystems?

- Es gehen Eingaben hinein.

- In Abhängigkeit der Eingaben passiert etwas im Inneren des Informatiksystems.
- Am Ende wird ein Ergebnis produziert.

Zweite Motivation von endlichen Automaten

- *Endliche Automaten* sind ein *Gegenstück* zu regulären Grammatiken.
- Wohingegen reguläre Grammatiken Probleme nur *beschreiben*, können endliche Automaten Probleme auch *lösen*.

Dritte Motivation von endlichen Automaten

- *Endliche Automaten* sind ein *besonders schönes* Modell von Informatiksystemen.
- Es hat wunderschöne theoretische Eigenschaften.

1.3.2 Was sind endliche Automaten?

Idee

Die Idee hinter endlichen Automaten.

1. Der Automat liest eine *Eingabe* Zeichen für Zeichen.
2. Je nach gelesenem Zeichen wechselt der Automat seinen internen *Zustand*.
3. Am Ende wird eine *Ausgabe* produziert, die nur vom letzten Zustand abhängt.

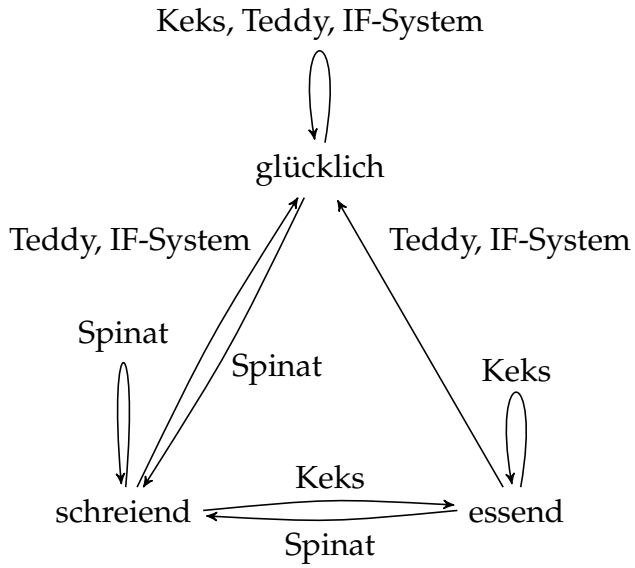
Ein Kleinkind als endlicher Automat

- Der Automat verfügt über drei Zustände: glücklich, schreiend, essend.
- Am Anfang ist der Automat im Zustand glücklich.
- Mögliche Eingaben sind: Keks, Spinat, IF-System, Teddybär.

Die Effekte der Eingabe sind folgende:

Alt-Zustand	Keks	Spinat	Teddybär/IF-System
glücklich	glücklich	schreiend	glücklich
schreiend	essend	schreiend	glücklich
essend	essend	schreiend	glücklich

Der Kleinkind-Automat anders aufgemalt.



Geben Sie einen Automaten an, der herausfindet, ob eine 0-1-Folge mindestens zwei 1en enthält.

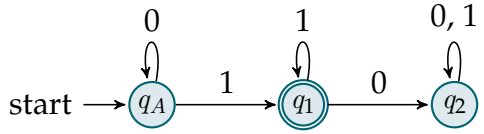
Definition

Die mathematische Definition eines Automaten.

Definition 8 (Endlicher Automat)**.** Ein *endlicher Automat* hat die folgenden Bestandteile:

1. Ein *Eingabealphabet* Σ aus dem die Eingabesymbole kommen.
2. Eine (endliche) *Zustandsmenge* Q.
3. Einen *Startzustand* $q_A \in Q$.
4. Eine Menge von *akzeptierenden Zuständen* $F \subseteq Q$.
5. Einen *Zustandsgraphen*, dessen Knoten Zustände sind und dessen Kanten mit Eingaben beschriftet sind.

Beispiel



Akzeptierte Sprache.

Definition 9 (Akzeptierte Sprache)**.** Bei *Eingabe eines Wortes* $w \in \Sigma^*$ arbeitet ein endlicher Automat wie folgt:

1. Er beginnt im Startzustand q_A.
2. Dann liest er das erste Zeichen von w und wechselt den Zustand entsprechend dem Zustandsgraphen.
3. Dann liest er das zweite Zeichen und wechselt den Zustand wieder entsprechend; und so weiter.
4. Ist das Wort komplett gelesen, so schauen wir, ob wir in einem *akzeptierenden Zustand* sind.
   - Wenn ja, so *akzeptiert* der Automat w.
   - Wenn nein, so *verwirft* der Automat w.

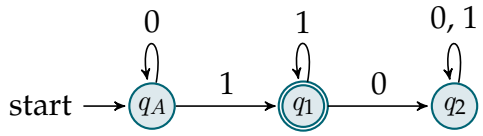
Die Menge aller von einem Automaten M akzeptierten Wörter bezeichnen wir als *die von M akzeptierten Sprache*, abgekürzt $L(M)$.

Beispiele

Beispiele von endlichen Automaten.

Beispiel

Ein Automat, der folgende Sprache akzeptiert: $\{w \mid w$ besteht aus 0en, gefolgt von 1en$\}$.



Nur der Zustand q_1 ist akzeptierend.

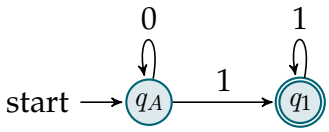
Ein paar Vereinbarungen.

- Akzeptierende Zustände werden durch eine doppelten Kreis angedeutet.
- Wir erlauben, dass in einem Zustand zu einer Eingabe kein Pfeil vorliegt.

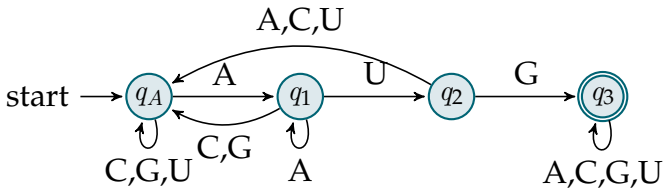
  Dann wird das Wort *sofort verworfen, egal ob der Zustand akzeptierend ist oder nicht*.
- Gibt es Zustände, bei denen zu einer Eingabe kein Pfeil vorliegt, nennen wir den Automaten *unvollständig*, sonst *vollständig*.

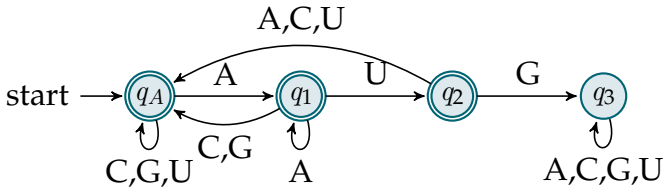
Beispiel

Ein Automat, der dieselbe Sprache wie eben akzeptiert: $\{w \mid w$ besteht aus 0en, gefolgt von 1en$\}$.



Geben Sie die Sprachen an (in Mengenschreibweise oder umgangssprachlich), die die folgenden Automaten akzeptieren:





Implementation

Man kann einen Automaten in ein Programm auf Zeichenketten umwandeln.

- Den *Zustand* speichert man als Wert in einem Attribut.
- Die *Eingabesymbole* sind einfach Zeichen.
- Die *Zustandsübergänge* kann man durch Verzweigungen implementieren.

Bemerkungen:

- Die Struktur von Automatenklassen ist sehr fest und kann *automatisch erzeugt* werden, wenn man den Automaten gegeben hat.
- Automaten sind sehr schnell: Ihre Rechenzeit ist *linear* bezogen auf die Länge der Eingabe.

Beispiel für eine Automatenklasse.

```
class Automaton1:
  def __init__(self):
    self.zustand= "qA"        # qA = "qA", q1 = "q1", q2 = "q2"

  def handleInput (self, zeichen):
    if self.zustand == "qA":
      if zeichen == '0':
        self.zustand= "qA"
      elif zeichen == '1':
        self.zustand= "q1"
      else:
        self.zustand= "q2"
    elif self.zustand == "q1":
      if zeichen == '0':
        self.zustand= "q2"
      elif zeichen == '1':
        self.zustand= "q1"
      else:
        self.zustand= "q2"

  def isAccepted (self, zeichenkette):
    self.zustand= "qA"
    for zeichen in zeichenkette:
      self.handleInput(zeichen)
    return self.zustand == "q1"
```

Übergangsfunktion als Tabelle

Bei dem folgenden Automaten ist s_0 der Anfangszustand, s_0 ist (auch) der einzige Endzustand.

Zustand/Folgezustand	s_0	s_1	s_2
s_0	0	1	–
s_1	1	–	0
s_2	–	0	1

Der Automat ist (MSWWF 1999, S. 93) entnommen.

- Geben Sie an, welche Sprache der Automat akzeptiert.

  Hinweis: Interpretieren Sie die Folgen aus 0 und 1 als Binärzahlen und überlegen Sie, welche Zahlen akzeptiert werden und welche nicht. Dazu können Sie z. B. die Zahlen systematisch von Ihrem Automaten überprüfen lassen.
- Erstellen Sie den zugehörigen Automatengraphen.
- Erstellen Sie eine Automatenklasse, die diesen Automaten implementiert.

Alternative Tabellendarstellung

Um die Übertragung in die Implementierung der Übergangsfunktion zu vereinfachen, kann die Tabelle so dargestellt werden, dass die erste Spalte weiterhin den Zustand enthält, von dem ausgegangen wird, die obere Zeile aber alle möglichen Eingaben. Dann bestehen die Einträge aus dem Folgezustand, der sich ergibt:

Zustand/Eingabe	0	1
s_0	s_0	s_1
s_1	s_2	s_0
s_2	s_1	s_2

Übergangsfunktion als Tabelle 2/2

Zu dem folgenden Automaten sind z_0 der Anfangszustand, z_2 und z_3 die beiden Endzustände.

Zustand/Folgezustand	z_0	z_1	z_2	z_3
z_0	1	0	–	–
z_1	–	–	1	0
z_2	1	0	–	–
z_3	–	–	1	0

Übungen zu endlichen Automanten finden Sie unter A.1.1.

1.3.3 Was können endliche Automaten?

Was sie können

Der Hauptsatz der Automatentheorie.

Zur Erinnerung:

- Mit Hilfe von regulären Grammatiken lassen sich Probleme *beschreiben*.
- Mit Hilfe von endlichen Automaten lassen sich Probleme *akzeptieren* oder *entscheiden*.

Was ist nun das Verhältnis von regulären Grammatiken und endlichen Automaten?

Satz 10. *Sei L eine Sprache. Dann sind folgende Aussagen äquivalent:*

1. *Es gibt eine reguläre Grammatik, die L erzeugt.*
2. *Es gibt einen endlichen Automaten, der L akzeptiert.*

Bemerkungen zum Hauptsatz.

- Falls wir ein Problem mit einer regulären Grammatik *beschreiben* können, dann können wir es auch *sehr effizient* entscheiden.
- Diese »Umwandlung« von regulären Grammatiken in Automaten kann man automatisieren.
- Der Hauptsatz ist nicht ganz leicht zu beweisen.
- Aufgrund des Hauptsatzes nennen wir eine Sprache *regulär*, wenn sie von einer regulären Grammatik erzeugt wird oder von einem endlichen Automaten akzeptiert wird.

Praktische Beispiele von regulären Sprachen.

Folgende Sprachen sind regulär:

- Die erlaubten Python-Bezeichner.
- Die erlaubten Python-Kommentare.
- Die erlaubten WWW-Adressen.
- Die erlaubten Dateinamen.

Was sie nicht können

Es gibt Sprachen, die nicht regulär sind.

Leider ist nicht jede Sprache regulär. Hier ein Beispiel:

Beispiel
Die Sprache $\{a^n b^n \mid n \geq 1\}$ ist nicht regulär.

Nennen Sie Gründe, weshalb die Sprache nicht durch einen Automaten akzeptiert werden kann. Was ist das Problem?

Typische Sprachen, die nicht regulär sind.

Folgende Sprachen sind in der Regel nicht regulär:

- Sprachen, bei denen man etwas mitzählen muss und die Zahlen beliebig groß werden können.
- Sprachen, bei denen man längere Wortteile vorne mit Wortteilen hinten vergleichen muss.

Insbesondere sind nicht regulär:

- Die korrekten arithmetischen Ausdrücke
- Die erlaubten HTML-Texte
- Die erlaubten Python-Programme
- Die erlaubten Texte in anderen Programmiersprachen

Zusammenfassung – Endliche Automaten

Endliche Automaten – Zusammenfassung

1. Endliche Automaten sind ein *sehr einfaches* und *sehr schnelles* Modell von Informatiksystemen.
2. Endliche Automaten *akzeptieren* Wörter und (dadurch) Sprachen.
3. Endliche Automaten können genau dasselbe wie *reguläre Grammatiken*.
4. Endliche Automaten können *längst nicht alles*.

1.4 Reguläre Ausdrücke

Pattern Matching

Detailkompetenzen

- Konzept des regulären Ausdrucks verstehen.
- Probleme als reguläre Ausdrücke beschreiben.
- Mächtigkeit regulärer Ausdrücke kennen.

Die Suche in und die Überprüfung von Texten ist eine der Hauptbeschäftigungen von Informatiksystemen. Ständig muss etwas gefunden werden wie Dateinamen, Gensequenzen oder Zahlen in Tabellen; ständig muss etwas überprüft werden wie die korrekte Form von Postleitzahlen, E-Mail-Adressen, Internet-Adressen oder Pizzabestellscheinformularadressangabefeldern.

Wie wir in späteren Vorhaben sehen werden, ist es gar nicht so leicht, Programme zu schreiben, die solches Suchen und Überprüfungen effizient durchführen. Um so besser, dass die Informatik seit langer Zeit ein Konzept zu bieten hat, das diese Aufgaben ungemein erleichter: Mit *regulären Ausdrücken* kann man herausfinden, ob Zeichenketten mit bestimmten – teils recht komplizierten – Eigenschaften in einem Text vorkommen.

Leider sind reguläre Ausdrücke oft, sagen wir, kryptisch. Selbst ein Profi wird den regulären Ausdruck `"[.?!][]\"')]*\\($\\| $\\|\t\\| \\)[ \t\n]*"` nicht sofort verstehen. Dies ist laut Emacs-Handbuch »a simplified version of the regexp that Emacs uses, by default, to recognize the end of a sentence together with any whitespace that follows.«

Auch wenn reguläre Ausdrücke fürchterlich komplex und unlesbar werden können, so ist die Grundidee doch recht einfach: Jeder Ausdruck beschreibt eine Menge von Wörtern, auf die er *passt*. Die einfachsten Ausdrücke passen einfach nur auf ein einzelnes Wort und sonst nichts. Man kann dann aber Ausdrücke zu komplexeren Ausdrücke zusammenbauen und diese beschreiben dann auch größere und komplexere Mengen von Wörtern. Die Hauptschwierigkeit bei regulären Ausdrücken ist auch nicht, dieses Konzept zu verstehen. Problematisch ist, dass sich keine einheitliche Syntax herausgebildet hat, wie man die Ausdrücke nun aufschreiben sollte. Hier kocht jeder sein eigenes Süppchen, welche wir nun alle auslöffeln müssen.

1.4.1 Einführung

Muster-Suche in Texten

Worum geht es bei der Mustersuche?

Das Problem, ein *Muster* innerhalb eines Textes zu finden, taucht in vielen Kontexten auf:

- Man sucht in einem elektronischen Dokument nach dem Vorkommen eines bestimmten Wortes; beispielsweise das Wort »Energiewende« in einem Koalitionsvertrag.
- Eine Suchmaschine möchte anzeigen, wo überall auf einer Webseite das Wort »flauschig« vorkommt.
- Man sucht in einer DNA-Sequenz nach allen Vorkommen einer bestimmten Basensequenz.

Das gesuchte Muster ist aber in der Regel *kein einzelnes Wort*:

- Statt »flauschig« würde man auch »Flauschig« finden wollen. Eventuell auch »FLAUSCHIG«.
- Sucht man die Basensequenz eines Gens, so würde man an Polymorphismus-Stellen auch zulassen, dass dort im Text vom Muster abgewichen wird.

Was ist ein Muster?

Muster und Muster-Suche

- Ein *Muster* (englisch *pattern*) *beschreibt* eine *Menge von möglichen Worten*.
- Bei der *Muster-Suche* (englisch *pattern search*) sind ein Muster und ein (längerer) Text gegeben.

  Ziel ist es, *Stellen im Text zu finden*, an denen ein Wort steht, das vom Muster beschrieben wird.

Beispiel

Wir werden bald definieren, dass das Muster `ab[cd]` die beiden Strings `abc` und `abd` beschreibt.
Würde man nach `ab[cd]` in `Ich werd' das abchecken.` suchen, so gäbe es genau einen Treffer, nämlich am Anfang von `abchecken`.

Was im Folgenden passiert.

- Es werden *reguläre Ausdrücke* eingeführt; dies ist eine Art, Muster aufzuschreiben.
- Es wird erklärt, welche Mengen an Wörtern ein regulärer Ausdruck beschreibt.
- Es wird *nicht* erklärt, wie die Algorithmen funktionieren, die nach Ausdrücken in Texten suchen.

  Es sei lediglich erwähnt, dass es *sehr schnelle* Algorithmen gibt und dass man *mehrere Kapitel über Theoretische Informatik* braucht, um diese wirklich zu verstehen.

Theoretischer Hintergrund

Wörter im Sinne der Theorie

Definition 11. 1. Ein *Alphabet* ist eine nichtleere, endliche Menge.

2. Die Elemente eines Alphabets nennt man *Buchstaben* oder auch *Symbole*.
3. Eine *endliche Folge* von Buchstaben nennt man ein *Wort*.

Beachte:

- Alphabete werden häufig mit griechischen Großbuchstaben bezeichnet, also Γ oder Σ.

  Praktische Beispiele sind die Menge $\{0, 1\}$ (bei Informatikern beliebt), die Menge $\{A, C, G, T\}$ (bei Biologen beliebt) und die Zeichenmenge des UNICODE.
- In »Wörter« können Leerzeichen (!) als Symbole auftauchen.
- Es gibt auch ein *leeres Wort*, abgekürzt ϵ oder λ, das Länge Null hat.

Die vornehme Bezeichnung für Mengen von Wörtern: Sprachen.

Definition 12. Eine *Sprache* ist eine Menge von Wörtern über einem Alphabet.

- Sprachen müssen weder sinnvoll noch interessant sein,
- sie können endlich sein oder auch unendlich.
- Ein Beispiel einer endlichen Sprache ist $\{AAA, AAC, AAT\}$;
- ein Beispiel einer unendlichen Sprache die Menge aller Basensequenzen, die `TATA` genau zweimal enthalten.

Muster-Suche, noch einmal etwas formaler.

- Ein *regulärer Ausdruck* wird ein *einzelnes Wort* sein, wie `ab[cd]` oder `x|y|z*`.
- Wir werden für jeden regulären Ausdruck R eine bestimmte Sprache $L(R)$ angeben, von der wir sagen werden, dass R *sie beschreibt*. Beispielsweise wird $L(\texttt{ab[cd]}) = \{abc, abd\}$ gelten.
- Das *Muster-Such-Problem* ist dann, zu einem gegeben regulären Ausdruck R und einem längeren Wort w ein Teilwort u in w zu finden mit $u \in L(R)$.

Praktischer Hintergrund

Das Programm Grep.

- Das Programm Grep löst das Muster-Such-Problem,
- wobei GREP für »Global search for REgular Patterns« steht.
- Parameter sind
  1. ein regulärer Ausdruck und
  2. der Name einer Datei, in der nach Vorkommen des Musters gesucht werden soll.
- Findet Grep das Muster in einer Datei, so gibt es die Zeile aus, in der das Muster vorkommt; mit der Option `-n` kann man sich die Zeilennummer auch anzeigen lassen.
- Aus historischen Gründen sollte man auch immer die Option `-E` angeben, damit die »moderne« Syntax für reguläre Ausdrücke benutzt wird.

Beispiel

```
1 humbert@abercorn:~$ grep -n -E "Philosophie" faust.txt
2 450:Habe nun, ach!  Philosophie,
3 humbert@abercorn:~$
```

1.4.2 Reguläre Ausdrücke

Die einfachsten Ausdrücke

Die einfachsten regulären Ausdrücke sind einfache Wörter.

Definition 13 (Reguläre Ausdrücke 1 – Atomare Ausdrücke)**.** Sei Σ ein Alphabet, in dem bestimmte Sonderzeichen nicht vorkommen. Dann ist jedes Wort w über diesem Alphabet ein regulärer Ausdruck, im Folgenden mit dem Buchstaben R abgekürzt.

Die von R beschriebene Sprache $L(R)$ ist einfach $\{w\}$, sie enthält also einfach nur w.

- Für diese einfachen Ausdrücke ist der formale Apparat mit Wörtern und Sprachen und $L(R)$ natürlich Overkill.
- Trotzdem ist schon hier das *Muster-Such-Problem* interessant: Es ist einfach das Problem, in einem Text ein ganz bestimmtes Wort zu finden.

Beispiele

```
humbert@abercorn:~$ grep -n -E "heilig" faust.txt
1522:... Knurre nicht, Pudel!  Zu den heiligen
1544:Das heilige Original
1638:Ich versenge dich mit heiliger Lohe!
3811:Und hier mit heilig reinem Weben
3899:Und warf den heiligen Becher
3965:Ob das Ding heilig ist oder profan;
4156:Beim heiligen Antonius
5203:Ihr seligmachend ist, sich heilig quäle,
6914:Was will der an dem heiligen Ort?
6928:Ihr Engel!  Ihr heiligen Scharen,
humbert@abercorn:~$
```

Die Alternative

Dies oder jenes...

Definition 14 (Reguläre Ausdrücke 2 – Alternativen)**.** Sind R_1 und R_2 beides reguläre Ausdrücke, so auch $R_1 \mid R_2$.

Die von $R_1 \mid R_2$ beschriebene Sprache $L(R_1 \mid R_2)$ ist die Vereinigung der Sprachen $L(R_1)$ und $L(R_2)$:

$$L(R_1 \mid R_2) = L(R_1) \cup L(R_2)$$

- Ein senkrechter Strich kann also genutzt werden, um *alternativ* nach den einen Wörtern oder nach den anderen Wörtern zu suchen.
- Der senkrechte Strich gehört zu den *Sonderzeichen*. Will man tatsächlich nach einem senkrechten Strich suchen, so muss man ihm einen Backslash voranstellen.

Beispiele

```
humbert@abercorn:~$ grep -n -E "Philosophie|Theologie|Medizin"
   faust.txt
450:Habe nun, ach!  Philosophie,
451:Juristerei und Medizin,
452:Und leider auch Theologie
2522:Fast möcht ich nun Theologie studieren.
2550:Wollt Ihr mir von der Medizin
2560:(Laut.) Der Geist der Medizin ist leicht zu fassen;
humbert@abercorn:~$
```

Die Verkettung

Erst dies, dann das...

Definition 15 (Reguläre Ausdrücke 3 – Konkatenation)**.** Sind R_1 und R_2 beides reguläre Ausdrücke, so auch R_1R_2.

Die von R_1R_2 beschriebene Sprache $L(R_1R_2)$ ist die so genannte *Verkettung* der Sprachen $L(R_1)$ und $L(R_2)$:

$$L(R_1R_2) = \{uv \mid u \in L(R_1), v \in L(R_2)\}.$$

- Die Sprache $L(R_1R_2)$ enthält alle Wörter w, die man irgendwie in zwei Teile $w = w_1w_2$ trennen kann, so dass der *Präfix* in $L(R_1)$ und der *Suffix* in $L(R_2)$ liegt.
- Man kann *runde Klammern* benutzen, wenn Verkettungen und Alternativen gemischt werden, um die Reihenfolge der Auswertung vorzugeben.
- Wieder kann man auch nach Klammern selbst suchen, wenn man einen Backslash voranstellt.

Beispiele

```
humbert@abercorn:~$ grep -n -E "(Phi|Theo)lo(soph|g)ie" faust.txt
450:Habe nun, ach!  Philosophie,
452:Und leider auch Theologie
2522:Fast möcht ich nun Theologie studieren.
humbert@abercorn:~$
```

Wie lautet $L(R)$ jeweils für folgende Ausdrücke R?

1. `(0|1|2)(3|4|5)`

2. `0|((1|2|3)3(4|5))(6|7)`
3. `(a||b)(c|d)`
4. `\||(\||\()(\)|\\|\|)`

   (Keine Angst, so etwas kommt zwar in Programmen, aber nicht in Klausuren vor.)

Der Kleene-Stern

Dies oder das oder dies oder das...

Definition 16 (Reguläre Ausdrücke 4 – Wiederholungen)**.** Ist R ein regulärer Ausdruck, so auch R^*.

Die von R^* beschriebene Sprache $L(R)$ ist der so genannte *Kleene-Stern* von $L(R)$:

$$L(R^*) = \{u_1 \dots u_n \mid u_i \in L(R)\}$$

- Die Sprache $L(R^*)$ enthält alle Wörter, die beliebige Aneinanderreihungen von *Wörtern in* $L(R)$ sind.
- In der Aneinanderreihungen kann ein Wort beliebig oft vorkommen, muss es aber nicht.
- Auch das leere Wort ϵ liegt in $L(R^*)$.
- Die Sprache $L(R^*)$ ist *unendlich*.

Beispiele

```
humbert@abercorn:~$ grep -n -E "as*e" faust.txt
...
5632:Solvet saeclum in favilla.
5751:Und die langen Felsennasen,
5752:Wie sie schnarchen, wie sie blasen!
5754:Durch die Steine, durch den Rasen
5772:Aus belebten derben Masern
5773:Strecken sie Polypenfasern
5788:Fasse wacker meinen Zipfel!
...
humbert@abercorn:~$
```

1. Welche Wörter der Länge ≤ 6 gehören zu $L(\texttt{(ab|cab)*})$?

2. Beschreiben Sie $L(R)$ für folgenden regulären Ausdruck R:
   `aug((a|u|g|c)(a|u|g|c)(a|u|g|c))*(uag|uaa|uga)`
3. Geben Sie drei Wörter an, die vom Ausdruck `0(1*0|2*0)*3` erzeugt werden.
4. Finden Sie einen regulären Ausdruck, der alle Worte über $\{a, b, c\}$ beschreibt, die mit a oder b beginnen und deren drittletzter Buchstabe ein c ist.

Reguläre und arithmetische Ausdrücke

Analogie von regulären und arithmetischen Ausdrücken.

Syntaktische Analogien

Arithmetische Ausdrücke

Bestandteile:

1. Zahlen
2. Unäre Operationen wie Wurzel.
3. Binäre Operationen wie
   - Addition
   - Multiplikation
   - Potenz
4. Klammern

Reguläre Ausdrücke

Bestandteile:

1. Ein-Wort-Sprachen
2. Unäre Operationen wie Kleene-Stern.
3. Binäre Operationen
   - Vereinigung
   - Verkettung
4. Klammern

Analogie von regulären und arithmetischen Ausdrücken.

Semantische Analogien

Arithmetische Ausdrücke

1. Ein arithmetischer *Ausdruck* und sein *Wert* sind verschiedene Dinge:

   Der *Ausdruck* $(5 + 6) \cdot 2$ hat den *Wert* 22.

2. Eine Zeichenfolge wie 22 kann man sowohl als Ausdruck als auch als Wert lesen.

Reguläre Ausdrücke

1. Ein regulärer *Ausdruck* und sein *Wert* sind verschiedene Dinge:

   Der *Ausdruck* `(a|b)*a` hat den *Wert* $\{w \mid w$ besteht aus a's und b's und endet auf a$\}$.

2. Der Ausdruck `hallo` hat als Wert gerade die Ein-Wort-Sprache $\{$hallo$\}$.

1.4.3 Anwendungen

Einsatzgebiete

Einsatzgebiete von regulären Ausdrücken.

- Innerhalb von Programmen werden reguläre Ausdrücke zur *Eingabekontrolle* benutzt.
- Innerhalb von Programmen werden reguläre Ausdrücke zum *Parsen von Eingaben* benutzt.

  Beispiel: Zerlegung einer URL in ihre Bestandteile.
- Menschen können reguläre Ausdrücke für *Suchanfragen* benutzen.
  - In der Datenbanksprache »SQL« kann man statt `like` auch *`regexp`* benutzen, um nach Attributen zu filtern.
  - In Microsoft Word kann man im Suchen-Dialog einen *Mustervergleich* einschalten.
  - Das Programm `grep` sucht nach einem Vorkommen eines regulären Ausdrucks in einer Datei.

Probleme beim Einsatz von regulären Ausdrücken.

Notationsprobleme:

- Will man reguläre Ausdrücke aufschreiben, so muss man irgendwie alles mit ASCII-Zeichen aufschreiben.
- Leider hat sich keinerlei Standard zum Aufschreiben von regulären Ausdrücken durchgesetzt.

Effizienzprobleme:

- Manche Arten regulärer Ausdrücke lassen sich *effizienter behandeln* als andere.
- Deshalb ist es manchmal sinnvoll, nur *bestimmte Arten von regulären Ausdrücken zuzulassen*.
- Einige Programme vereinfachen deshalb die Notation, sind dann aber inkompatibel mit anderen Notationen.

Typische Notationen für reguläre Ausdrücke.

Auch wenn die Notation für reguläre Ausdrücke nicht einheitlich ist, so gilt wenigstens *oft*:

- Statt `a|b|c|d|e|x|y|z` kann man auch schreiben `[abcdexyz]` oder auch kürzer `[a-exyz]`.
- Statt einer Aufzählung aller möglichen Zeichen kann man einfach einen Punkt schreiben, er steht für ein beliebiges Zeichen.
- Statt `(|`A`)`, was »gar kein Vorkommen von A oder ein Vorkommen von A« bedeutet, kann man schreiben A`?`.
- Statt AA`*`, was »mindestens ein Vorkommen von A« bedeutet, kann man schreiben A`+`.
- Schreibt man $A\{n\}$, wobei n eine Zahl ist, so bedeutet dies »genau n Vorkommen von A«.
- Schreibt man $A\{n, m\}$, so bedeutet dies »mindestens n und höchstens m Vorkommen von A«.
- Sucht man tatsächlich nach einem Sonderzeichen, so kann man ihm einen Backslash voranstellen.

Referenz: Reguläre Ausdrücke in Grep

Die folgenden Regeln geben einen kurzen Überblick über die genaue `grep`-Regel-Syntax. In der darauf folgenden Tabelle sind einige Beispiele angegeben, die diese Regeln verdeutlichen.

1. Die Sprache aller Zeichenketten, die das Zeichen `a` enthalten, wird einfach durch den regulären Ausdruck `a` beschrieben. Dasselbe gilt für alle anderen Zeichen, sofern sie keine Sonderzeichen sind.
2. Die folgenden Zeichen sind Sonderzeichen:

   `\ . ^ $ [ ] ( ) * ? + | /`

   Sonderzeichen haben besondere Funktionen in regulären Ausdrücken. Um trotzdem Sprachen zu definieren, die diese Zeichen enthalten, stellen Sie dem entsprechenden Zeichen den Backslash `\` voran. *(Beispiel 1)*
3. Das Sonderzeichen `.` steht für genau ein beliebiges Zeichen (auch ein Leerzeichen). *(Beispiele 3, 14, 15)*
4. Das Sonderzeichen `^` steht für den Anfang einer Zeichenkette. *(Beispiele 4, 12–15)*
5. Das Sonderzeichen `$` steht für das Ende eines Wortes. *(Beispiele 5, 12–15)*
6. Mit eckigen Klammern können mehrere Möglichkeiten für genau ein Zeichen definiert werden. *(Beispiele 6, 7, 13)*
7. Folgt auf eine öffnende eckige Klammer das Zeichen `^`, so wird die Zeichenmenge in der Klammer negiert (es bedeutet *hier* also *nicht* den Anfang der Zeichenkette). *(Beispiel 7)*
8. Zeichenmengen in eckigen Klammern können mit dem Zeichen `-` abgekürzt definiert werden. Statt `[0123456789]` kann man also `[0-9]` schreiben. *(Beispiel 13)*
9. Reguläre Ausdrücke werden verkettet, indem sie einfach hintereinander geschrieben werden. Der Ausdruck `ab` steht also für die Sprache aller Zeichenketten, die irgendwo den Teilstring `ab` enthalten. *(Beispiele 8–15)*
10. Die Zeichen `?`, `*` und `+` können regulären Ausdruck nachgestellt werden *(Beispiele 9–15)*

    `?` : Der vorangehende Ausdruck soll einmal oder keinmal vorkommen.

    `*` : Der vorangehende Ausdruck soll keinmal oder beliebig oft direkt hintereinander vorkommen.

    `+` : Der vorangehende Ausdruck soll einmal oder öfter direkt hintereinander vorkommen.

11. Durch runde Klammern können reguläre Ausdrücke gruppiert werden. Dies ist im Zusammenhang mit den Operationen `?`, `*`, `+` nützlich *(Beispiel 12)*: Ohne Gruppierung beziehen sich diese Operatoren immer nur auf das Zeichen direkt davor *(Beispiele 9–11)* beziehungsweise die eckige Klammer direkt davor *(Beispiel 13)*.
12. Mit dem Sonderzeichen `|` können zwei reguläre Ausdrücke vereinigt (»verodert«) werden. *(Beispiel 15)*

Beispiele für die Regeln:

	Ausdruck	*Die erzeugte Sprache enthält alle Zeichenketten, die …*
1	`\?`	… ein Fragezeichen enthalten.
2	`\\`	… einen Backslash enthalten.
3	`.`	… mindestens ein Zeichen lang sind.
4	`^a`	… mit `a` beginnen.
5	`a$`	… mit `a` enden.
6	`[abc]`	… irgendwo ein `a`, `b` oder `c` enthalten.
7	`[^abc]`	… irgendwo ein Zeichen enthalten, das kein ein `a`, `b` oder `c` ist.
8	`abc`	… den Teilstring `abc` enthalten.
9	`ab?a`	… den Teilstring `aa` oder den Teilstring `aba` enthalten.
10	`ab*a`	… einen der Teilstrings `aa`, `aba`, `abba`, `abbba` und so weiter enthalten.
11	`ab+a`	… einen der Teilstrings `aba`, `abba`, `abbba` und so weiter enthalten.
12	`^a(ab)*b$`	… mit einem `a` beginnen, gefolgt von beliebig vielen Wiederholungen von `ab`, und mit einem `b` enden.
13	`^[_a-zA-Z][_a-zA-Z0-9]*$`	… gültige Python-Bezeichner sind.
14	`^A.*s$`	… mit `A` beginnen und mit `s` enden.
15	`^A.*s$\|^.s$`	… mit `A` beginnen und mit `s` enden oder mit `s` enden und nur zwei Zeichen lang sind (nicht »entweder oder«!).

Zusammenfassung – Reguläre Ausdrücke

Reguläre Ausdrücke – Zusammenfassung

Worte und Sprachen

- Ein *Wort* ist eine endliche Folge von Symbolen über einem *Alphabet*.
- Eine *Sprache* ist eine beliebige (!) Menge von Worten über einem festen Alphabet.

Regulärer Ausdruck

- Ein *regulärer Ausdruck* R ist selbst ein Wort über einem speziellen Alphabet, das einige Sonderzeichen enthält.
- Er *beschreibt* eine Sprache $L(R)$ nach den in diesem Kapitel beschriebenen Regeln.

Das Muster-Such-Problem

Beim Muster-Such-Problem hat man einen regulären Ausdruck R gegeben und einen Text und man *sucht* nach Vorkommen von Worten aus $L(R)$ in dem Text. Das Problem ist auch bei *Eingabekontrollen* wichtig.

Vorhaben 2

Kommunikation – Netzwerke[1]

2.1 Welche Kompetenzen sollen Sie in diesem Vorhaben erwerben?[2]

Die Schülerinnen und Schüler

- beschreiben und erläutern Netzwerk-Topologien, die Client-Server-Struktur und Protokolle sowie ein Schichtenmodell in Netzwerken (IF4, A).
- untersuchen und bewerten anhand von Fallbeispielen Auswirkungen des Einsatzes von Informatiksystemen sowie Aspekte der Sicherheit von Informatiksystemen, des Datenschutzes und des Urheberrechts (IF5, A),
- untersuchen und bewerten Problemlagen, die sich aus dem Einsatz von Informatiksystemen ergeben, hinsichtlich rechtlicher Vorgaben, ethischer Aspekte und gesellschaftlicher Werte unter Berücksichtigung unterschiedlicher Interessenlagen (IF5, A)
- analysieren und erläutern Algorithmen und Methoden zur Client-Server-Kommunikation (IF2 LK, A),
- ***entwickeln und implementieren Algorithmen und Methoden zur Client-Server-Kommunikation (IF2 LK, I),***
- analysieren und erläutern Protokolle zur Kommunikation in einem Client-Server-Netzwerk (IF4 LK, A),
- entwickeln und erweitern Protokolle zur Kommunikation in einem Client-Server-Netzwerk (IF4, M).

[1] Teile dieses Abschnitts wurden den Planungsunterlagen der Veranstaltung »Einführung in die Informatik – Teil 1« aus dem Wintersemester 2012/2013 von Prof. Dr. Till Tantau zu dem Thema **Rechnernetze und das Internet** entnommen und an einigen Stellen geändert.

[2] Die in den Kompetenzen hervorgehobenen Elemente sind für den Leitungskurs vorgesehen.

In dem Vorhaben über Datenbanken haben wir gelernt, dass Informatiksysteme viel mit Verwaltungsbeamten zu tun haben. Ein interessanter Aspekt fehlt aber noch: Fast seit den Anfängen der Informatik *reden Informatiksysteme gerne mit anderen Informatiksystemen*. In der Tat scheint der Tag nicht mehr fern, an dem ein »normales« Informatiksystem gar nicht mehr »ohne Internet kann«. Insofern sind Informatiksysteme schon heute sehr soziale Wesen, die gerne mal ein ausführliches Schwätzchen darüber halten, was auf ihren Speichermedien so alles los ist.

Damit Informatiksysteme miteinander reden können, müssen sie sich erstmal auf eine Sprache einigen. Hier herrscht eine geradezu babylonische Sprachverwirrung: Redet ein Informatiksystem in der Sprache »AppleTalk« (gibt es wirklich) auf ein anderes Informatiksystem ein, das lieber »TCP/IP« spricht, so erscheint letzterem dies als »wirres Geschwätz«. Aus diesem Grund benötigt man *Protokolle* und gegebenenfalls *Übersetzer*.

Informatiksysteme sind eigentlich sehr gut in der Lage, sich alleine miteinander zu unterhalten. Gelegentlich will aber auch ein Mensch mit einem Informatiksysteme reden, was dieser Tage in der Regel dadurch geschieht, dass man einen Webserver »ansurft« und sich dessen Webseiten anschaut. War dies früher noch ein technisch recht einfacher Vorgang, sind die Dinge im Zeitalter des Web 2.0, wo jeder mitmachen kann/will/soll, komplizierter geworden: Webseiten werden *dynamisch gebaut* – wie dies geschieht, wollen wir uns anschauen.

2.2 Rechnernetze und das Internet

- Konzept des Kommunikationsprotokolls kennen.
- OSI-Schichtenmodell kennen.
- Unterschied zwischen LAN und Internet kennen.
- Vor- und Nachteile netzbasierter Projekte und Ansätze beurteilen können.

Praktisch jedes Informatiksystem, das in der Schule steht, ist an das Internet angeschlossen. Über Kupferkabel, Glasfaserkabel oder per Funk sind die Informatiksysteme mit dem Netz der Netze verbunden – und kommunizieren dabei fleißig mit anderen Informatiksystemen, selbst wenn man dies gar nicht vermuten würde. Das aller Mindeste ist, dass die Informatiksysteme durch eine Art »Ich-lebe-noch-Nachricht« den restlichen Systemen versichern, dass sie noch Strom haben. Gleichzeitig müssen sie sich so genannter Port-Scans erwehren, über die böse Programme Herrschaft über sie erlangen wollen (mehr dazu später im Vorhaben 3 über Sicherheit). Richtig geschwätzig werden Informatiksysteme, wenn sie für einen Menschen Daten zu anderen Informatiksystemen schicken.

Das weltweite Volumen der Kommunikation zwischen Informatiksystemen in jeder Sekunde ist wahrhaft gigantisch, es steigt exponentiell an und ein Ende ist erstmal nicht in Sicht. Diese Datenströme unter Kontrolle zu halten, ist die Aufgabe von Rechnernetzen und ihrer Kommunikationsprotokolle.

Das »Design« eines Rechnernetzes ist eine komplexe Aufgabe. Es sollte effizient sein, ausfallsicher und für zukünftige Erweiterungen offen. Das Hauptprotokoll des Internets, das den einfach zu merkenden Namen *Internet-Protocol* trägt, hat die ersten beiden Eigenschaften: Nachrichtenpakete lassen sich sehr effizient herstellen, analysieren und verteilten. Die Ausfallsicherheit wird wirklich großgeschrieben.

Betreffend Zukunftssicherheit schneidet das Internet-Protokoll nicht sonderlich gut ab. Im Design war beispielsweise vorgesehen, dass es maximal 4 Milliarden Informatiksysteme im Netz geben wird. Zum Zeitpunkt des Designs mag dies eine unglaublich große Zahl gewesen sein, heute ärgern sich *sehr* viele Leute über diese katastrophale Unterdimensionierung.

2.3 Informatiksysteme, die miteinander reden

2.3.1 Was reden Informatiksysteme miteinander?

Was haben Informatiksysteme miteinander zu besprechen?

- Die ersten Informatiksysteme waren nur in der Lage, sich »mit ihrem Anwender« zu unterhalten.
- Schon bald begann man aber, Informatiksysteme *miteinander zu verbinden*, beispielsweise über eine Telefonverbindung mittels zweier *Modems* (Modulator-Demodulator).
- Heute sind Informatiksysteme *sehr oft* über Netze, insbesondere durch das Internet miteinander verbunden.
- In der Zukunft werden Informatiksysteme vermutlich *fast immer* mit anderen Informatiksystemen verbunden sein.

2.3.2 Wie reden Informatiksysteme miteinander?

Wie reden zwei Informatiksysteme miteinander?

Damit zwei Informatiksysteme miteinander reden können, benötigt man:

- Eine physikalische Verbindung, entweder durch Kabel oder durch Funk.

- Eine gemeinsame »Sprache«, technikverliebt *Kommunikationsprotokoll* genannt.
- Kommunikationsprotokolle können *unterschiedlich kompliziert* sein und *unterschiedliche Aspekte* der Kommunikation regeln.
- Beispiele von Protokollen sind *Ethernet*, ATM, DSL, *TCP*, *IP*, *HTTPS* oder *ssh*.

Wie reden viele Informatiksysteme miteinander?

- Wir wollen nun *viele* Informatiksysteme miteinander reden lassen (wie im Internet, Netz der Schule, etc.).
- Das Hauptproblem ist dann, dass man nicht je zwei Informatiksysteme direkt durch Kabel verbinden kann.

Dieses Problem löst man wie folgt:

1. Die Informatiksysteme werden durch ein *Kommunikationsnetz* verbunden.
2. Will ein Informatiksystem mit einem anderen reden, so schickt es diesem eine *Nachricht*.
3. Diese Nachricht wird *durch das Netz gereicht*, sehr ähnlich einer *Postzustellung*.
4. Die Informatiksysteme im Netz fungieren sowohl als *Zusteller* als auch als *Absender* als auch als *Empfänger*.

2.3.3 Das Schichtenmodell

Die unterschiedlichen Aspekte von Kommunikation.

Ein Informatiksystem möchte einem anderen Informatiksystem eine Nachricht schicken. Dann müssen sich die Informatiksysteme über verschiedene Dinge einig sein:

1. Das physikalische Protokoll.

   Welche Spannungen, welche Geschwindigkeit wird benutzt?
2. Das Vermittlungs-Protokoll.

   Wo steht die Adresse auf einer Nachricht? Wie sind Adressen formatiert?
3. Das Transport-Protokoll.

   Wo und wie beschwert man sich, wenn eine Nachricht nicht ankommt?

Das OSI-Schichtenmodell legt sieben mögliche Protokollebenen fest.

1. Die *Bitübertragungsschicht*.
2. Die *Sicherungsschicht*.
3. Die *Vermittlungsschicht*.
4. Die *Transportschicht*.
5. Die *Sitzungsschicht*.
6. Die *Darstellungsschicht*.
7. Die *Anwendungsschicht*.

Protokolle regeln im Allgemeinen, wie die Kommunikation auf ein oder zwei dieser Schichten funktioniert. An einer Kommunikation sind folglich oft mehrere Protokolle beteiligt.

Verschicken von 10 Tonnen Äpfeln

1. Die *Bitübertragungsschicht*: Werden die Äpfel in Lastwagen oder im Flugzeug transportiert? In welche Kisten werden sie gepackt?
2. Die *Sicherungsschicht*: Wie stellt man sicher, dass die Äpfel nicht gedrückt werden und nicht mit anderen Ladungen vermischt werden?
3. Die *Vermittlungsschicht*: Wie schreibt man die Adresse auf die Kisten? Benutzt man vier- oder fünfstellige Postleitzahlen?
4. Die *Transportschicht*: Was passiert, wenn eine Kiste verschwindet? Wie wird erreicht, dass eine neue Kiste der richtigen Sorte geschickt wird?
5. Die *Sitzungsschicht*: Wie wird das Apfel-Verkaufs-Geschäft abgewickelt? Wann gilt der Vertrag als erfüllt?
6. Die *Darstellungsschicht*: –
7. Die *Anwendungsschicht*: Wie funktioniert der Apfelhandel?

Verschicken von Webseiten im OSI-Schichtenmodell

1. Die *Bitübertragungsschicht*: Welche Spannung wird auf den Leitungen verwendet? Welche Signalstärke beim Funk?
2. Die *Sicherungsschicht*: Wie stellt man sicher, dass keine physikalischen Störungen auftreten? Was tun, wenn sie doch auftreten?
3. Die *Vermittlungsschicht*: Wohin schreibt man die Adresse der Nachrichten, mit deren Hilfe die Webseiten versandt werden?
4. Die *Transportschicht*: Was passiert, wenn Bild oder ein Teil der Webseite verschwindet?
5. Die *Sitzungsschicht*: Woher weiß der Bahn-Webserver, dass man sich bei ihm angemeldet hat?
6. Die *Darstellungsschicht*: In welchem Format sind die Webseiten geschrieben? (Zum Beispiel HTML oder PDF?)
7. Die *Anwendungsschicht*: Welcher Browser wird verwendet?

Zu welchen Schichten gehören folgende Protokolle/Anwendungen?

1. E-Mail
2. DSL
3. Python-Shell
4. WLAN (schnurloses Surfen)

2.4 Lokale Netze

2.4.1 LAN und WLAN

Local Area Networks – LANs.

- Ein *Local Area Network* ist ein Netzwerk in dem eine kleine Anzahl Informatiksysteme verbunden sind.
- So werden die Informatiksysteme einer Wohnung oder eines Gebäudes als LAN verbunden.
- Beispielsweise bilden die Informatiksysteme im Informatikraum ein LAN, genauso die Informatiksysteme des Selbstlernzentrums.
- LANs werden von einer *Administratorin verwaltet*.

- LANs sind sehr schnell, da die Kabellängen gering sind und nur wenige Informatiksysteme miteinander reden können.
- Ein *Wireless Local Area Network* ist ebenfalls ein LAN, nur ist die Verbindung per Funk. Die Verwaltung funktioniert genauso.

2.4.2 Das Protokoll Ethernet

Das Protokoll der LANs: Ethernet

- LANs benutzen in aller Regel das in den 80'ern entwickelte *Ethernet-Protokoll*.
- Es regelt die ersten beiden OSI-Schichten, also das Physikalische der Übertragung.
- Ethernet überträgt zwischen 100MBit pro Sekunde bis 1GBit pro Sekunde.
- Die Netztopologie ist ein Bus: Ein langes Kabel, an dem alle Informatiksysteme durch Stichleitungen hängen.

2.5 Globale Netze

2.5.1 Das Internet

Geschichte des Netz der Netze.

- Das *Internet* ist ein globales Netz, das lokale Netze miteinander verbindet.

  Es ist *das einzige solche Netz*, weshalb es auch schlicht *das Netz* genannt wird.
- Es ist aus dem militärischen ARPANET (advanced research projects agency net) hervorgegangen.
- Die Anzahl der Informatiksysteme, die über das Netz verbunden sind, wächst exponentiell an.
- Der Begriff *Online-Sein* ist mittlerweile gleichbedeutend damit, mit dem Internet verbunden zu sein.

Designkriterien des Internet.

- Jedes Informatiksystem hat eine *Internet-Adresse*.
  - IP-Adressen sind Nummern, bestehend aus vier Bytes.
  - Diese werden als Dezimalzahlen durch Punkte getrennt aufgeschrieben.
  - Beispielsweise hat dieses Informatiksystem die Adresse 132.195.95.10.
- Das Internet ist *weitgehend dezentral* organisiert – lediglich die Vergabe der Adressräume an Firmen und Universitäten wird zentral durchgeführt.
- Das Internet ist *redundant vernetzt*.
  - Es gibt also viele Möglichkeiten, eine Nachricht von einem Informatiksystem zu einem anderen zu schicken.
  - Fällt ein Informatiksystem aus, so übernehmen andere Informatiksysteme automatisch die Vermittlung.
- Das Internet arbeitet *paketorientiert*.
  - Es werden grundsätzlich nur Pakete zwischen IP-Adressen vermittelt.
  - Pakete haben eine Maximallänge; längere Nachrichten werden in viele kleine Pakete aufgeteilt.

2.5.2 Die Protokolle IP und TCP

Das Internet-Protokoll (IP).

- Das IP ist ein Stufe-3-Protokoll (Vermittlungsschicht).
- Seine Aufgabe ist folgende:
  - Ein Informatiksystem mit der Adresse A möchte ein Paket an einen Informatiksystem mit der Adresse B schicken.
  - Dazu baut es ein *Paket*, in dessen *Adressfeld* die Adresse B steht.
  - Dieses Paket reicht es an den nächsten *Router* weiter, den A kennt.
  - Der Router schaut in seiner *Routingtabelle* nach, ob er die Adresse B kennt. Wenn ja, stellt er das Paket direkt zu.
  - Sonst schickt er das Paket an den nächsten Router »in Richtung des Zieles«, dieser verfährt analog.
- Das IP garantiert *nicht*, dass ein Paket auch ankommt.

- Das IP garantiert *nicht*, dass mehrere nacheinander geschickte Pakete in derselben Reihenfolge ankommen.

Ein Paket nach Bergkamen.

```
traceroute to gw.stadt-bergkamen.de (212.37.49.49)
 1  math-gw.urz.uni-wuppertal.de (132.195.95.254)
 2  transfer0.lan.uni-wuppertal.de (132.195.254.62)
 3  xr-wup1-te1-3-888.x-win.dfn.de (188.1.234.13)
 4  cr-fra2-te0-9-0-6-8.x-win.dfn.de (188.1.146.37)
 5  te0-1-0.decix1.as12355.net (80.81.192.240)
 6  te0-0-1-3.pffm1.as12355.net (212.37.63.68)
 7  te2-2.pelue1.as12355.net (212.37.63.113)
 8  te2-3.pekam1.as12355.net (212.37.63.13)
 9  gw.stadt-bergkamen.de (212.37.49.49)
```

Ein Paket in die Neue Welt.

```
traceroute to pgf.cvs.sourceforge.net (216.34.181.112)
 1  math-gw.urz.uni-wuppertal.de (132.195.95.254)
 2  transfer0.lan.uni-wuppertal.de (132.195.254.62)
 3  xr-wup1-te1-3-888.x-win.dfn.de (188.1.234.13)
 4  xr-dor1-te2-1.x-win.dfn.de (188.1.144.49)
 5  xr-boc1-te2-1.x-win.dfn.de (188.1.144.46)
 6  cr-dui1-te0-7-0-0.x-win.dfn.de (188.1.146.33)
 7  cr-fra2-hundredgige0-9-0-3.x-win.dfn.de (188.1.144.178)
 8  ae53.edge5.Frankfurt1.Level3.net (212.162.4.5)
 9  ae-4-90.edge4.Frankfurt1.Level3.net (4.69.154.200)
10  Savvis-level3-1x10G.frankfurt.Level3.net (4.68.111.174)
11  cr2-te-0-1-1-0.ft3.savvis.net (206.28.100.85)
12  206.28.96.162 (206.28.96.162)
13  hr4-xe-8-0-0.elkgrovech3.savvis.net (204.70.195.122)
14  das6-v3035.ch3.savvis.net (64.37.207.170)
15  64.27.160.194 (64.27.160.194)
16  cvs.sourceforge.net (216.34.181.112)
```

Das Transmission-Control-Protocol (TCP).

- Das TCP ist ein Stufe-4-Protokoll (Transportschicht).
- Seine Aufgaben sind folgende:

  - Ein Informatiksystem mit der Adresse A möchte eine *lange Nachricht* an ein Informatiksystem mit der Adresse B schicken.
  - Dazu zerlegt TCP die Nachricht in viele *Pakete*.
  - Diese verschickt es mittels IP an B.
  - Alle Pakete werden mit *Seriennummern* versehen.
  - Am Ziel *sortiert* TCP die Pakete, falls sie von IP durcheinander gebracht wurden.
  - Falls IP Pakete verloren gehen, fordert sie TCP erneut an.
- Da man fast immer beide Protokolle zusammen benutzt, spricht man auch (etwas falsch) vom *TCP/IP*-Protokoll.

2.5.3 Domains

Von Zahlen und Namen.

- Die Internet-Adressen sind für Menschen schwer zu merken.
- Deshalb existiert ein *Namensdienst* für das Internet, der *Domain-Name-Service (DNS)*.
- Seine Aufgabe ist es, *Namen* wie `l119v.studs.math.uni-wuppertal.de` in IP-Adressen wie 132.195.133.147 zu übersetzen.
- DNS ist hierarchisch. Die Übersetzung von `l119v.studs.math.uni-wuppertal.de` funktioniert wie folgt:
  - Zunächst wird beim Root-Server nachgefragt, welcher Server für `de` zuständig ist.
  - Dann wird bei diesem nachgefragt, wer innerhalb von `de` für `uni-wuppertal` zuständig ist.
  - Dann wird bei diesem nachgefragt, wer für `math` zuständig ist.
  - Dann wird bei diesem nachgefragt, wer für `studs` zuständig ist.
  - Dieser weiß dann die Adresse von `l119v`.

Wie erreicht man, dass der Root-Server nicht mit Milliarden von Anfragen pro Sekunde bombardiert wird?

2.5.4 Internet-Dienst E-Mail

Der Internet-Dienst »E-Mail«.

- *E-Mail* ist ein Protokoll der Schicht 5 (Sitzungsschicht).
- E-Mail wird *mittels TCP/IP verschickt*, man *könnte sie aber auch anders verschicken*. (Was früher auch geschah.)
- Der hintere Teil einer *E-Mail-Adresse* ähnelt einem Internet-Namen:
  - Bei `humbert@uni-wuppertal.de` *scheint* `uni-wuppertal.de` der Name eines Servers zu sein.
  - Er ist es aber nicht – *E-Mail hat einen eigenen Namensraum, der dem des Internet nur ähnelt*.

2.5.5 Internet-Dienst WWW

Der Internet-Dienst »WWW«.

- *WWW* benutzt das Protokoll HTTP, welches ein Schicht 5 Protokoll (Sitzungsschicht) ist.
- HTTP verschickt Webseiten in der Regel *mittels TCP/IP*.
- Der mittlere Teil einer *WWW-Adresse* enthält einen Internet-Namen:
  - Bei `http://ddi.uni-wuppertal.de/personen/humbert.html` ist tatsächlich `www-madin.math.uni-wuppertal.de` der Name eines Informatiksystems (nämlich 132.195.116.183).
  - Es ist *Aufgabe des Web-Servers*, den *lokalen Teil der Adresse*, also beispielsweise `/personen/humbert.html`, in eine lokale Datei auf dem Informatiksystem 132.195.116.183 zu übersetzen.

2.5.6 Internet-Dienst ssh

Der Internet-Dienst »ssh«.

- *ssh* (secure shell) ist ein Protokoll der Schicht 5 (Sitzungsschicht).
- Es dient dazu, eine *sichere Verbindung* zu einem Informatiksystem aufzunehmen.
- Dazu gibt man einfach den Namen oder die IP-Adresse eines Informatiksystems an, mit dem man sich verbinden will.

- Dann wird eine Verbindung mit diesem Informatiksystem aufgebaut und man kann arbeiten »als säße man an diesem Informatiksystem«.
- Gibt man noch die Option `-X` an und ist alles korrekt eingestellt, kann man so sogar Fenster statt auf dem entfernten Informatiksystem auf dem eigenen Informatiksystem anzeigen.

Beispiel

```
ssh -X wmai10.math.uni-wuppertal.de
```

Zusammenfassung – Rechnernetze und das Internet

Rechnernetze und das Internet – Zusammenfassung

1. Informatiksysteme unterhalten sich miteinander unter Benutzung von *Protokollen*.
2. Es gibt Protokolle für *verschiedene Ebenen*.
   - Auf der untersten Ebene gibt es beispielsweise *Ethernet*.
   - Auf der mittleren Ebene gibt es beispielsweise *TCP/IP*.
   - Auf der obersten Ebene gibt es beispielsweise *E-Mail* und *HTTP*.
3. Informatiksysteme im Internet sind durch *IP-Adressen* eindeutig identifiziert, *Domain-Namen* dienen nur der Bequemlichkeit von Menschen.

Vorhaben 3

Kryptologie[1]

3.1 Welche Kompetenzen sollen Sie in diesem Vorhaben erwerben?[2]

Die Schülerinnen und Schüler

- analysieren und erläutern Eigenschaften, ***Funktionsweisen*** und Einsatzbereiche symmetrischer und asymmetrischer Verschlüsselungsverfahren (IF4, A).

Über manche Dinge sollte man besser schweigen. Stellen Sie sich vor, Ihr Informatiksystem teilt freudig anderen Informatiksystemen alles mit, was auf Ihren Speichern so steht. Dort finden sich höchstwahrscheinlich Daten zur Ihren E-Mails, Ihre Passwörter und vermutlich auch über Ihre Liebhaber (definitiv aber darüber, wen Sie gerne als Liebhaber hätten). Aus diesem Grund ist die *Sicherheit von Informatiksystemen*[3] ein wichtiges Thema. Dabei geht es darum, Zugriff durch Unbefugte zu verhindern, aber auch darum, wie man Daten ganz banal vor, sagen wir, Feuer schützt. Eines der spannendsten Teilgebiete der Sicherheit von Informatiksystemen ist die Kryptologie, also die Frage, wie man Daten verschlüsselt, entschlüsselt und die verschlüsselten Daten *knackt*. Dieses Gebiet ist nicht nur deshalb spannend, weil es nach Geheimdiensten und Spionagethrillern klingt, sondern weil es auch wirklich schöne Anwendungen von reiner Mathematik in der Praxis darstellt.

3.2 Kommunikationssicherheit

Verschlüsselung: Der digitale Briefumschlag

[1] Teile dieses Abschnitts wurden der Veranstaltung »Einführung in die Informatik – Teil 1« aus dem Wintersemester 2012/2013 von Prof. Dr. Till Tantau zu den Thema **Sicherheit** entnommen (dort: Kapitel 45 und 46) und an einigen Stellen geändert.

[2] Die in den Kompetenzen hervorgehobenen Elemente sind für den Leitungskurs vorgesehen.

[3] Diese Eigenschaft wird oft auch als *Computersicherheit* bezeichnet.

Detailkompetenzen

- Konzept der Verschlüsselung verstehen
- Unterschied zwischen symmetrischen und asymmetrischen Verfahren kennen
- Konzept der digitalen Unterschrift verstehen
- Programme zur Verschlüsselung einsetzen können

Verschlüsselung von Daten gehört seit der Antike zu den Grundmethoden der Kriegsführung. Eine sehr alte (und sehr einfache) Verschlüsselungsmethode ist beispielsweise die Cäsar-Chiffre, bei der einfach jeder Buchstabe durch einen Buchstaben etwas weiter hinten im Alphabet ersetzt wird. Ob sich Julius Cäsar dieses Verfahren allerdings selbst ausgedacht hat? Berthold Brechts lesender Arbeiter würde sich sicherlich fragen: »Cäsar eroberte ganz Gallien. Hatte er nicht wenigstens einen Kryptographen dabei?«

Im zweiten Weltkrieg wurde erstmals im großen Stil Kryptographie *technisiert*, insbesondere in Form der *Enigma*[4]. Es entwickelte sich ein Katz-und-Maus-Spiel, bei dem die deutschen Truppen ihre Enigma kryptographisch immer sicherer machten, während die Alliierten ihre *Kryptoanalyse* gleichzeitig immer weiter verbesserten. Die meisten mit Hilfe der Enigma verschlüsselten Funksprüche der deutschen Truppen konnten von den Alliierten mitgelesen werden. Zur Bedeutung des Brechens des Enigma-Codes: »Die Kompromittierung der Enigma wird als ein strategischer Vorteil angesehen, der den Alliierten den Gewinn des Krieges erheblich erleichtert hat. Es gibt sogar Historiker, die diese Tatsache für kriegsentscheidend halten, denn die Entzifferungen waren nicht nur auf militärisch-taktischer Ebene (Heer, Luftwaffe und Marine) eine große Hilfe, sondern sie erlaubten aufgrund der nahezu vollständigen Durchdringung des deutschen Nachrichtenverkehrs auf allen Ebenen (Polizei, Geheimdienste, diplomatische Dienste, SD, SS, Reichspost und Reichsbahn) auch einen genauen Einblick in die strategischen und wirtschaftlichen Planungen der deutschen Führung. Speziell schätzten die Alliierten die Authentizität der aus Enigma-Funksprüchen gewonnenen Information, die aus anderen Quellen, wie Aufklärung, Spionage oder Verrat, nicht immer gegeben war. So konnten die Briten ihre zu Beginn des Krieges noch begrenzten Ressourcen optimal koordinieren und gezielt gegen die deutschen Schwächen einsetzen, und später, zusammen mit ihren amerikanischen Verbündeten, die Überlegenheit noch besser ausspielen« (*Enigma (Maschine)* 2008, Abschnitt: Geschichtliche Konsequenzen).

Einer der Hauptgründe, weshalb der Enigma-Code gebrochen werden konnte, war, dass das folgende – schon 1883 von Kerckhoff formulierte – Prinzip nicht eingehalten wurde:

[4] Die Bezeichnung *Enigma* kommt aus dem Griechischen (αίνιγμα) und bedeutet Rätsel.

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

Die heute verwendeten Verschlüsselungsalgorithmen haben diese Schwachstelle nicht, man kann die Algorithmen in jedem Lehrbuch nachlesen. Trotzdem sei darauf hingewiesen, dass ihre Sicherheit nur *vermutet* wird – es gibt keinen Beweis, dass irgend eines der heute eingesetzten Standardverfahren wirklich sicher ist.

3.2.1 Ziele der Sicherheit von Informatiksystemen

Worum geht es bei Sicherheit von Informatiksystemen?

Bei der *Sicherheit von Informatiksystemen* geht es um folgende Anliegen:

1. Schutz vor und die Aufrechterhaltung des Betriebs bei
   - Ausfall von Teilen des Systems (Stromausfall, Absturz)
   - Angriffen auf das System (durch Hacker, korrumpierte Mitarbeiter)

   Diese *Systemsicherheit* wird uns im nächsten Abschnitt interessieren.
2. Schutz von Daten und Kommunikation vor
   - Spionage
   - Fälschung

   Diese *Daten- und Kommunikationssicherheit* wird in diesem Abschnitt thematisiert.

3.2.2 Verschlüsselung

Ziele

Ziele der Verschlüsselung von Daten und Kommunikation

Vertraulichkeit Es muss sichergestellt werden, dass Daten und Kommunikation nur von »den Guten« gelesen werden können.

(Meine Daten gehen niemand etwas an.)

Integrität Es muss sichergestellt werden, dass Daten und Kommunikation nicht verfälscht werden können.

(Aus 1000 Euro dürfen nicht 10000 Euro werden.)

Authentizität Es muss sichergestellt werden, dass Daten und Kommunikation wirklich von den behaupteten Personen stammen.

(Die E-Mail mit Alices Absenderadresse wurde tatsächlich von Alice versandt;

der Online-Banking-Server muss wirklich der Server meiner Bank sein.)

Beurteilen Sie Postkarten, versiegelte Briefe und die Eröffnung eines Bankkontos bei einer Online-Bank in Bezug auf die drei Kriterien.

3.2.3 Symmetrische Verschlüsselung

Symmetrische Verschlüsselung mittels eines Schlüssels

- Zum Schutz vor unbefugtem Lesen kann man Daten und Kommunikation *verschlüsseln*.

  Dazu benutzt man ein *Verschlüsselungsverfahren* sowie einen *geheimen Schlüssel*.

- Beim *Verschlüsseln* (encryption) wird ein *Klartext* m (message) zusammen mit dem *Schlüssel* k (key) in ein *Chiffretext* $c = e(m, k)$ verwandelt.

  Dies entspricht dem »Abschließen« mit dem Schlüssel.

- Beim *Entschlüsseln* (decryption) wird der Chiffretext zusammen mit dem Schlüssel in den Klartext zurückverwandelt, d. h. $m = d(c, k)$.

  Dies entspricht dem »Aufschließen« mit dem Schlüssel.

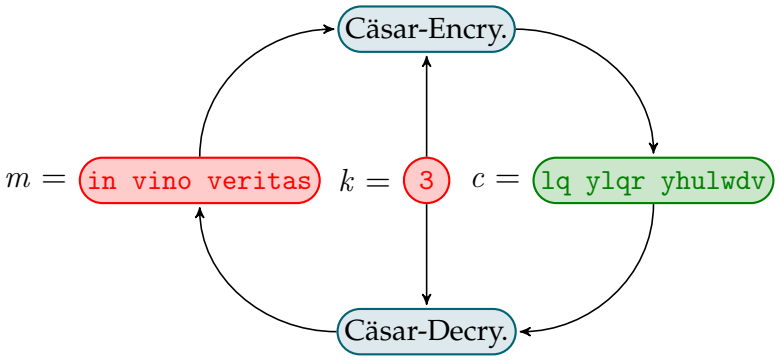
- Da man zum »Auf- und Zuschließen« denselben Schlüssel verwendet, spricht man von *symmetrischen Verfahren*.

Verfahren I: Die Cäsar-Chiffre

- Bereits in der Antike nutzte Julius Cäsar Verschlüsselungen, wenn er Befehle an seine Feldherren übermittelte.
- Das als *Cäsar-Chiffre* bekannte Verfahren funktioniert wie folgt: Jeder Buchstabe der Nachricht wird durch den Buchstaben ersetzt, der *drei Buchstaben später im Alphabet* kommt.
- Allgemeiner kann man statt »drei Buchstaben später« auch »k Buchstaben später« benutzen.
- Mathematisch ist also e die Funktion, die eine Nachricht und eine Zahl k nimmt und jeden Buchstaben der Nachricht um k viele Stellen im Alphabet vorwärts schiebt. Entsprechend schiebt d jeden Buchstaben um k viele Stellen zurück.

Betätigen Sie sich als *Kryptoanalytiker*! Wie kann man – ohne Kenntnis des Schlüssels k – eine mit einer Cäsar-Chiffre kodierte Nachricht entschlüsseln?

Beispiel einer Cäsar-Verschlüsselung



Rot = geheim, Grün = öffentlich

Verfahren II: Der One-Time-Pad

Da die Cäsar-Chiffre offenbar nicht sonderlich sicher ist, benötigen wir ein besseres Verfahren:

One-Time-Pad-Algorithmus (Ver- und Entschlüsselung)

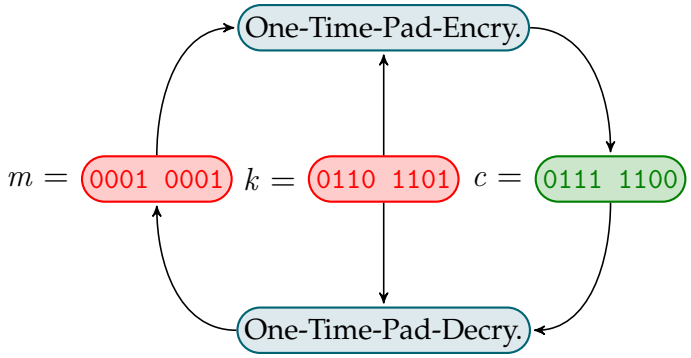
Eingaben seien die Nachricht m und der Schlüssel k *gleicher Länge*

1. Schreibe Nachricht und Schlüssel als Bitstrings auf (wie im ersten Abschnitt).
2. Bilde nun das bitweise XOR von Nachricht und Schlüssel. Dies bedeutet: Flippe das i-te Bit der Nachricht, wenn das i-te Bit des Schlüssels eine 1 ist.

- Wie der Name schon sagt, kann man das Verfahren leider nur einmal pro Schlüssel (sicher) verwenden.
- Außerdem sind die Schlüssel schrecklich lang.

Beispiel einer One-Time-Pad-Verschlüsselung

Rot = geheim, Grün = öffentlich



State-of-the-Art in Sachen symmetrische Verschlüsselung.

- Moderne symmetrische Verschlüsselungverfahren kann man sich als eine *sehr clevere Mischung* aus Cäsar-Verfahren und One-Time-Pad vorstellen.
- Lange Zeit war das amerikanische DES (data encryption standard) das wichtigste Verfahren. Wegen Problemen wie Ausfuhrverbot, zu kurze Schlüssellänge und Patenten wurde es vor gut 10 Jahren durch ein moderneres und sicheres Verfahren ersetzt.
- Der neue internationale Standard nennt sich AES (advanced encryption standard).

Sicherheit von Verschlüsselungsverfahren.

Wann ist ein Verfahren »sicher«?

- *Informationstheoretisch sicher* heißt ein Verfahren, wenn man Chiffretexte ohne Kenntnis des Schlüssels nicht entschlüsseln kann.
- Leider kann man zeigen, dass nur One-Time-Pad-Verfahren in diesem Sinn sicher sind.
- *Komplexitätstheoretisch sicher* heißt ein Verfahren, wenn es für die Entschlüsselung kein wesentlich schnelleres Verfahren gibt, als alle Schlüssel durchzuprobieren.
- Bei Schlüssellängen ab 500 Bits ist solch eine Sicherheitsstufe dann *in diesem Universum nicht zu brechen*.

3.2.4 Asymmetrische Verschlüsselung

Eine ungewöhnliche Idee.

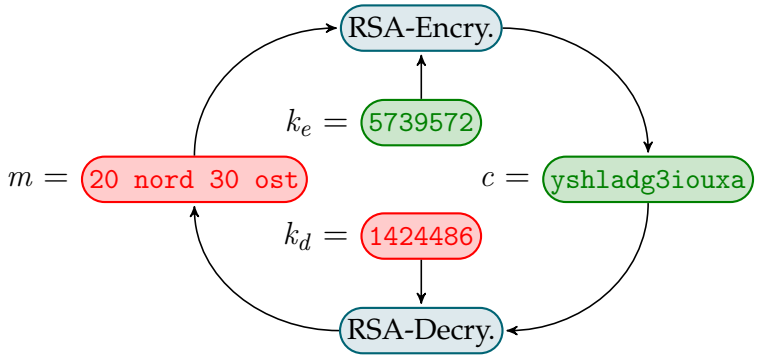
- Symmetrische Verschlüsselungsverfahren haben den *Nachteil*, dass Kommunikationspartner erstmal einen *Schlüssel sicher austauschen müssen*.
- Rivest, Shamir und Adleman haben 1977 ein Verfahren vorgeschlagen, *bei dem man keinen Schlüssel auszutauschen braucht* – das RSA-Verfahren. (Es gab vorher auch schon ein schlechteres, geheimgehaltenes Verfahren.)
- Heute ist dies ein Spezialfall der *Klasse der asymmetrischen Verschlüsselungsverfahren*.

Idee der asymmetrischen Verfahren

- Man benutzt *zwei Schlüssel*.
- Wenn man eine Nachricht mit einem ersten Schlüssel »abschließt«, kann man sie *nur mit dem zweiten Schlüssel »aufschließen«*.
- Kennt man einen der Schlüssel, so kann man daraus nicht mit vertretbarem Aufwand den anderen Schlüssel generieren.

Ablauf einer asymmetrischen Verschlüsselung.

- Die Royal Airforce möchte den Aufenthaltsort von Prinz Harry an den Buckingham Palace schicken.
- Dazu braucht sie nur den *öffentlichen Schlüssel 5739572 des Buckingham Palace* zu kennen.
- Nur im Palast kennt man den *privaten Schlüssel 1424486 des Palastes* und kann die Nachricht entschlüsseln.



3.2.5 Das El-Gamal-Verfahren

Im Folgenden soll das *El-Gamal-Verfahren* vorgestellt werden. Allerdings werden – der Darstellung von (Bongartz und Unger 2006) folgend – zum besseren Verständnis die mathematischen Operationen zunächst durch (zu) radikal vereinfachte ersetzt. Am Ende wird dann noch angedeutet werden, wie es in Wirklichkeit geht.

Grundideen

- Zum *Verschlüsseln* benutzen wir eine *einfache mathematische Operation*. Konkret wird dies *im Wesentlichen eine Multiplikation sein*.
- Zum *Entschlüsseln* muss man diese Operation also rückgängig machen, man muss also im Wesentlichen dividieren, was aber prinzipiell *schwerer* ist als Multiplizieren.
- Genauer nehmen wir an, dass Addition, Subtraktion und Multiplikation *leicht* sind, Division hingegen *sehr schwer*.
- Der *private Schlüssel* hilft uns aber, *statt der Division* die Nachricht doch schnell zu entschlüsseln.

Da das El-Gamal-Verfahren mathematische Operationen benutzt, müssen sinnigerweise sowohl Nachrichten wie Schlüssel auch mathematische Objekte sein, konkret Zahlen:

- Die Nachricht ist eine Zahl; beispielsweise $m = 5$.
- Der private Schlüssel ist eine Zahl; beispielsweise $k_d = 13$.
- Der öffentliche Schlüssel ist ein Paar, bestehend aus einer Zufallszahl z und dem Produkt dieser Zahl mit dem privaten Schlüssel; beispielsweise $k_e = (z, k_d \cdot z) = (11, 143)$.
- *Könnte man dividieren*, so kann man offenbar den privaten Schlüssel k_d leicht aus dem öffentlichen Paar k_e errechnen. Wir nehmen aber in der radikalen Vereinfachung an, dass man nicht effizient dividieren kann.

Eine *Verschlüsselung* funktioniert nun wie folgt:

- Zur Verschlüsselung wählt man *eine zufällige Zahl aus*, beispielsweise $s = 9$.
- Die Verschlüsselung der Nachricht $m = 5$ mit dem Schlüssel $k_e = (z, k_d \cdot z) = (11, 143)$ besteht aus zwei Teilen:

$$t_1 = m + s \cdot k_d \cdot z = 5 + 9 \cdot 143 = 1292$$
$$t_2 = s \cdot z = 9 \cdot 11 = 99$$

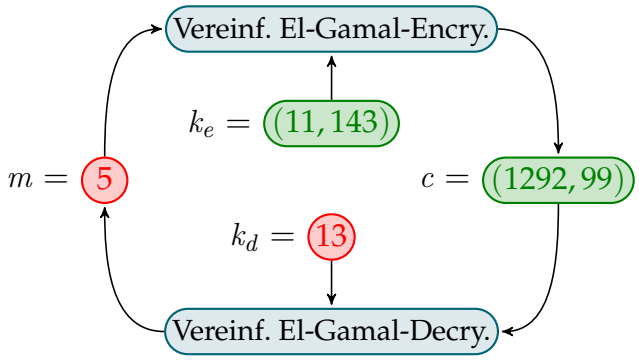
  Also ist die Verschlüsselung von 5 das Chiffrat $(1292, 99)$.

Umgekehrt läuft nun das Entschlüsseln wie folgt ab:

- Es gilt $t_1 = m + t_2 \cdot k_d$ und somit $m = t_1 - t_2 \cdot k_d$.
- Um also aus dem Chiffrat $(1292, 99)$ die Nachricht 5 zu rekonstruieren, kann man beim Entschlüsseln *unter Kenntnis von* k_d wie folgt rechnen:

$$m = t_1 - t_2 \cdot k_d = 1292 - 99 \cdot 13 = 5.$$

- Es wird nur addiert, subtrahiert und multipliziert.



Soweit zu dem vereinfachten El-Gamal-Verfahren. In Wirklichkeit müssen offenbar andere mathematische Operationen angewandt werden, da ja die Annahme, man könne nicht effizient dividieren, recht unrealistisch ist.:

- Im *echten El-Gamal-Verfahren* werden die Grundoperationen wie folgt ersetzt (p sei eine große Primzahl – es gibt Verfahren, solche Primzahlen schnell zu bestimmen):
  1. Addition wird ersetzt durch »Modulo-Multiplikation«:
     $a + b$ wird zu $(a \cdot b) \bmod p$.
  2. Subtraktion wird ersetzt durch »Modulo-Division«:
     $a - b$ wird zu $(ab^{p-2}) \bmod p$.
  3. Multiplikation wird ersetzt durch »Modulo-Exponentenbildung«:
     $a \cdot b$ wird zu $a^b \bmod p$.
  4. Division wird ersetzt durch den »Modulo-Logarithmus«.
- Man überlegt sich nun recht leicht, dass zunächst die Modulo-Multiplikation recht flott berechnet werden kann, selbst wenn die Zahlen a, b und p alle einige Hundert Stellen haben. Dazu benutzt man einfach das schriftliche Multiplizieren und Dividieren, wie man es aus der Schule kennt.

- Für die Modulo-Division und -Exponentenbildung ist zunächst nicht klar, wie diese schnell funktionieren sollen: Es ist in der Tat nicht möglich, in vertretbarer Zeit a^b zu berechnen, wenn a und b jeweils hunderte Ziffern haben. Der Grund ist einfach, dass in diesem Fall das Ergebnis länger ist als es Atome im Universum gibt.

  Nun soll aber auch nicht a^b berechnet werden, sondern a^b mod p. Der Trick ist grob folgender: Will man beispielsweise a^{256} mod p berechnen, so kann man dies schnell machen, indem man nacheinander a bin p, a^2 bin p, a^4 bin p, a^8 bin p, a^{16} bin p, a^{32} bin p, a^{64} bin p, a^{128} bin p und a^{256} bin p berechnet. Jede Zahl in der Reihe ergibt sich, indem man die vorherige Zahl quadriert und modulo p rechnet. Will man also a^b bin p berechnen für ein b mit tausend Stellen, so muss man diese Operation etwa eintausend Mal ausführen – was noch eine vertretbare Laufzeit ergibt.

Entscheidend für die Sicherheit des El-Gamal-Verfahrens ist nun, dass man für die Berechnung des Modulo-Logarithmus schlicht kein schnelles Verfahren kennt.

3.2.6 Sichere E-Mail

Vertraulichkeit: Digitale Briefumschläge

Eine Nachricht soll vertraulich sein.

Ziel: Vertraulichkeit

- Eine Nachricht soll einer Person zugestellt werden.
- Nur diese Person soll die Nachricht lesen können.

Dies nennt man auch einen *digitalen Briefumschlag*.

Methode

- Die Person erzeugt ein RSA-Paar (k_e, k_d).
- Der Schlüssel k_e wird »allgemein bekanntgegeben« und heißt nun *öffentlicher Schlüssel*.
- Nachrichten werden mit dem öffentlichen Schlüssel der Person verschlüsselt.
- Effekt: Nur diese Person kann die Nachricht (mit dem nur ihr bekannten privaten Schlüssel k_d) entschlüsseln.

Praktische Umsetzung: Erzeugen des Schlüsselpaares.

- Zur Erzeugung eines Schlüsselpaares benutzt man ein *geeignetes Programm* wie beispielsweise `openssl`.

(Das Verfahren wird gleich noch etwas komplizierter werden, aber kümmern wir uns erstmal um den einfachen Fall.)

Praktische Umsetzung: E-Mail an Dorothee.

Schritt 1: Wir brauchen Dorothees öffentlichen Schlüssel.

- Um eine Mail zu verschlüsseln, muss das E-Mail-Programm den *öffentlichen Schlüssel der Person kennen, der man eine E-Mail schreiben möchte*.
- Diesen öffentlichen Schlüssel kann einem die Person beispielsweise vorher geschickt haben – damit es einfacher umgesetzt werden kann, gibt es Schlüsselserver (Key-Server), auf die man seinen öffentlichen Schlüssel überträgt, sobald man ein Schlüsselpaar erzeugt hat.
- E-Mail-Programme oder die Schlüsselverwaltung erlauben es, öffentliche Schlüssel zu *importieren* (ein eigener Menüpunkt) und zu *exportieren*.

So sieht der öffentliche Schlüssel von Dorothee aus:

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2.0.22 (GNU/Linux)

mQGiBEp3/3QRBACfsQLWfuk3bqarUZl1jkweoJiYhxvNbts4f9jvHEBzcZ0vcyB6
QTjwtt7snJh/0lZ1m5ceZ+3oRi+ZG/qPdMzPnnBU1jAdDz9byuP3zzxDpQ8sjcdj
H5CbOMbs1XSe4UDCmshPi8x/ixfRD894aBFvuQuGQIE21ghz/oyiJoOR3wCgs55d
OtZquMwMgB7Xs/N6n8GdidcD/35909DBua/3nfjqD+CNxFUzNnjqJyX8wdqtD2YO
5u9aWlUQqt/43X2KOskrGbvvo4TfDogBD/j9eDABSSNCVsJv4DTSSaGaMMoPFLgj
WpflRsqKcmYD1QygRZv0I+r2LE+DgZ03DuBOppSW/Eij8ymDLjdMmY8hHnipUfi1
HGi3A/sFkMRNgc0Le5wtiqwGO9poGnaV2Axbr1K2/Ta5SIHQw1imm+C+F/4+ItUn
28YJ3psqs0SwyDXFBdP6lnxa5n68WuAPtxOFkSNm8adttI3i15l+a07KWbkFCpC
xGZDajzRf1hDHuz5ToDog5tEa7RTfObPb4N/EzPJMqRh/uLuaLQ5RG9yb3RoZWUg
TXVlbGxlciA8RG9yb3RoZWUuTXVlbGxlckBtYXRoLnVuaS13dXBwZXJ0YWwuZGU+
iGAEExECACACGyMGCwkIBwMCBBUCCAMEFgIDAQIeAQIXgAUCU9EcSgAKCRDgqkIB
AQOMlduzAKCV4m0w+X2vowoOvjTmP/NP2/4sEwCePRGrKElqNJx7pgKP05Szf8Ow
pKiIRgQTEQIABgUCS5+bNwAKCRD8ie760d3KINRwAJ9mMF+ExBh6jkio+IunP1zO
S8xr5QCghtOHDGD2z1R+zbiDlBgYmjOeUfKJARwEEAECAAYFAk96OfcACgkQvDb+
5lv2hI6CswgAqNZDWRUwagjxo5Kfkd8aKy7hEgBKonyoU99bvoltkl2tHZs5b1RL
OtrzC8Voj6MOqNOyQpb4IWpALPDnzzIZR1PF+W4S643W3nAtMQ4S4E89/ED4tlvI
8a17yYKn9Bm/wKl2hnc8zu19+Y9pnkqsbQzVlOACaf+rmUHLmRW5ZhVIZWAE3Fxc
gkfV/W5Lo/6PpooSVF+oejlmn28+WFn5aahrdMXrrWOetrJ2xSA4yzgRN2ah+8wQ
OjYQZ2xyNnek+Gwgw9lTaolbPFhmhbR7fu1I62It21AjZisqZ1ZnSEexzY8tWY7t
fEv9Hs1kxBuIdqFTwNnbA8gTDQnmhIuq7IhGBBARAgAGBQJSG+HvAAoJECYh0iJT
N8v2LDAAoJIpYx2CmobpOwJu3FTc3iMBJuKtAJ9d7bg83fzomQgyg+3qOhMWiaQR
AohgBBMRAgAgAhsjBgsJCAcDAgQVAggDBBYCAwECHgECF4AFAlPVdcEACgkQ4KpC
AQEDjJXawACfYeAwbkP464OHXxIUAI6SeEZMW3YAn06FvJUBNbr0Z3mnEdg43BcG
ftshiGYEExECACYFAkp3/3QCGyMFCQlmAYAGCwkIBwMCBBUCCAMEFgIDAQIeAQIX
gAAKCRDgqkIBAQOMlZTOAKCyIzVBuLzP+vie8ooryqONwoEaeQCeIwCLmSvqczto
hCvTX73jXrkUeC2IZgQTEQIAJgIbIwYLCQgHAwIEFQIIAwQWAgMBAh4BAheABQJT
3zsdBQkUrnCpAAoJEOCqQgEBA4yVkKcAoIPByGeiTVSP9zOTRH4AlrznSnPFAJ9C
ZtHOMls7NaqIcQA95ViHQ8Pp4rkCDQRKd/90EAgA+wKxJLzBjABgz3GPngBI85Su
elCijdBEhK644EfbPETDzVyUiOJczsbnvUR7OllLg/5P7UJd31mSfSwIPKeyp3Gvv
Sn0IfSVqz4qVAepWmHp9YR//3veTHz5qF1dbCNeGoWZ6Mnn/aZCmk6YWAI5mEPHi
QADBGhmxzowLOrb92BKCwqG3Kjnq8p7h9H+94LcsfMUa2ouE/p8SZpAJa+J0gXrz
DhZNp21BXwlmYyHiJxA2DVL2Vp2njeqBnAD8N1aK+q47UI5da9RXpOjT8TiOUpm
hOXfOh/LRYOQrEA1RpVycF5dycJb04xseHz5EceuAu4724O3NMQjPoWKJPkz5wAD
BQgA7+V+WF+5ygwRm1OOnyrrr/nfRrbGCZWyRbLACSh+L+ivqkuuGw1jhHxYqR5V
kyfc0K9J6onBze1fz3jUY2VAhJ6TJGTgWjuPLHhidTESAxbseaPDusZHSiJb5UHl
y74h4uFkv/m+EPrMJdk07EFyvvPxTxELWhx8DO5AukHEFBjge5ibgztaDEpZes7X
FW3qi1U7vFEtdGf+d7fs+30OBbK8x4yFFoL6zo25h4mcaOXVbhKOQtepKwbHUf2+
kdjY78XI8UrZNs3isdtI+slQjFI28pk9rbXU5w7nvjxkaCk7hraaFrtPODTmxrrA
S38FQH2w6FctOwZv3K7ssBcfcIhJBBgRAgAJAhsMBQJT0RxaAAoJEOCqQgEBA4yV
```

```
rykAnjP/SfO2wMGuCLIxKoUGixzQj6xJAKCbZ+uqtOW2DnMq5W16XT/7wo2g/IhJ
BBgRAgAJAhsMBQJT1XXBAAoJEOCqQgEBA4yVKjAAnj56wUTqD0tq5ZecH6EJpZYt
ifaTAKCc9AsxhiK/UrYI8CP2ceXswF7+44hPBBgRAgAPAhsMBQJT3zsnBQkUrnCz
AAoJEOCqQgEBA4yV9k4An3lRCd0lvxeOypW4oUA+gqtsb7THAJ0aCP85dmlXqBgE
FgsjP8fkrZrfzg==
=bCjs
-----END PGP PUBLIC KEY BLOCK-----
```

- Er kann von http://pgp.mit.edu/ mit

  http://pgp.mit.edu/pks/lookup?op=vindex&search=0xE0AA420101038C95 geprüft und heruntergeladen werden.
- Alternativ kann Dorothee den Schlüssel zunächst von ihrem Mail-Programm oder von der Schlüsselverwaltung *exportieren* und dann verschicken (zur Not per Post).

Praktische Umsetzung: E-Mail an Dorothee.

Schritt 2: Schreiben und Abschicken der E-Mail.

- Hat man dem eigenen System erstmal Dorothees öffentlichen Schlüssel »beigebracht«, so kann man ihr eine verschlüsselte E-Mail schreiben.
- Bei modernen E-Mail-Programmen muss man dazu einfach auf einen Verschlüsselungsknopf drücken.
- An Dorothee wird dann ein mit dem *öffentlichen Schlüssel von Dorothee verschlüsselter* Text geschickt.
- Um den Text zu entschlüsseln, braucht man den privaten Schlüssel von Dorothee.
- Folglich kann niemand außer Dorothee – *nichtmal der Autor der Mail* – den Text entschlüsseln.

```
From: "Prof. Dr. L. Humbert" <humbert@uni-wuppertal.de>
Date: Mon, 17 Aug 2015 08:32:29 +0200
To: Dorothee.Mueller@math.uni-wuppertal.de
Message-ID: <C0FEF7F1-1C1C-49F3-88E7-BD155A0D2B94@uni-wuppertal.de>

-----BEGIN PGP MESSAGE-----
Version: APG v1.1.1

hQIOA9WZm8EQ/VTkEAf+J+gwO42bJ6QhwgTds6RZo1kHvhcxXER706GWBPo1xvk5
UZFvfLPrMf5b1mbyIx6Kwi63eOU/24nViP/eDqasweyz02oxrb+VqkLTElNI/U/W
T7OMcVY9tp7ezVKGJA96hq6Hodwy+WWMLBQNuXpGYmZsf2boo2rnrX8ayyUZkwgK
zeLEHhxPOw8dguUxQBlwzoHkO6KrLDoCcU/4vRkMoFQ1VrvmqxYBlSKdlkYOuKPo
rytIbpRhs3Ecor/pkf6DFXG6IOo8L8GplNZyfDmD/itEotWIfmj45iG35QVg5G23
t06EFKlssDldpmKXOA0sJL7bigWLIOSaem7y8iMFywgAs9FQfn/Eqb9Cq4dpzeuV
vy9NkEEAZXEaLVinM4uO5T9LmbbKHf/Z55nrJ+pwmzYnCgYZsRO7aeFw1wkmAoGO
r3bUcYQ85XxK0hx/rKBkl2+qnTWJEifVD8AqJAhUPlCbomKZhNaNjCdNTv7GgrPf
Fm3hSHGLVFQa8rEqFmWc4pSRTRdLaNncQUYlHAp+dMQa1F/TpX5L7QMgBd8cJXXG
BCg7XPgWHL2B3geOtPOMlne/fR6QKUgKaFpC6grftjKu+oNdR5q4DIT5EIAxIPdv
RvHosHnNhPEiMNL5chSAMeVzfJVE/HphLP7K87Hz2paI9fHY4StlZQeLfze0rCy+
BoUCDgPSqAFEkqzjOxAH/A2C5iiJJ//U9Uo8MfoDPOXBFb8a7qpRBeYjq1ZGB9gO
loV9SZ3PiBo4YO1II/rolKKwxh9xdXphG+bXijopvGycpSCMPqXmEig7Hf6t8OJG
w9AxP48F9m7RHKRy6XidkzzNjfNul4XdJb57sWPEOx3wg9bZarDuWYdpAk02DF5F
zOLs8BJ3izlxVZkOlxcyKIATVJpfBqaAr9YIWD6QyzFEXX6vAZ4mlesCVmRYxTOL
jZMAMwk2yrKqExdAtPAl69ZBnA1z62+fPjkxBUUeKY6QK8A68u/Oh9MK1QFlYRIC
wGHrX7d8DgBt6H2woMOpH4ltxfEbB27JnMq88/kcX5OH/1AYsNVnSwmomRyTSnVd
2IPQACqYW8uxYblieZkDDbxAUMLuC8bZ/3aBEOh1b7dOhYMCZx+gWkpGGB4C9m05
ZBYtZU4VgK7fiZAPWcWy7iw4GGOI8Repr4HB9IrNyM+GCIdPO+KLPjOVjD3do2Wl
```

```
BLZ2LoZZVuAscLc5PzTJ2gTRlP2cbhfSpOws97zzYhE5VfZJdkbzGBDfl5AZNG4T
2wMBGRVuRFNgVjhBIeR/tnuIcxTsFhcZD4rrDc/OsPSQSrnXtXL/gZ4ySbYORK+i
8RDMVJMOg8ZNh5vOPjCSkdhYngT7budQdHo+s5LdKTghLaoGPi/nABA6zyESjEyr
evXSwRIBeLwV7WAs68CLc5cBUC8rpJCWxFeG9xO/W4pqV8uzTwGMGbcr9EmArODd
2hjP4KCIX1otnsu9vCXwQPIMQPh62BFHH4t/MMy6w/bhZKoNOUEb5mF/a/KQTr5r
jMq5IprVBX+9QqKHkbsAMUyRAZ64GkrQOTjOrUfywlFQPbrAXF5Z2oZPOYbV1bs3
14VgCyGdkmJ861WWt19MGEoZe3xsxDdNODFWSdtottxM08DaxBZC8GbdHzAGEY3Z
5QOYTWXCxEQtiCXNOQzs2nfyxCsKYsUU77WCZOOHzMaS5gCPXqgyo8Qv3qaql2h9
nQ6Jh/iivgcnYRJdeqe9vm/YB2uUTVCvecQaOacRGrUxP27zm7nAk+ZjRUK7Iynk
O6lQxsbr7NN8HnUh5YaU7TyKS8OxPWMZL21Zxb7jOo6ly3Dhz9NbKY3dOuax1a/4
DM161o+tIe4lwvhjMC1vtyaffGChD4IE3q9bpCzsimINp750ZrgMwGVFvjW3E3l0
Jsvv4WyfsxQCnZR7Us7HVBNRdah8p+VbhyIAkgHsp4VzU7LcPkolBlhHpOMahfwS
AjSDc3Lefaf9n71/x+XATyWtng/WUOfMEOc/XknPl9yi39ghHWE8
=Cuf0
-----END PGP MESSAGE-----
```

Zusammenfassung: Wer bekommt welchen Schlüssel?

- Generell gilt: Ihr *privater* Schlüssel ist nur Ihnen bekannt (auch nicht Ihrem Lover!) und möglichst gut auf Ihrem Informatiksystem geschützt.
  - Am sichersten ist es, wenn der Schlüssel auf einer separaten Karte liegt, die ihrerseits durch eine PIN geschützt ist – dies ist im Prinzip mit dem neuen Personalausweis möglich.
  - In der Praxis liegt Ihr privater Schlüssel aber auf Ihrem Informatiksystem.
  - Dort ist er selbst wieder durch ein Master-Passwort (Passphrase) symmetrisch verschlüsselt und damit selbst vor Diebstahl des Informatiksystems oder vor bösartigen Hackern geschützt.
- Generell gilt: *Öffentliche* Schlüssel sind möglichst breit gestreut und überall verfügbar. Sie sollten auf Ihren Systemen die öffentlichen Schlüssel aller Ihnen bekannter Personen haben.

3.2.7 Authentizität: Digitale Unterschriften

Eine Nachricht soll digital unterschrieben werden.

Ziel: Authentizität

Es soll garantiert werden, dass eine Nachricht von einer bestimmten Person stammt. Dies nennt man auch eine *digitale Unterschrift*.

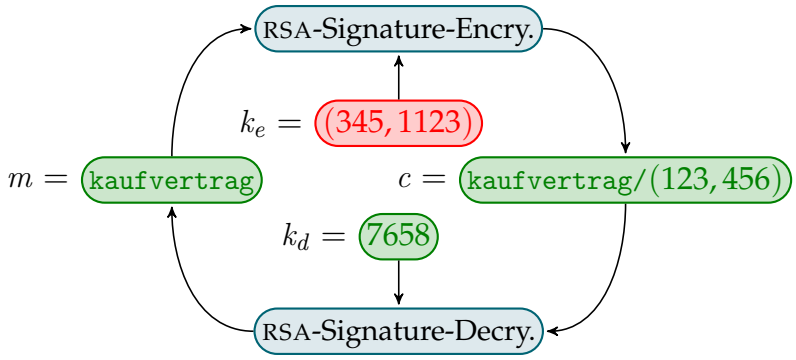
Methode

Man benutzt dieselben Schlüssel wie beim Verschlüsseln von Nachrichten, *nur umgekehrt*:

- Die zu unterschreibende Nachricht wird mit dem *privaten Schlüssel* k_e verschlüsselt und dieser Text an die Originalnachricht angehängt.
- Überprüfung: Man entschlüsselt den verschlüsselten Teil mit dem öffentlichen Schlüssel k_d und vergleicht ihn mit dem behaupteten Text.

- Effekt: Nur die Person, die k_e kennt, kann die unterschriebene Nachricht erzeugen.

Ablauf einer digitalen Unterschrift.



Praktische Umsetzung: Unterschriebene E-Mail an eine Kollegin im ZfsL Hamm.

- Um eine E-Mail zu unterschreiben, braucht man lediglich den eigenen privaten Schlüssel – und den haben wir ja schon erzeugt.
- Damit jemand die E-Mail überprüfen kann, braucht er den öffentlichen Schlüssel – diese sind ja aber frei zugänglich.

Achtung: Mit »Signatur« wird auch manchmal der Standard-Text am Ende einer Mail bezeichnet. Dieser ist *keine* digitale Unterschrift.

```
From: "Prof. Dr. L. Humbert" <humbert@zfsl.ham.nw.schule.de>
To: ...
Subject: ZfsL -- FreiFunk stabil
X-Enigmail-Draft-Status: N1110
Organization: ZfsL - Hamm, Arnsberg - Fachleitung Informatik Gymnasium
 Gesamtschule
Message-ID: <55D0AC87.2030702@zfsl.ham.nw.schule.de>
Date: Sun, 16 Aug 2015 17:30:15 +0200
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:38.0) Gecko/20100101
 Thunderbird/38.2.0
MIME-Version: 1.0
Content-Type: text/plain; charset=utf-8
Content-Transfer-Encoding: 8bit

-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

Sehr geehrte Frau ...

... 34 ausgelassene Zeilen ...
Diese Mail ist digital unterschrieben. Es ist schlicht nicht m=F6glich, =20=

dass irgendwer au=DFer Ludger Humbert diese Mail geschrieben hat.
... 21 ausgelassene Zeilen ...

https://twitter.com/hashtag/PflichtfachInformatik
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v2

iEYEARECAAYFAlXQrIcACgkQOQdtl7v4Qu0EiwCguS75CBflUAoL2ucNwsNaOsQh
```

```
31 OGUAnjqvbVaRQZwS3AP230GbWFHhkSZW
32 =WRek
33 -----END PGP SIGNATURE-----
```

3.2.8 Sicheres Surfen

Sicheres Surfen funktioniert wie sichere E-Mail.

- Wenn Sie mit einem Online-Shop oder Ihrer Bank kommunizieren, haben Sie *dieselben Problem wie bei E-Mail*:
  - Niemand soll die Kommunikation abhören können.
  - Sie müssen sicher sein können, dass Ihre Bank auch wirklich Ihre Bank ist.
- Diese Probleme werden auch genauso gelöst:
  - Die Bank oder der Online-Shop hat ein Schlüsselpaar, das nun aber *keine Person* identifiziert sondern *eine Webseite*.
  - Das Schlüsselpaar ist über eine Kette von Certificate Authorities (CAs) unterschrieben.
- Die Protokolle *https* und *ssh* bauen auf diese Art sichere Kanäle auf.

Zusammenfassung – Kommunikationssicherheit

Kommunikationssicherheit – Zusammenfassung

1. Bei einer *symmetrischen Verschlüsselung* wird derselbe Schlüssel zum Ver- und Entschlüsseln verwendet.
2. Bei einer *asymmetrischen Verschlüsselung* gibt es einen öffentlichen und einen privaten Schlüssel, die *komplementär* zueinander wirken.
3. Die sichere Kommunikation im Internet und bei E-Mail beruht auf *vertraulichen Nachrichten* und *digitalen Unterschriften*.

3.3 Systemsicherheit

Von Viren, Würmern und SQL-Spritzen

Detailkompetenzen

- Bedrohungsszenarien der Sicherheit von Informatiksystemen kennen und einschätzen können
- Schutzmaßnahmen für die Sicherheit von Informatiksystemen kennen und ergreifen können
- Beispiel eines Sicherheitslochs verstehen

In dem Film *War Games* aus dem Jahr 1983 löst der sympathische junge Held beinahe den Dritten Weltkrieg aus, indem er per Modem zufällig Telefonnummern anruft und dabei auf die Nummer des Zentralcomputers des Pentagons trifft. Das Informatiksystem gibt sich als Spieleserver aus. Die wirklich spannenden Spiele wie *Global Thermonuclear War* sind aber durch ein Passwort geschützt. Mit ein paar Nachforschungen kommt der Held an das Passwort (der Name des Sohnes des Programmierers) und kann dann mit dem Informatiksystem eine Partie wagen. Leider setzt das Informatiksystem das Spiel gleich in die Realität um und nur der Held kann das System in eine Endlosschleife schicken, woraufhin es im wahrsten Sinne des Wortes durchbrennt. (Wenn die Schulsysteme jedesmal anfangen würden zu qualmen, wenn Sie mal wieder eine Endlosschleife programmiert haben, wäre die Schule schon längst abgefackelt(Extrabreit 1980).)

Wie sieht die Bedrohungslage 25 Jahre später aus? Sie werden sich nicht mehr mit einem Modem in das Pentagon einwählen können; darf man zumindest hoffen. Jedoch gibt es das moderne Äquivalent zu solchen »Hintertürchen« zu Systemen immer noch. Ein besonders krasses Beispiel aus dem Jahr 2008 werden Sie in diesem Abschnitt des Vorhabens 3 kennen lernen: Die Webseite der Justizvollzugsanstalten des US-Staates Oklahoma. Über deren Webseiten hätte man zwar nicht den Dritten Weltkrieg auslösen, aber zumindest die Justizverwaltung gehörig durcheinander bringen können.

Bei Sicherheit und Informatik[5] geht es aber nicht nur darum, Informatiksysteme möglichst gut abzuschotten. Seit Ende der neunziger Jahre hat ein Prozess begonnen, in dessen Rahmen Menschen immer mehr Persönliches digitalisieren und häufig anderen Menschen zugänglich machen. Dies gilt für digitale soziale Räume wie Facebook ebenso wie für die personalisierte Google-Seite, durch die die Firma Google die komplette Historie Ihrer Suchanfragen protokollieren und auswerten kann. Bei der Sicherheit von Informatiksystemen geht es auch um die Frage, wie hier der Schutz der Daten vor Verlust und unbefugtem Zugriff zu gewährleisten ist. Dieses Problem hat viele Aspekte, unter anderem rechtliche, technische, algorithmische und auch soziale.

[5] Häufig wird diese Eigenschaft auch als *IT-Sicherheit* – bei uns als *Sicherheit von Informatiksystemen* – bezeichnet.

Wiederholung: Worum geht es bei Sicherheit von Informatiksystemen?

Bei der *Sicherheit von Informatiksystemen* geht es um folgende Anliegen:

1. Schutz vor und die Aufrechterhaltung des Betriebs bei
   - Ausfall von Teilen des Systems (Stromausfall, Absturz)
   - Angriffen auf das System (durch Hacker, korrumpierte Mitarbeiter)

   Diese *Systemsicherheit* wird uns in diesem Abschnitt interessieren.
2. Schutz von Daten und Kommunikation vor
   - Spionage
   - Fälschung

   Diese *Daten- und Kommunikationssicherheit* war Thema des letzten Abschnitts.

Wie erreicht man Sicherheit von Informatiksystemen?

Es gibt verschiedene Maßnahmen, um die Sicherheit von Informatiksystemen zu erhöhen. *Perfekte Sicherheit kann es nicht geben.*

Mögliche Maßnahmen sind:

Redundanz Daten liegen mehrfach vor.

Stichwörter: Backups.

Abschottung Es wird schwierig gemacht, in das System hineinzukommen.

Stichwörter: Passwörter und Firewalls.

Aktive Kontrolle Es wird aktiv im laufenden Betrieb überprüft, ob das Systemverhalten normal ist.

Stichwort: Virenchecker, Vier-Augen-Prinzip, Intrusion-Detection

Verschlüsselung Alle Daten werden verschlüsselt. Ohne die Schlüssel sind die Daten nichts wert.

Stichwörter: ssh (secure shell), pgp (pretty good privacy), gpg (gnu privacy guard), https (http secure)

3.3.1 Was ist zu schützen?

Informatiksysteme müssen geschützt werden.

1. Hardware kann ausfallen, was *Datenverlust* und/oder *Produktivitätsverlust* zur Folge hat.
2. Hardware kann sich *bösartig verhalten*:
   - Moderne Informatiksysteme sind Mehr-Prozessor- und Mehr-Benutzer-Systeme.
   - Sie können *selbstständig* mit anderen Informatiksystemen kommunizieren.
   - Dadurch kann ein Informatiksystem *von außen übernommen werden*.
   - Dies bedeutet, dass sich jemand als ein Benutzer ausgibt und dann dem Informatiksystem (bösartige) Befehle erteilt.
   - Ist der Angreifer geschickt, so merkt man davon nichts. Solche Informatiksysteme heißen dann *Zombies*.

Daten sind wichtiger als Hardware.

Neulich auf einem Schild in einem kleinen Laden in Berlin:

> Gestern wurde in diesen Laden eingebrochen und mehrere Informatiksysteme gestohlen. Die Informatiksysteme sind uns egal, aber wir benötigen die Daten auf den Informatiksystemen! Bitte, ihr Diebe, gebt uns die Daten zurück, Diskretion und eine Belohnung garantiert!

- Praktisch alle Daten, die in den Systemen einer Firma lagern, sind schützenswert.
- Ein paar wichtige sind:
  - Kundenkontaktdaten,
  - Forschungsergebnisse,
  - Lagerbestandsdatenbanken oder
  - Buchhaltung.
- Diese Daten müssen nicht nur gegen Diebstahl, sondern auch gegen Verlust durch Feuer, etc. geschützt werden.

Private Daten müssen geschützt werden.

- Zu Ihrer Privatsphäre gehört nicht nur Ihre Wohnung, sondern auch Ihre externen Speichermedien (wie die Festplatte).
- Das Bundesverfassungsgericht hat dies kürzlich sogar in einem Grundsatzurteil zu einem Grundrecht erhoben.
- Sie müssen aber Ihre Grundrechte auch selbst aktiv verteidigen.
- Viele Leute legen ihre Daten *völlig ungeschützt* und *für alle lesbar* ab.

Motto
Zeige mir deinen Web-Browser-Cache und ich sage dir, was für ein Mensch du bist.

3.3.2 Wovor ist zu schützen?

Die größte Sicherheitslücke: Der Mensch.

- Die größte Gefahr für Systeme geht häufig von den *Benutzern* aus.
- In Unternehmen können die *eigenen Mitarbeiter* ihren Zugriff nutzen, um Daten oder gleich die ganze Hardware zu stehlen.
- Menschen können *auf Zettel aufgeschriebene Passwörter* ausspionieren.
- Menschen benutzen oft ganz *leicht zu erratende* Passwörter wie `gott` oder auch `26121975`.

Moral
Der »Faktor Mensch« muss in jedes Sicherheitskonzept einbezogen werden.

Die zweite Sicherheitslücke: Bösartige Programme.

- Damit bösartige Software überhaupt zum Zuge kommt, muss sie erstmal *ausgeführt* werden.
- Normalerweise geschieht dies *nicht freiwillig* durch den Benutzer oder das Betriebssystem.
- Vielmehr nutzt bösartige Software so genannten *Sicherheitslöcher* aus (dazu gleich mehr).

Glossar der Schädlinge.

Wurm Programm, das sich selbstständig über ein Netzwerk ausbreitet, indem es Kopien von sich selbst an andere Informatiksysteme schickt.

Trojaner Programm, das etwas sinnvolles oder hübsches macht, aber eine Schadensroutine enthält (normalerweise das Informatiksystem zum Zombie macht).

Virus Programmteil, der eine Kopie von sich selbst an andere Programme anhängt und immer dann gestartet wird, wenn ein infiziertes Programm gestartet wird.

Wie bekommt man diese Schädlinge?

Leider kein Science-Fiction: Der Wurm der Apokalypse.

Eine Studie der Wurmforscher des ICSI (International Computer Science Institute, Berkeley) ergab 2005 Folgendes:

- Würmer sind in der Regel *extrem schlecht* und schlampig programmiert.
- Ein *sehr gut programmierter Wurm*, der alle bekannten Tricks nutzt und in ein einziges IP-Paket passt, könnte sich in ca. *30 Sekunden weltweit* ausbreiten.
- *In wenigen Minuten* könnte er das Internet komplett lahmlegen.
- Lädt er noch einen Firmware-Flasher nach (sehr schwierig), dann könnte er die weltweite Informatiksysteminfrastruktur *für Wochen lahmlegen*.
- Die Folgen für die Weltwirtschaft könnte man wohl als apokalyptisch bezeichnen.

3.3.3 Welche Maßnahmen helfen?

Gegen Datenverlust helfen nur Sicherungskopien.

Daten müssen regelmäßig gesichert werden. Punkt.

Glossar der Sicherungsmaßnahmen.

Kennwörter Systeme werden zum Schutz in verschiedene *Bereiche* aufgeteilt, zu denen man nur mittels des richtigen Kennworts Zugriff bekommt.

Firewall Programm oder Informatiksystem, das die Verbindung eines Informatiksystems oder eines Teilnetzes zum Internet überwacht. Es lässt nur als *sicher eingestufte* und *vertrauenswürdige* Kommunikation zu.

Virenscanner Programm, das interne und externe Speicher nach den *ihm bekannten* Viren, Würmern oder Trojanern durchsucht.

IDS Intrusion-Detection-Systeme beobachten das Verhalten von (vernetzten) Informatiksystemen. Im Falle von auffälligem Verhalten (beispielsweise massenhafte E-Mails) wird das Informatiksystem gestoppt oder verlangsamt.

3.3.4 Fallbeispiel eines Sicherheitslochs

Die Methode: SQL-Injection

Das Sicherheitsloch »SQL-Injection«.

- SQL-Injection ist ein Methode, Schwachstellen von *Web-Servern* auszunutzen, die auf eine *SQL-Datenbank* zugreifen.
- Die *Angreifer* sind Menschen oder Informatiksysteme.
- Die *Angegriffenen* sind Web-Server.
- Ziel des Angreifers ist es, den Web-Server dazu zu bringen, dass er *SQL-Code des Angreifers ausführt*.
- Man sagt, der Angreifer *»injiziert SQL-Code in den Web-Server«*, daher der Name.

Ablauf einer SQL-Injection.

Normale Kommunikation

1. Nutzer trägt Daten in ein Web-Formular ein
2. Web-Client schickt Formular-Daten an den Web-Server
3. Web-Server führt daraufhin einen SQL-Befehl aus, um Daten aus der Datenbank zu holen
4. Web-Server schickt Antwort an den Web-Client

Kommunikation mit SQL-Injection

1. Nutzer trägt *ungewöhnliche Daten* in ein Web-Formular ein
2. Web-Client schickt Formular-Daten an den Web-Server
3. Web-Server führt SQL-Befehl aus, der aber aufgrund der ungewöhnlichen Daten *ungewöhnliche Effekte* hat.
4. Web-Server schickt *nicht gewollte* Antwort an den Web-Client

Fiktives Beispiel: Firmen-Intranet

Das Intranet von Molecular Sheep.

- Die Firma Molecular Sheep verfügt über ein *Intranet*.
- Dieses bietet Zugang auf firmeninterne Daten (wie die Fellfarben von Dolly und Flauschi).
- Man muss sich *authentifizieren*, um Einlass zu erhalten.
- Die Liste der berechtigten Personen und deren Passwörter ist in einer Datenbanktabelle gespeichert.
- Leider wurde an der Sicherheit gespart, weshalb die Eingangskontrolle schlampig programmiert wurde.

Der Login-Vorgang von Molecular Sheep.

```
<!-- HTML-Login-Seite -->
<form action="http://molecular-sheep.com/login.java" method="post">
  <p>User:     <input name="user" type="text"/> </p>
  <p>Password: <input name="pass" type="text"/> </p>
  <p><input name="submitButton" value="Login" type="submit"/></p>
</form>
```

Wenn sich User `ich` mit dem Passwort `gott` einloggt, wird folgende Anfrage an den Web-Server von Molecular-Sheep geschickt:

```
http://molecular-sheep.com/login.java?user=ich&pass=gott
```

Daraufhin ruft der Web-Server das Programm `login.java` auf mit den Parametern `ich` und `gott` auf. Darin:

```
boolean checkPassword (String user, String password) {
  ...
  String sqlQuery =
    "select␣*␣from␣password_table␣where␣user_name=\"" +
    user + "\"␣and␣password=\"" + password + "\"";

  Statement statement = connection.createStatement();
  statement.executeQuery (sqlQuery);
  if (statement.getMoreResults () == false)
    return false;
  else
    return true;
}
```

Das Sicherheitsloch von Molecular-Sheep.

Login für User `ich` mit Passwort `gott`

Der `sqlString` lautet

```
select * from password_table where
  user_name="#\color{red}ich#" and password="#\color{red}gott#";
```

Dies liefert genau mehr als null Treffen, wenn es den passenden Eintrag in der Tabelle `password_table` gibt.

Login mit SQL-Injection als User mit leerem Passwort

Ein Angreifer tippt nun als »Benutzernamen« folgendes ein:

```
#\color{red}egal\char‘\"\ or true; -{}-#
```

Dann wird folgender SQL-Befehl ausgeführt:

```
select * from password_table where
  user_name="#\color{red}egal\char‘\"\ or true; -{}-#"␣and␣
     password="";
```

Da `--` einen Kommentar in SQL beginnt, liefert dies immer Treffer.

Moral

1. Vertraue *niemals* Benutzereingaben!
2. Benutzereingaben dürfen *niemals* ohne besondere Vorkehrungen mit Befehlen vermischt werden.

Reales Beispiel: www.doc.state.ok.us

Little Shop of Horror für Datenschützer: Webseite des Department of Corrections, Oklahoma.

- Im US-Staat Oklahoma sind (im Jahr 2008) die Insassen *aller Gefängnisse* über das Internet zugreifbar.
- Für *jeden Gefangenen* sind über ein *komfortable Suchfunktion* folgende Daten bequem abrufbar:
  - Name, Geburtsdatum, Rasse (!),
  - Foto(s) des Gefangenen (!!),
  - Komplettes Vorstrafenregister.
- Für ehemalige Sexualstraftäter sind auch *nach der Entlassung (für mindestens 15 Jahre bis lebenslang)* verfügbar:
  - aktuelle Anschrift,
  - Telefonnummer.
- Sexualstraftaten sind dort neben Vergewaltigung auch »distribution of obscene videos« (= Verkauf von Pornos).
- Sehr liebevoll gemacht ist auch die Seite mit dem *Hinrichtungs-Countdown* für die Menschen im Todestrakt.

(Mit dem Betreiben einer solchen Seite würden Sie sich in Deutschland strafbar machen.)

Das Sicherheitsloch des DOC Oklahoma.

- Die Daten zu den (ehemaligen) Gefangenen finden sich in einer relationalen Datenbank.
- Die Suche in diesen Daten wird durch eine SQL-Anfrage bewerkstelligt.
- Das (absolut vollkommen unvorstellbar gigantische) Sicherheitsproblem besteht darin, dass die SQL-Anfrage in einen Link eingebettet ist.
- Dieses Sicherheitsproblem bestand zwischen den Jahren 2005 und 2008.
- Das DOC wurde auf das Problem aufmerksam gemacht, reagierte durch (völlig nutzlose) Änderungen der SQL-Anfrage.
- Das DOC löste das Problem erst, als man ihm die (ebenfalls in der Datenbank gespeicherte) Liste der medizinischen Behandlungen des Personals des DOC schickte.

Der Link im Original, umformatiert.

```
<a href="http://docapp8.doc.state.ok.us/pls/portal30/url/page/sor_roster?sqlString=
    select distinct
      o.offender_id,doc_number,o.social_security_number, o.date_of_birth,
      o.first_name,o.middle_name,o.last_name,o.sir_name,sor_data.getCD(race) race,
      sor_data.getCD(sex) sex,l.address1 address,l.city,l.state stateid,l.zip,
      l.county,sor_data.getCD(l.state) state,l.country countryid,
      sor_data.getCD(l.country) country,decode(habitual,'Y','habitual','')
      habitual,decode(aggravated,'Y','aggravated','') aggravated,
      l.status,x.status,x.registration_date,x.end_registration_date,l.jurisdiction
    from registration_offender_xref x, sor_last_locn_v lastLocn, sor_offender o,
      sor_location l, (select distinct offender_id
                       from sor_location
                       where status = 'Verified' and upper(zip) = '73064' ) h
    where lastLocn.offender_id(%2B) = o.offender_id and
          l.location_id(%2B) = lastLocn.location_id and
          x.offender_id = o.offender_id and
          x.status not in ('Merged') and x.REG_TYPE_ID = 1 and
          nvl(x.admin_validated,to_date(1,'J')) >= nvl(x.entry_date,to_date(1,'J'))
          and x.status = 'Active' and x.status <> 'Deleted' and
          h.offender_id = o.offender_id
    order by o.last_name,o.first_name,o.middle_name&sr=yes">
Print Friendly </a>
```

Das Sicherheitsloch (eher Sicherheitsabgrund) dieser Webseite lässt sich ausnutzen, indem Sie einfach eine eigene Web-Seite erstellen, auf der Sie den Link kopieren, dann aber den Link-Text an entscheidenden Stellen ändern.

Was müssen Sie ändern, um

1. die Liste der Gefangenen zu bekommen, die eigentlich von der Liste gestrichen sind?
2. herauszubekommen, welche anderen interessanten Tabellen lesbar sind?
3. Gefangene aus der Datenbank zu löschen?
4. fiktive Gefangen in die Datenbank einzutragen?
5. dem Spuk ein Ende zu bereiten und die ganze Datenbank zu löschen?

Zusammenfassung – System-Sicherheit

System-Sicherheit – Zusammenfassung

1. *Daten, Kommunikation und Informatiksysteme* sind vielfältigen Gefahren ausgesetzt, die hauptsächlich von (unvorsichtigen, bösartigen oder dummen) *Menschen* ausgehen sowie von *Würmern*.
2. Gute Passwörter, regelmäßiges Schließen von Sicherheitslöchern und die Benutzung von Verschlüsselung bieten *guten – aber längst nicht perfekten – Schutz*.
3. Seine Daten zu schützen sollte genauso selbstverständlich sein, wie das Abschließen der Wohnungstür oder des Fahrrades.

4. Der Staat Oklahoma ist allerdings der Meinung, dass all dies für ihn nicht gilt.

- Kernideen des Urheberrechts kennen
- Konzept der Informationellen Selbstbestimmung verstehen

Wenn man ein Buch über Volkswirtschaft aufschlägt, so geht es in der Regel mit einer kleinen Erörterung los, wie die Wirtschaft eines Landes (oder gleich der ganzen Welt) aufgebaut sein sollte. Da werden dann Alternativen wie Planwirtschaft oder Marktwirtschaft erörtert und – zumindest in westlichen Lehrbüchern – recht schnell geschlossen, dass allein die Marktwirtschaft selig machend ist. Das Argument ist, grob gesprochen, dass allein die Marktwirtschaft in der Lage sei, über das Instrument des Marktes Angebot und Nachfrage effizient zur Deckung zu bringen. Die ganze Daseinsberechtigung der Marktwirtschaft fußt darauf, dass ein knappes Gut (sei es Land, Wasser, Öl oder Quietscheenten) an denjenigen gegeben werden sollte, der bereit ist, den höchsten Preis zu zahlen, da er oder sie es besonders dringend braucht.

Interessant ist nun, dass bei *geistigen* Produkten diese Logik eigentlich gar nicht greift. Ideen sind gerade *keinen* knappen Güter. Es kostet *fast nichts* eine Software zu kopieren. In einer reinen Marktwirtschaft könnte man deshalb auch mit dem Schreiben von Büchern oder Software, mit Patenten oder mit dem Komponieren von Musik kein Geld verdienen (solange es sich nicht um Auftragsarbeiten handelt). Weder Herr Spielberg noch Frau Rowling wären steinreich, Avatar würde kein Geld einspielen.

Es bedarf eines besonderen Konstrukts, um geistige Produkte überhaupt einer Marktlogik zugänglich zu machen: geistiges Eigentum. Hierdurch wird ein reichlich vorhandenes Gut künstlich verknappt – was eigentlich der Grundidee der Marktwirtschaft widerspricht. Es hätte jedoch recht radikale Auswirkungen, würde man dieses Konstrukt wieder abschaffen:

- Musikstücke wären grundsätzlich frei kopierbar. Komponisten würden kaum noch Geld verdienen, Pop-Stars nur noch über ihre Live-Auftritte.
- Bücher wären frei kopierbar, der Autor eines normalen Buches würde zwar noch etwas daran verdienen, Bestseller würden jedoch dem Autor nichts bringen (außer vielleicht Ruhm und Ehre).
- Software wäre frei kopierbar. Software-Firmen würden nur noch mit Auftragsarbeiten Geld machen; »Standard-Software« wie Windows ließe sich nicht mehr verkaufen.

- Patente gäbe es nicht. Damit könnten technische Produkte wie ein Mercedes oder ein iPhone exakt nachgebaut werden. Die privatwirtschaftliche Arzneimittelforschung käme zum Erliegen, da sich mit neuen Medikamenten nicht mehr Geld verdienen ließe als mit den alten.

Ob man in einer solchen Welt würde leben wollen, möge jeder selbst entscheiden.

3.4 Urheberrecht – Fragen, Antworten

3.4.1 Was ist durch das Urheberrecht geschützt?

Grundlage des Urheberrechts: Geistiges Eigentum

- Ist man *Eigentümer* einer *dinglichen Sache*, so kann man entscheiden, was mit der Sache passieren darf.

  Beispiel: »Du darfst meine Puppe nicht benutzen!«

  Beispiel: »Du darfst auf meiner Yacht schlafen.«

- Ist man *Eigentümer* einer *geistigen Sache*, so kann man entscheiden, was mit der Sache passieren darf.

  Beispiel: »Du darfst mein Foto nicht auf deine Webseite stellen!«

  Beispiel: »Du darfst mein Programm nicht kopieren.«

Wieso gibt es eigentlich Eigentum? Und wieso gibt es geistiges Eigentum?

Einige Gegenspieler des geistigen Eigentums

Schon das normale Eigentumsrecht gilt nicht absolut (Artikel 14 des Grundgesetzes). Geistiges Eigentum unterliegt noch mehr Beschränkungen:

- Recht auf Zitate: Man darf zitieren, selbst wenn der Eigentümer dies nicht will.
- Recht der privaten Kopie: Man darf (für sich) Kopien anlegen, selbst wenn der Eigentümer dies nicht will.
- Recht des Künstlers: Man darf parodieren, selbst wenn der Eigentümer dies nicht will.
- Recht des Forschers: Man darf forschen an Ideen und Patente nutzen, selbst wenn der Eigentümer dies nicht will.

Das Urheberrecht schützt Werke

- Immer, wenn ein Mensch eine »persönliche geistige Schöpfungen« erstellt, so ist er oder sie *Urheber* des Werks. Der Begriff ist dabei weit gefasst.
- Das Urheberrecht hat man automatisch und bis zum Tod, wenn man ein Werk erstellt hat – ein »Copyright-Vermerk« ist nicht nötig und hat keine Relevanz. Man behält es selbst dann, wenn man das Werk verkauft,
- jedoch kann man anderen Personen die *Nutzung* eines Werks erlauben:
  - Hier gibt es viele Spielarten (von »alle dürfen mein Bild benutzen« über »wenn du mir 100 Euro gibst, dann darfst du dir mein Werk anhören« bis zu »niemand darf das jemals sehen«).
  - Die verschiedenen Möglichkeiten sind das eigentlich Wichtige am Urheberrecht.

Urheber welcher Werke sind Sie persönlich?

Zentrale Regeln des Urheberrechts.

Merke

- Wenn Sie ein Werk erstellt haben, so darf dies erstmal niemand anderes nutzen, wenn Sie dies nicht explizit oder implizit erlaubt haben.
- Wenn Sie das Werk von jemand anderem nutzen wollen, so müssen sie explizit oder implizit eine Erlaubnis bekommen.

3.4.2 Darf ich Musik von einem Freund kopieren?

Wie legal sind Tauschbörsen?

- Musikstücke sind natürlich geistige Werke und damit geschützt.
- Folglich dürfen Sie beispielsweise ein digitales Musikstück nur nutzen, wenn Ihnen der Urheber dazu das Recht eingeräumt hat.
- Bei einer Tauschbörse ist dies typischerweise nicht der Fall...

Ein Freund bietet Ihnen an, dass Sie seinen Lieblingssong kopieren können.

1. Dürfen Sie das?
2. Was ist, wenn er Ihnen anbietet, seine gesamte Musiksammlung zu kopieren?

Rechtliche Konsequenzen eines Urheberrechtsverstoßes.

Wenn Sie gegen das Urheberrecht verstoßen, müssen Sie mit zwei Konsequenzen rechnen:

1. Der Verstoß ist strafbar. Es kann also ein Strafverfahren gegen Sie angestrengt werden.
2. Der Geschädigte hat einen Schadensersatzanspruch. Er kann von Ihnen das Geld verlangen, das ihm durch Ihren Verstoß entgangen ist.

Sie laden einen Song von einer illegalen Tauschbörse.

- Welche strafrechtlichen Konsequenzen würde dies haben?
- Wie viel Schadensersatz würden Sie leisten müssen?

Sie bieten den Song dann in einer anderen Tauschbörse an.

- Welche strafrechtlichen Konsequenzen würde dies haben?
- Wie viel Schadensersatz würden Sie leisten müssen?

3.4.3 Welche Bilder darf ich auf meiner Webseite benutzen?

Darf ich ein Bild von Brad Pitt auf meine Webseite stellen?

- Fotos und Videos sind genauso geschützt wie Musikstücke.
- Das Urheberrecht hat der Fotograph, nicht die Person, die fotografiert oder gefilmt wurden.
- Die fotografierte Person können sich aber aufgrund ihrer informationellen Selbstbestimmung dagegen wehren, dass ihr Foto verwendet wird.

Sie finden auf bradpittfan.com ein Bild von Brad Pitt. Dürfen Sie dieses auf Ihrer Seite nutzen? Falls nein, welche Konsequenzen könnte das haben?

Sie scannen ein Foto von Brad Pitt aus der Bravo. Dürfen Sie dies auf Ihrer Webseite nutzen?

3.4.4 »Freie« Lizenzen

Was sind »Freie Lizenzen«?

- Oftmals möchte man, dass ein urheberrechtlich geschütztes Werk frei zugänglich sein soll. Dies kann viele Gründe haben:
  - Sie sind der Meinung, das ganze Konzept des geistigen Eigentums sei Blödsinn. »Niemand kann eine Idee besitzen.«
  - Sie sind einfach ein guter Mensch und wollen etwas Gutes für die Menschheit tun.
- Wenn Sie einfach allen Menschen das uneingeschränkte Nutzungsrecht einräumen, dann kann aber jemand anderes Ihr Werk nehmen, leicht verändern und dann »unfrei« machen.
- Dies können Sie dadurch verhindern, dass Sie allen Menschen folgendes Nutzungsrecht einräumen:

  > Du kannst mit meinem Werk machen, was du willst. Wenn du es aber veränderst und/oder weitergibst, dann musst du für das neue Werk auch genau dieses Nutzungsrecht allen Leuten einräumen.

- Dies nennt man auch manchmal ein »Copyleft« (statt Copyright);
- die Details werden in *freien Lizenzen* geregelt.

Fallbeispiel: Diese Skripte

Die Bilder in unseren Skripten unterliegen alle (bis auf eines) freien Lizenzen, klein angedeutet unter den Bildern.

- Welche der *Bilder* und welche der *Texte* aus diesem Skript können Sie selbst auf Ihre Webseite packen oder in Ihre Bachelor-Arbeit übernehmen?
- Müssen Sie besondere Vorkehrungen treffen?
- Wen müssen Sie um Erlaubnis fragen?
- Finden Sie das Bild, das illegal im Skript ist.

3.5 Informationelle Selbstbestimmung

3.5.1 Die Idee

Das Konzept der »Informationellen Selbstbestimmung«.

- Unter »Informationeller Selbstbestimmung« versteht man das Recht einer Person darauf zu entscheiden, was mit Daten über sie geschehen darf.
- Anders als bei geistigem Eigentum geht es nicht um Werke *von* der Person, sondern um Information *über* die Person.

Beispiel: Klausurnoten
Welche Note Sie in einer Klausur geschrieben haben, ist eine Information über Sie. Diese Information ist zwar nicht Ihr »geistiges Eigentum«, trotzdem können Sie verlangen, dass die Note nicht öffentlich gemacht wird.

Beispiel: Kontostand
Ihr Kontostand ist ebenfalls nicht Ihr »geistiges Eigentum«, trotzdem können Sie verlangen, dass Ihre Bank niemandem Ihren Kontostand mitteilt.

3.5.2 Was man über eine Person speichern darf

Das Gebot der Datensparsamkeit

- Trotz Ihrer informationellen Selbstbestimmung brauchen die Uni oder Ihre Bank oder ein Internet-Händler natürlich Daten über Sie.
- Deshalb ist es Ämtern und Firmen auch gestattet, Daten über Sie zu sammeln und zu verarbeiten.
- Jedoch darf nur gesammelt werden, *was wirklich notwendig ist* – es sei denn, Sie erlauben mehr.
- Die Daten dürfen nicht *weitergegeben werden* – es sei denn, Sie erlauben es –
- und müssen *gelöscht werden*, wenn sie nicht mehr gebraucht werden.

Die schufa- und die Werbe-Klausel.

- Bei Kreditverträgen oder bei manchen Mietverträgen will die Bank oder der Vermieter eine SCHUFA-*Auskunft* über Sie.
- Jedoch kann die SCHUFA keine Daten über Sie herausrücken ohne Ihre Einwilligung. Umgekehrt kann die Bank oder der Vermieter der SCHUFA auch keine Daten über Sie vermitteln.
- Deshalb müssen Sie unterschreiben, dass die Bank, der Vermieter und die SCHUFA das explizit dürfen.
- Bei »Werbe-Klauseln« liegen die Dinge ähnlich: Sie müssen explizit erlauben, dass Ihre Daten an Werbefirmen weitergereicht werden.

Aus einem Gewinnspiel:

> Ich stimme der telefonischen Unterbreitung interessanter Angebote über Zeitschriftenabonnements durch die Axel Springer AG und der IBIG GmbH sowie der Verwendung meiner personenbezogenen Daten zur Abwicklung des Gewinnspiels zu. Mir ist bekannt, dass die Teilnahme am Gewinnspiel unabhängig von dieser Einwilligung ist.

3.5.3 Ausblick: Scoring

Das Problem des Geo-Scorings

- Das Recht auf Informationelle Selbstbestimmung betrifft erstmal nur Personen.
- Spannend wird es, wenn Daten über *Orte in der Umgebung eines Menschen* gesammelt werden.
- Aus diesen Daten lässt sich eine *Bewertung (Scoring)* Ihrer Nachbarschaft ermitteln – in Bezug auf viele unterschiedliche Indikatoren.
- Dies könnte auf Dauer dazu führen, dass es zu einer stärkeren sozialen Segregation kommt.

Beispiel
Wer es sich leisten kann, der zieht nicht an Orte mit, sagen wir, hoher Kriminalitätsrate und geringer Kaufkraft.

Es gibt noch keinen gesellschaftlichen Konsens darüber, was beim Geo-Scoring erlaubt sein sollte und was nicht.

Zusammenfassung – Datenschutz

Datenschutz – Zusammenfassung

1. Das Gesetz schützt, in bestimmten Grenzen, geistiges Eigentum.
2. Das Kopieren und Nutzen geistigen Eigentums wird durch Lizenzen geregelt – selbst bei »freier Software« oder »freien Medien«.
3. Sie haben ein Recht auf Informationelle Selbstbestimmung – nutzen Sie es!

Anhang A

Arbeits- und Informationsblätter, Lernzielkontrollen

A.1 Vorhaben Q2-1

A.1.1 Übungen, die vom Automatengraph ausgehen

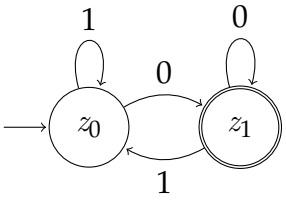
Aufgabe: Für jeden der beiden unten angegebenen Automatengraphen sind anzugeben/-zu erstellen:

a) die charakteristischen Mengen, die den Automaten kennzeichnen

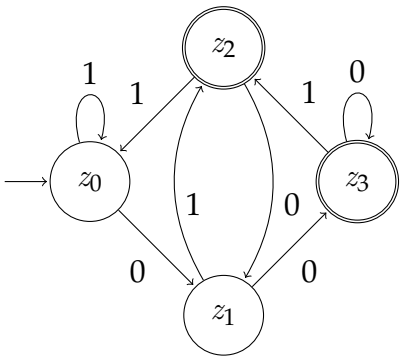
b) die akzeptierte Sprache

c) die Zustandsübergangstabelle

d) ein Programm, das diesen Automaten implementiert



deterministischer Automat – (nach Richter 2005, S. 50 – oben)



deterministischer Automat – (nach Richter 2005, S. 50 – unten)

A.1.2 Endliche Automaten – Lernzielkontrolle

1. Aufgabe (7 Punkte) endlicher Automat

Bei dem durch die Zustandsübergangstabelle angegebenen Automaten ist z_0 der Anfangszustand, z_1 ist der einzige Endzustand.

Zustand/Folgezustand	z_0	z_1
z_0	1	0
z_1	1	0

Beachten Sie, dass Sie die Reihenfolge der Teilaufgaben so wählen, dass Sie die Teile, die Sie schnell bearbeiten können zunächst erledigen.

a) Geben Sie die fünf charakteristischen Mengen und Elemente des Automaten an.

b) Erstellen Sie zugehörigen Automatengraphen.

c) Programm: erstellen Sie eine Automatenklasse, die diesen Automaten implementiert.

d) Geben Sie an, welche Sprache der Automat akzeptiert.

Der zugehörige Automatengraph findet sich als erste Abbildung in A.1.1

A.2 Vorhaben Q2-2

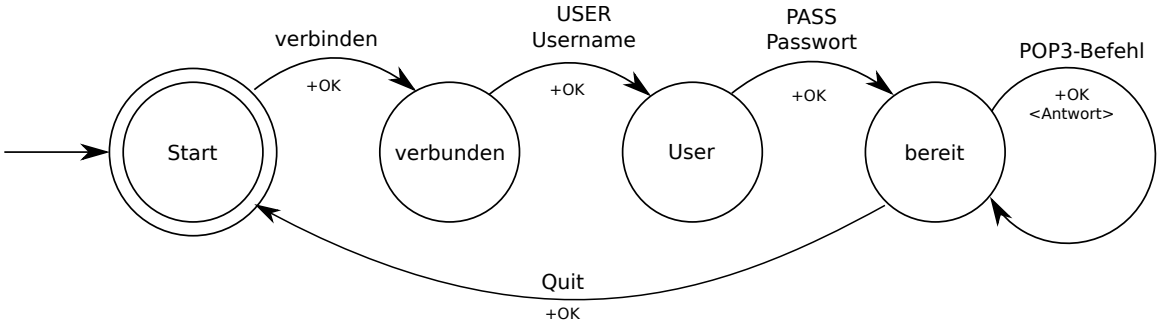
A.2.1 Kommunikation – Netzwerke

POP3-Protokoll

In der RFC 1939 wird das *Post Office Protocol Version 3* (POP3) beschrieben. Mit ihm lassen sich E-Mails aus einem Postfach beim Provider abholen. POP3 benutzt den TCP-Port 110.

Verbindet sich ein POP3-Client mit dem POP3-Server, so sendet dieser als Begrüßung `+OK`. Der Benutzer meldet sich dann mit `USER` Name und `PASS` Passwort an.

Die einzelnen Schritte des Verbindungsaufbaus lassen sich gut mit einem Zustandsdiagramm darstellen:



Nach erfolgreicher Anmeldung kann der Benutzer mit den folgenden POP3-Befehlen arbeiten.

`STAT`	liefert die Anzahl der verfügbaren E-Mails und deren Gesamtgröße
`LIST`	liefert eine numerierte Liste der verfügbaren E-Mails samt Größe
`RETR #`	holt E-Mail mit der Nummer #
`DELE #`	markiert die E-Mail mit der Nummer # zum Löschen
`NOOP`	No Operation
`RSET`	alle Löschmarkierungen werden aufgehoben
`QUIT`	Verbindung beenden und markierte E-Mails löschen

Zwischen Befehlswort und Parameter steht genau ein Leerzeichen, der ganze Befehl wird mit Enter beendet. Jede Antwort des POP3-Servers beginnt mit `+OK` oder `ERR`, worauf weitere Informationen bis zum Ende eine Zeile folgen können. Beim `LIST` und `RETR` Kommando gibt es mehrzeilige Antworten. Das Ende einer Antwort wird durch eine Zeile, die nur einen Punkt enthält, signalisiert.

Anmerkung: Im POP3 Protokoll ist vermerkt, dass die zum Löschen markierten Mails erst mit dem Schließen der Verbindung gelöscht werden. Außerdem dient der Befehl `NOOP` dazu, die Verbindung aufrecht zu erhalten, so dass es zu keinem Timeout kommt.

Aufgabe:

1. Erstellen Sie ein Sequenzdiagramm, indem die Anmeldung an einen POP3-Server, das Holen der ersten Nachricht und das Schließen der Verbindung dargestellt wird.
2. Nutzen Sie in der Textkonsole das Programm Telnet, um sich mit einem POP3-Server zu verbinden. Die genaue Adresse und die Zugangsdaten erfahren Sie von ihrem Lehrer.
3. Führen Sie dort folgende Aktionen aus.
   - Lassen Sie sich die Liste der vorhandenen E-Mails geben.
   - Lesen Sie eine der vorhandenen Mails.
   - Löschen Sie eine der Mails.
4. Überprüfen Sie das Sequenzdiagramm auf seine Korrektheit anhand Ihrer Aktionen mit Telnet.

SMTP-Protokoll

Das SMTP-Protokoll (Simple Mail Transfer Protocol) wird in der RFC 821 und RFC 822 beschrieben. Es beschreibt, wie der Versand von E-Mails zu erfolgen hat. Dabei ist nicht nur der Versand an einen Mail-Server abgedeckt, sondern auch zwischen verschiedenen Mail-Servern. Diese dienen als Zwischenstationen, bis eine Mail das Zielpostfach erreicht hat.

Anmeldung am SMTP-Server

Das SMTP-Protokoll beinhaltet keine Authentifizierungsmechanismen. Daher ist es möglich, dass jeder beliebige Benutzer Mails über jeden SMTP-Server versenden kann. Um dieses zu verhindern, hat man nachträglich verschiedene Sicherheitsmechanismen eingeführt. Bei SMTP-after-POP3 muss man sich zuerst am POP3 Server anmelden. Anschließend hat man für eine kurze Zeitspanne die Möglichkeit, mit der gleichen IP-Adresse eine Mail zu versenden.

Eine neuere Erweiterung des SMTP-Protokolls ist SMTP-Auth (Simple Mail Transfer Protocol Service Extension for Authentication). Bei diesem Verfahren müssen zunächst ID und Passwort entsprechend kodiert an den SMTP-Server übermittelt werden, bevor eine Mail versandt werden kann.

SMTP-Befehle

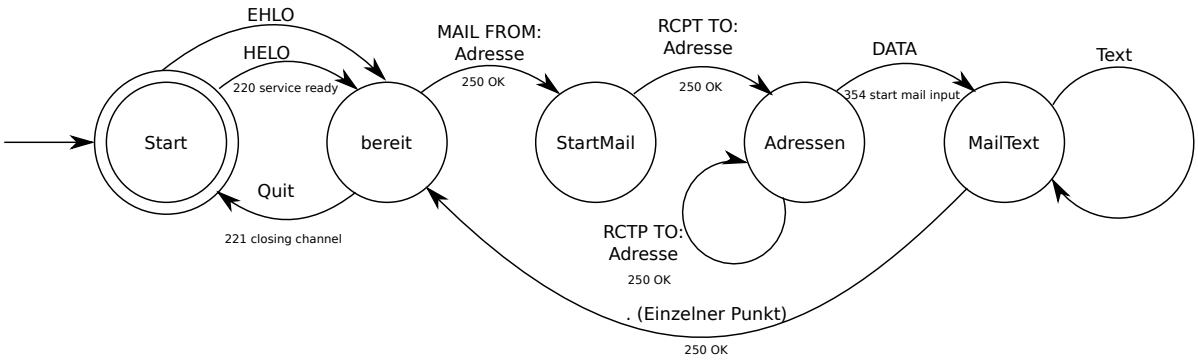
`HELO`	(Hello) Startet die SMTP Sitzung und identifiziert den Client am Server.
`EHLO`	(Extended Hello) Liefert nach dem Start zusätzliche Informationen über den Server zurück.
`MAIL FROM:`	Leitet die Mailübertragung ein und liefert gleich die Absender-Adresse mit.
`RCPT TO:`	Gibt die Adresse eines oder mehrerer Empfänger an; das Kommando kann deshalb mehrmals ausgeführt werden.
`DATA`	Kennzeichnet den Beginn der eigentlichen E-Mail-Nachricht; das Ende wird mit einem einzelnen Punkt in einer Zeile gekennzeichnet.
`RSET`	Bricht eine bereits eingeleitete Mailübertragung ab; die Verbindung zwischen Client und Server bleibt bestehen.
`VRFY`	(Verify) überprüft, ob eine Empfänger-Adresse verfügbar ist.
`NOOP`	Bewirkt eine Antwort vom Server; verhindert die Trennung der Verbindung durch einen Timeout.
`QUIT`	Beendet die Verbindung zum SMTP Server; der Server liefert eine letzte Antwort zurück.

Reihenfolge und Pflichtfelder

Beim Mailversand über SMTP ist zu beachten, dass eine Reihenfolge der Befehle einzuhalten ist. So wird zum Einleiten einer Mail mit `MAIL FROM:` begonnen, es folgt ein oder mehrmals die Angabe der Empfänger mit `RCPT TO:` um anschließend mit `DATA` die Mail einzuleiten. Der Server antwortet dann mit entsprechenden Statuscodes.

Innerhalb der folgenden Datenübertragung kommt zuerst der Kopf einer Mail. Jeweils in einer Zeile sind dazu verschiedene Angaben zu machen. Mit `from:` und `to:` werden der Absender und der oder die Empfänger angegeben. Da diese in der Regel nicht weiter verarbeitet werden, können sie sich von den Angaben zu den Befehlen des Protokolls unterscheiden. Durch `subject:` kann man den Betreff der Mail angeben.

Abgeschlossen wird eine Mail mit einer Zeile, die nur einen Punkt als einziges Zeichen enthält. Damit wird der Datenblock abgeschlossen und die Mail versandt.



Aufgabe:

1. Erstellen Sie ein Sequenzdiagramm, das den Ablauf beschreibt, wenn Sie an eine Freundin eine Mail versenden, die nur aus einer Begrüßung, einem kurzen Text und einer Verabschiedung besteht.
2. Nutzen Sie in der Textkonsole das Programm Telnet, um sich mit einem SMTP-Server zu verbinden. Die genaue Adresse und die Zugangsdaten erfahren Sie von ihrem Lehrer.
3. Geben Sie an, wie mit dem Protokoll die Blindkopie (BCC) realisiert wird.
4. Beschreiben Sie die Auswirkungen des SMTP-Protokolls bzgl. der Anmeldung am SMTP-Server für den heutigen Datenverkehr. Gehen Sie dabei besonders auf Möglichkeiten des Missbrauchs ein.

A.3 Vorhaben Q2-3

A.3.1 Sicherheit – Verschlüsselung

Kryptographie

Wissenschaft, die sich damit beschäftigt, wie Texte (Nachrichten) so verschlüsselt werden können, dass eine nicht authorisierte Entschlüsselung verhindert wird.

Kryptoanalyse

Wissenschaft, die sich mit dem Brechen (Knacken) vorhandener Kryptosysteme beschäftigt.

Kryptologie

Umfasst die beiden Gebiete **Kryptographie** und **Kryptoanalyse**.

Kryptosystem

Definition:

Ein Kryptosystem (oder auch Chiffre) ist ein Quintupel $\mathbf{S} = (M, C, K, \mathscr{E}, \mathscr{D})$

Zur Bedeutung der einzelnen Teile des Kryptosystems

- M, C und K sind endliche Mengen
  - M – Klartextraum (englisch: »message space«)
  - C – Schlüsseltextraum (englisch: »ciphertext space«)
  - K – Schlüsselraum (englisch: »key space«)
- $\mathscr{E} = \{E_k \mid k \in K\}$ ist eine Familie von Funktionen $E_k : M \to C$, die für die Verschlüsselung (englisch: »encrypt«) verwendet werden. $\mathscr{D} = \{D_k \mid k \in K\}$ ist eine Familie von Funktionen $D_k : C \to M$, die für die Entschlüsselung (englisch: »decrypt«) verwendet werden.
- Für jeden Schlüssel $e \in K$ gibt es einen passenden Schlüssel $d \in K$, so dass für jede Nachricht $m \in M$ folgende Gleichung zutrifft: $D_d(E_e(m)) = m$

Ein **symmetrisches** (»private-key«) Kryptosystem erfüllt die Bedingung: $d = e$
D. h. es gibt einen gemeinsamen Schlüssel, der verwendet wird, um die Nachricht zu verschlüsseln und die verschlüsselte Nachricht zu entschlüsseln.

Ein **asymmetrisches** (»public-key«) Kryptosystem erfüllt die Bedingung: $d \neq e$ **und** d praktisch(!) nicht aus e berechnet werden kann. d ist hier der private und e der öffentliche Schlüssel.

Die in diesem Informationsblatt angegebenen Elemente orientieren sich an den Einordnungen in (Rothe 2008, 145f).

Kryptoanalyse – Angriffstypen

- **Ciphertext-only-Angriff:** Die Kryptoanalytikerin kennt nur einige Schlüsseltexte und sie versucht daraus, die Klartexte oder die Schlüssel zu bestimmen. Hält ein Kryptosystem diesem Angriff nicht stand, so sollte es nicht verwendet werden.
- **Known-Plaintext-Angriff:** Die Kryptoanalytikerin kennt nur einige Paare von Schlüsseltexten und zugehörigen Klartexten, aus denen sie die verwendeten Schlüssel ermittelt oder andere Schlüsseltexte entschlüsselt.
- **Chosen-Plaintext-Angriff:** Die Kryptoanalytikerin kann Klartexte nach Belieben auswählen und erfährt die zugehörigen Schlüsseltexte, aus denen sie die Schlüssel bestimmen möchte.
- **Chosen-Ciphertext-Angriff:** Die Kryptoanalytikerin hat Zugang zum Entschlüsselungsgerät und kann einen Schlüsseltext wählen, um den zugehörigen Klartext zu konstruieren.
- **Key-only-Angriff:** Die Kryptoanalytikerin kennt nur den öffentlichen Schlüssel, aber sie hat noch keine Schlüsseltexte abgefangen. Sie versucht nun, den entsprechenden privaten Schlüssel zu bestimmen. Im Unterschied zu den vorgenannten Angriffsarten besteht darin, dass die Angreiferin nun so viel Zeit hat, wie sie möchte, um ihre Berechnungen durchzuführen.

. . . . . .

Man könnte sich fragen, ob vielleicht auch geheim gehalten werden sollte, welches Kryptosystem verwendet wird. Sicherlich könnte es die Aufgabe der Kryptoanalytikerin beträchtlich erschweren, wenn ihr das verwendete Kryptosystem verborgen bleibt (Security by obscurity).

Diese Annahme kann – und dies ist geschichtlich immer wieder bestätigt worden – nicht eingehalten werden. Daher hat man sich in der Kryptologie ein Prinzip zu Eigen gemacht, welches erstmals von dem niederländischen Philologen und Kryptologen Jean Guillaume Hubert Victor François Alexandre Auguste Kerckhoffs von Nieuwenhof (1835 bis 1903) in seinem Buch »La cryptographie militaire« formuliert wurde.

Kerckhoffssches Prinzip

Die Sicherheit eines Kryptosystems darf nicht von der Geheimhaltung des verwendeten Systems abhängen. Stattdessen darf die Sicherheit eines Kryptosystems nur von der Geheimhaltung der verwendeten Schlüssel abhängen.

Die in diesem Informationsblatt angegebenen Elemente orientieren sich an den Einordnungen in (Rothe 2008, 146f).

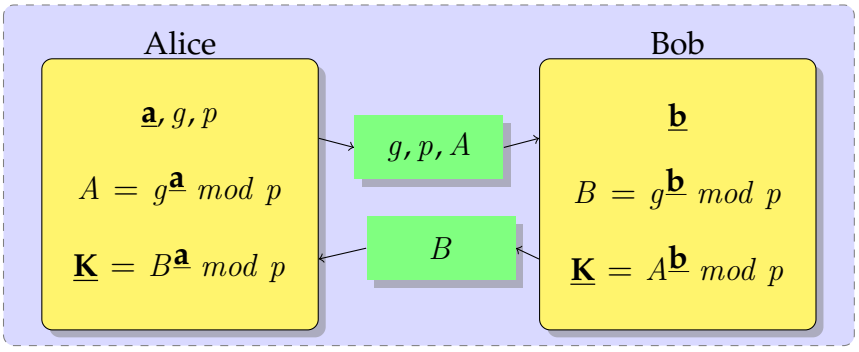
Sicherer Schlüsselaustausch über einen unsicheren Kanal

Wenn ein symmetrisches Verschlüsselungsverfahren verwendet wird, stellen der Schlüssel, seine Übergabe und Geheimhaltung Hauptangriffspunkte für das Verfahren dar. Im zweiten Weltkrieg konnte z. B. der Angriff auf die von der deutschen Wehrmacht benutzte Enigma durch versenkte U-Boote, in denen sich die Codebücher mit Tagesschlüsseln befanden, erfolgreich gestaltet werden.

Daher ließ – vor allem das Militär – an dem Aspekt der Absicherung eines Verfahrens zum Schlüsselaustausch forschen (der in Großbritannien erteilte Auftrag datiert aus den 60er Jahren). Die britischen Forscher James Ellis, Clifford Cocks und Malcolm J. Williamson äußerten Ideen, die allerdings aus Gründen der Geheimhaltung weder veröffentlicht noch patentiert wurden.

Später – nämlich 1976 – wird von den US-Amerikanern Martin Hellman gemeinsam mit Whitfield Diffie und Ralph Merkle an der Stanford-Universität (Kalifornien) ein Verfahren mit der gleichen Grundidee, wie bei den Briten, entwickelt und veröffentlicht. Das Verfahren zum öffentlichen Schlüsseltausch wird – bis heute – als **Diffie-Hellman-Schlüsselaustausch** bezeichnet.

Der prinzipielle Ablauf ist in der nebenstehenden Grafik dargestellt – dabei stellen die beiden Boxen in der Mitte den Teil der Kommunikation zwischen Alice und Bob dar, der beobachtet werden kann, da sie über einen unsicheren Kanal (z. B. das Internet) abgewickelt werden.
Obwohl sich Alice und Bob nie treffen, um einen gemeinsamen Schüssel auszutauschen, gelingt ihnen mit diesem Verfahren die Einigung auf einen gemeinsamen Schlüssel.
Die unterstrichenen Elemente werden nicht weitergegeben.

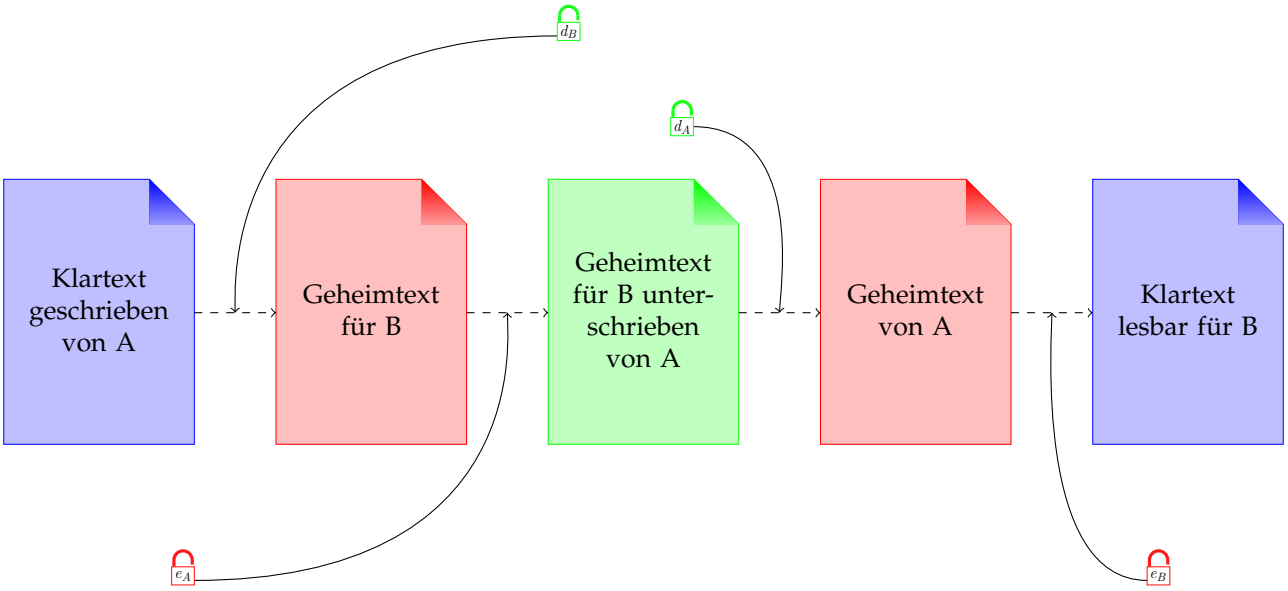


Beispiel zur Illustration – mit kleinen Zahlen

1. Alice schlägt Bob vor: $g = 2$ und $p = 13$.
2. Alice wählt die Zufallszahl $\underline{a} = 5$. Bob wählt die Zufallszahl $\underline{b} = 7$.
3. Alice berechnet $A = 2^5 \mod 13 = 6$ und sendet dieses Ergebnis an Bob.
4. Bob berechnet $B = 2^7 \mod 13 = 11$ und sendet dieses Ergebnis an Alice.
5. Alice berechnet $\underline{K} = 11^5 \mod 13 = 7$.
6. Bob berechnet $\underline{K} = 6^7 \mod 13 = 7$.
7. Beide erhalten das Ergebnis $\underline{K} = 7$.

Signieren einer Nachricht – digitale Unterschrift und Verschlüsselung

Der komplette Ablauf zur Verschlüsselung und zum Signieren (Unterschreiben) einer Nachricht mit anschließender Prüfung der Unterschrift und Entschlüsselung findet sich in der folgenden Darstellung.



Bitte beachten Sie – die Darstellung von Schnappschlössern entspricht nicht der ganzen »Wahrheit«, da die öffentlichen und privaten Schlüssel (d_A, d_B, e_A und e_B) je nach Situation als Schloss oder als Schlüssel verwendet werden.

Literatur

Bongartz, Dirk und Walter Unger (2006). »Public-Key-Kryptographie – Verschlüsseln mit öffentlichen Schlüsseln«. In: *Taschenbuch der Algorithmen*. Hrsg. von Berthold Vöcking u. a. Algorithmus der Woche 27. Berlin, Heidelberg: Springer, S. 157–169. ISBN: 978-3-540-76393-2. URL: https://is.gd/qD1KD6 (besucht am 11. 06. 2016).

Enigma (Maschine) (2008). URL: https://de.wikipedia.org/wiki/Enigma_%28Maschine%29 (besucht am 12. 11. 2015).

Extrabreit (1980). *Hurra, Hurra, die Schule brennt*. Songtext. URL: http://is.gd/PE4g0h (besucht am 10. 12. 2015).

MSWWF, Hrsg. (1999). *Richtlinien und Lehrpläne für die Sekundarstufe II – Gymnasium/Gesamtschule in Nordrhein-Westfalen – Informatik*. 1. Aufl. Schriftenreihe Schule in NRW 4725. MSWWF (Ministerium für Schule und Weiterbildung, Wissenschaft und Forschung des Landes Nordrhein-Westfalen). Frechen: Ritterbach Verlag.

Pieper, Johannes und Dorothee Müller, Hrsg. (2014). *Material für den Informatikunterricht*. Arnsberg, Dortmund, Hamm, Solingen, Wuppertal. URL: http://uni-w.de/1t (besucht am 29. 04. 2016).

Richter, Detlef (2005). »Konzept für einen zeitgemäßen Informatikunterricht am Beispiel einer Unterrichtsreihe ›Automatentheorie‹ in der Sek II«. Hausarbeit gemäß OVP. Hamm: Studienseminar für Lehrämter an Schulen – Seminar für das Lehramt für Gymnasien/Gesamtschulen. URL: https://is.gd/o3rgfM (besucht am 11. 06. 2016).

Rothe, Jörg (2008). *Komplexitätstheorie und Kryptologie. Eine Einführung in Kryptokomplexität*. eXamen.press. Berlin: Springer. ISBN: 978-3-540-79744-9.