Kryptologie im Informatikunterricht der Sekundarstufe I

Arno Pasternak
Fritz-Steinhoff-Gesamtschule Hagen
TU Dortmund

16. Dezember 2009

Inhaltsverzeichnis

1 Einbettung in den Informatik-Unterricht der Sek I **2**
1.1 Voraussetzungen . . . 2
1.2 Ziel der Unterrichtseinheit Kryptologie . . . 3
1.3 Der Computer als Verschlüsselungsautomat . . . 3

2 Die Unterrichtseinheit Kryptologie **4**
2.1 Das Caesar-Verfahren . . . 4
2.2 Das Vigenère-Verfahren . . . 8
2.3 Das Vigenère-Verfahren mit grafischer Oberfläche . . . 10
2.4 Public-Key - Verfahren . . . 13
2.5 Unterrichtliche Untersuchung des public-key-Verfahrens . . . 15
2.6 Signieren von Daten . . . 16

3 Aufbauende Unterrichtseinheiten **16**
3.1 E-Mail . . . 16
3.2 Betriebssyteme und Netze . . . 17
3.3 Online-Banking . . . 17

4 Schlussbemerkungen **18**

Zusammenfassung

Schüler und Schülerinnen wissen heute im Allgemeinen, dass Informationen im Netz nicht sicher sind, sondern eventuell „abgehört“ werden können. Daraus folgt die Forderung, Daten zu verschlüsseln.

Was das bedeutet und welches Verschlüsselungsverfahren für welchen Zweck das Geeignete ist, ist für Schüler und Schülerinnen nicht selbstverständlich klar. Ebenso wird dem Schüler bzw. der Schülerin nicht deutlich, welche Ideen und Ziele sich hinter welchem Verfahren verbirgt, wenn er oder sie in irgendeinem Anwendungsprogrammen irgendwelche „Knöpfe“ drückt.

Erstaunlicherweise ist es nun gar nicht notwendig, dass die Schüler grossartig mathematische Theorien erarbeiten müssen, um die Ideen der Verschlüsselungen zu verstehen. Gerade im Gegenteil: Ohne viel Mathematik kann ein Schüler von Heute lernen, was beim Verschlüsseln mit seinen Daten beim Sender und Empfänger passiert.

In diesem Beitrag wird gezeigt, wie im Informatikunterricht in der Sekundarstufe I wesentliche Verfahren der Kryptographie behandeln werden können.

Dabei werden der Reihe nach die Verfahren

- Caesar
- Vigenère
- Public Key
- Signieren von Daten

thematisiert.

Diese Unterrichtssequenz sollte meines Erachtens eine Pflichteinheit im Unterricht von Schülern und Schülerinnen der Sek I darstellen.

1 Einbettung in den Informatik-Unterricht der Sek I

1.1 Voraussetzungen

Sinnvoll ist es, dass die Schülerinnen und Schüler die Verletzlichkeit von Computersystemen in Netzen (im Gegensatz z.B. zu Telefonnetzen) kennen. Je nach Unterrichtsablauf kann dies beispielsweise durch folgende Vorkenntnisse bekannt sein:

- In einer technisch orientierten Einheit haben die Schüler z.B. an Hand einer Ampelsteuerung erfahren, wie diese durch Computer mit Hilfe eines Interfaces funktionieren. Anschliessend wurden zwei Computer gekoppelt, die ihre Steuerungen mit dieser Kopplung syncronisieren. Dann ist es nur noch ein kleiner Schritt, dass statt Synchronisierungsimpulsen Daten zwischen diesen beiden Computern übertragen werden können. Sind erst zwei Computer vernetzt, ist der Übergang zu einem Netz mit mehreren Computern (entweder praktisch oder theoretisch) leicht möglich. Den Schülern wird dabei schnell klar, dass Informationen in einem Netz nicht nur allein dem Empfänger zugänglich sind.

- Die Schüler haben beispielsweise HTML-Seiten erstellt, die dann per *scp, sft oder ftp* auf einen Web-Server übertragen wurden. Dabei sind grundsätzliche Strukturen der Kommunikation in einem Netz gleichartiger Knoten wie z.B. Client-Server-Strukturen besprochen worden. Ebenso ist den Schülern bekannt, dass die Kommunikation zwischen den beteiligten Knoten eine textbasierte Kommunikation ist[1]. Sie haben sinnvollerweise diese Kommunikation bei einer derartigen Sitzung mit einem textbasierten Clienten durchgeführt. Ebenso ist bekannt, dass der Zugriff auf einen Host über grössere Entfernungen über eine *Terminal- oder eine telnet- oder ssh-Sitzung* stattfinden kann. Die Gefährdung des Abhörens einer ftp- und/oder telnet-Sitzung in einem Netz ist thematisiert worden.

- Aufgrund der fast täglichen Berichte über Kriminalität im Netz kann notfalls auch ohne wesentliche technische Voraussetzungen die notwendigen Informationen über Empfindlichkeit von Kommunikation in Netzen thematisiert und verstanden sein.
  Auch wenn Schülerinnen und Schüler im Allgemeinen noch über keine EC- und Kreditkarte verfügen, kann die Weitergabe von persönlichen Daten beim elektronischen Einkaufen als lebenswirklicher Kontext verwendet werden, um das Thema Kryptologie vorzubereiten.

Im Sinne eines Unterrichtes *Informatik im Kontext*[2] wird aus dieses Szenarien festgelegt, sich mit den Möglichkeiten der Ver- und Entschlüsselung zu beschäftigen.

[1] zur Bedeutung der Nutzung von textbasierter Clienten siehe auch [Pas05]

[2] `http://www.informatik-im-kontext.de`, Zugriff: 27.11.2009

1.2 Ziel der Unterrichtseinheit Kryptologie

Durch den vorhergehenden Unterricht ist wie oben erwähnt den Schülern deutlich geworden ist, dass Verschlüsselung in Computernetzen eine sinnvolle und notwendige Sache ist. In der nun folgenden Sequenz soll erarbeitet werden, welche verschiedenen Techniken zum Verschlüsseln möglich sind. Dabei geht es nicht darum, dass die Schülerinnen und Schüler ein Verfahren als das allein Seligmachende kennenlernen, sondern dass die Auswahl eines geeigneten Verfahrens von der eingesetzten Technik als auch von der Art der Kommnikationspartner sowie des Inhaltes abhängig ist. Neben der aktuellen Bedeutung der Verschlüsselung in Netzen soll dabei auch die historische Entwicklung der Kryptologie nicht zu kurz kommen. Weitergehende Informationen kann man mit den verschiedensten Schwerpunkten in der Literatur beispielsweise unter [ADH+00, S.309ff], [Hro06, S.231ff], [Kip06], [Köh03], [Hei85, S.117] finden.

1.3 Der Computer als Verschlüsselungsautomat

Der Computer als technische Realisierung des universellen Automatens ist natürlich auch ein wunderbarer Ver- und Entschlüsselungsautomat. Generationen von Chiffrierern und Dechiffrierern wären froh gewesen, eine nur annähernd so gute Maschine wie den Computer gehabt zu haben. Die technisch phantastische *Enigma* [Kip06, S.211ff],[Bat96],[Str97, S.58ff] ist ein Nichts gegen den Computer als Chiffriermaschine.
Meines Erachtens ist es für Schüler wichtig, immer wieder die Mechanismen eines solchen Automaten selbst kennen zu lernen und auszuprobieren. Dazu ist Programmierung unablässig. Auch ein Algorithmus zur Ver- bzw. Entschlüsselung ist erst dann voll verstanden, wenn er automatisiert werden kann. Wenn man ihn aber automatisieren, sprich programmieren kann, dann soll man bzw. der Schüler es auch (zumindest des Öfteren) tun. Hromkovič schreibt: “Die historisch wichtigen Begriffe, welche die Informatik zur selbständigen Disziplin gemacht haben, sind die Begriffe **„Algorithmus“** und **“Programm“**. Und wo könnte man die Bedeutung dieser Begriffe besser vermitteln als beim Programmieren?“ [Hro08, S.11].

Die praktische Umsetzung der Algorithmen erfolgt in diesem Beitrag in der Skriptsprache *Tcl/Tk* [Ous95], [HM98], [Web00]. Diese Sprache ist relativ einfach und schnell zu erlernen. Es können damit problemlos Anwendungen sowohl mit textbasierter als auch grafischer Oberfläche erstellt werden. Da die Sprache ursprünglich nicht aus dem unterrichtlichen Umfeld entstammt, sind didaktische Massstäbe beim Entwurf der Sprache nicht Grundlage gewesen. Trotz allem ist sie nicht nur für Schüler eine der wenigen schnell erlernbaren Sprachen, mit denen programmtechnische Erfolge auch über den Unterricht hinaus auf allen heute üblichen Plattformen erzielt werden können.

Mit der hier genutzten Sprache *Tcl/Tk* ist die Einheit allerdings noch nicht durchgeführt worden. In der Fritz-Steinhoff-Gesamtschule Hagen wurde im Unterricht der Sekundarstufe I seit den 80-Jahren die Programmiersprache *COMAL* verwendet. Sie ist eine aus didaktischen Gründen entwickelte Sprache, die Børge Christensen und Benedict Løfstedt aus Dänemark entwickelt haben [Chr85]. Da sie leider nicht weiterentwickelt wurde, enthält sie nicht alle modernen Konstrukte und ist auch nicht für alle Betriebssysteme vorhanden. Diese im Unterricht mehrfach erprobten Programmbeispiele sind für diesen Beitrag daher nach Tcl/Tk übertragen worden. In dieser Form wird die Einheit dann auch in der Zukunft an der Fritz-Steinhoff-Schule unterrichtet werden. [3].

[3] Es versteht sich von selbst, dass sich jede andere Programmiersprache genauso oder nach Standpunkt des Lesers sogar besser für den Unterricht in der Sekundarstufe I im Allgemeinen und für diese Einheit im Speziellen eignet.

2 Die Unterrichtseinheit Kryptologie

Beginn der Erarbeitungsphase

Ver- und Entschlüsselung sind vom Prinzip nicht mit der Anwendung eines Computers verknüpft. Es ist daher sinnvoll, in der Erarbeitungsphase auch auf den Computer zu verzichten, um die vom Computer ausgehenden Einschränkungen und Möglichkeiten nicht als Grundlage der Ver- und Entschlüsselung zu sehen.

Die Schülerinnen und Schüler werden zu Beginn aufgefordert, einen vorgegebenen relativ kurzen Satz wie beispielsweise *Bald beginnt die Fussball-WM* z.B. in Partnerarbeit nach einem selbst zu findenden Verfahren zu verschlüsseln. Die dabei entstehenden codierten Sätze werden an die Tafel geschrieben. Alle Schüler sind nun aufgefordert, die codierten Sätze ihrer Mitschüler zu dechiffrieren.

Es ist erstaunlich, welche unterschiedlichen und teilweise phantasievollen Verfahren dabei auftreten. Es ist nicht selten, dass dabei Verfahren sind, die man niemals mit einem Computer realisieren würde und dadurch einen besonderen Sinn und Reiz erhalten. Bei der Analyse wird dann den Schülerinnen und Schülern oft relativ schnell klar, dass die Verfahren sich in die Klassen

- *Ersetzungsverfahren*, bei denen die Zeichen des Klartextes durch andere Zeichen derselben oder einer anderen Zeichenmenge ersetzt werden, sowie die
- *Tauschverfahren*, bei denen die Reihenfolge der Zeichen des Klartextes verändert wird oder
- eine Kombination dieser beiden Prinzipien

einteilen lassen. Sie stellen auch fest, dass manche ihrer Verfahren zwar eine eindeutige Verschlüsselung zulassen, aber nicht mehr eindeutig entschlüsselt werden können. Eine Verschlüsselung ohne eine Entschlüsselung ist in den meisten Fällen aber nicht erwünscht.

Bei einer weiteren Analyse wird den Schülern deutlich, dass eine maschinelle Bearbeitung –vor allem eine Bearbeitung mit Computern – von Codierungen erheblich vereinfacht wird, wenn man sich auf Buchstaben und/oder Ziffern bzw. Zahlen als Zeichen beschränkt.

2.1 Das Caesar-Verfahren

Es ist naheliegend, dass Militärs schon sehr früh Interesse an der Chiffrierung von Texten hatten. Es ist daher auch nicht sehr verwunderlich, wenn eines der einfachsten Verfahren auf *Caesar* zurückgeführt wird. Man findet in der Literatur vielfach allerdings ohne Quellenangabe die Behauptung, dass sich in den Werken von Caesar eine Beschreibung des Verfahrens enthalten ist. Das ist aber offensichtlich nicht korrekt. Allerdings findet sich beim römischen Schriftsteller *Sueton* eine derartige Beschreibung [Ihm08, Suet.Jul.56.6][4].

Wenn wir dieses Verfahren mit unserem Automaten - dem Computer - durchführen wollen, ist sofort einleuchtend, dass wir als Zeichenvorrat nicht das „normale“ Alphabet, sondern den ASCII-Zeichensatz verwenden. Falls nicht geschehen, bietet es sich hier an, den Schülern und Schülerinnen diesen Zeichensatz und seine Bedeutung (auch in seiner Bedeutung auf das Erstellen von ASCII-Texten z.B. mit einem einfachen Editor und formatierten Texten z.B. mit einer Textverarbeitung) zu vermitteln.

[4] Suet.Jul.56.6:
„[6] epistulae quoque eius ad senatum extant, quas primum uidetur ad paginas et formam memorialis libelli conuertisse, cum antea consules et duces non nisi transuersa charta scriptas mitterent. extant et ad Ciceronem, item ad familiares domesticis de rebus, in quibus, si qua occultius perferenda erant, per notas scripsit, id est sic structo litterarum ordine, ut nullum uerbum effici posset: quae si qui inuestigare et persequi uelit, quartam elementorum litteram, id est D pro A et perinde reliquas commutet.“

Die Programmierung des Verfahrens ist nicht sehr aufwändig. Auch die Schüler, die teilweise beim Programmieren grosse Schwierigkeiten haben, können zumindest diesen Programmteil nachvollziehen und zumeist auch erklären. Betrachten wir nun den Kern des Programms mit den praktisch identischen Prozeduren zur Ver- und Entschlüsselung nach Caesar[5]:

```
1  proc caesar_verschluesseln {} {
2    global satz
3
4    set stelle 0
5    set verschluesselter_satz ""
6    set laenge [string length $satz]
7
8    while {$stelle < $laenge} {
9      set zeichen [string index $satz $stelle]
10     scan $zeichen %c asciizahl
11
12     set asciizahl [expr $asciizahl + 3]
13     set zeichen [format %c $asciizahl]
14     set verschluesselter_satz $verschluesselter_satz$zeichen
15
16     set stelle [expr $stelle +1]
17     }
18   set satz $verschluesselter_satz
19   }
20
21
22 proc caesar_entschluesseln {} {
23   global satz
24
25   set stelle 0
26   set entschluesselter_satz ""
27   set laenge [string length $satz]
28
29   while {$stelle < $laenge} {
30     set zeichen [string index $satz $stelle]
31     scan $zeichen %c asciizahl
32
33     set asciizahl [expr $asciizahl - 3]
34     set zeichen [format %c $asciizahl]
35     set entschluesselter_satz $entschluesselter_satz$zeichen
36
37     set stelle [expr $stelle +1]
38     }
39   set satz $entschluesselter_satz
40   }
```

Für einen Pascal- oder Java-Kundigen Leser ohne Vorerfahrungen mit *Tcl/Tk* mag der Quelltext aufgrund der etwas ungewöhnlichen Syntax von Tcl erst beim zweiten Lesen verständlich sein. Man bedenke jedoch, dass diese Schwierigkeiten kein Schüler hat, der seine „Programmiersprachen-Ersterfahrungen“ mit dieser Sprache erhält.

In COMAL, das kurzgefasst auch als eine Reduzierung von Pascal verstanden werden kann, sehen diese beiden Routinen folgendermassen aus:

[5] In den hier vorgestellten Programmen bzw. Programmauszügen wird mit globalen Variablen wie z.B. `satz` gearbeitet. Es ist meines Erachtens strittig, ob man auch in der Sekundarstufe I bei der prozeduralen Programmierung völlig auf globale Variabeln verzichten soll. Eine objektorientierte Schreibweise ist in Erweiterungen von *Tcl/Tk* möglich.

```
0550 PROC caesar_verschlüsseln
0560   stelle:=1
0570   REPEAT
0580     asciizahl:=ORD(satz$(:stelle:))
0590     asciizahl:=asciizahl+verschlüsselungszahl
0600     satz$(:stelle:):=CHR$(asciizahl)
0610     stelle:=stelle+1
0620   UNTIL stelle>LEN(satz$)
0630 ENDPROC caesar_verschlüsseln
0640 //
0650 //
0660 PROC caesar_entschlüsseln
0670   stelle:=1
0680   REPEAT
0690     asciizahl:=ORD(satz$(:stelle:))
0700     asciizahl:=asciizahl-verschlüsselungszahl
0710     satz$(:stelle:):=CHR$(asciizahl)
0720     stelle:=stelle+1
0730   UNTIL stelle>LEN(satz$)
0740 ENDPROC caesar_entschlüsseln
```

In beiden Sprachen müssen den Schülerinnen und Schüler zusätzliche Informationen zur Umwandlung von *ASCII-Zeichen* in Zahlen sowie String-Behandlung gegeben werden. Dieses stellt natürlich eine Verständigungshürde dar, mit der mit Vorsicht umgegangen werden muss.

Die oben angegebenen Prozeduren sowohl in Tcl/Tk als auch in COMAL enthalten schon die Möglichkeit, die Verschiebung der Buchstaben im verschlüsselten Satz nicht nur um drei, sondern um beliebig viele Positionen durchzuführen. Um die Tauglichkeit dieses Verfahrens zu überprüfen, reicht es nicht aus, dieses zu programmieren und die korrekte Funktionsweise des Programmes bzw. der Prozeduren festzustellen. Wie tauglich dieses Verfahren mit und ohne Computerist, erhalten die Schüler von mir einen vorgegeben Satz wie z.B.

```
T]S[XRW/Xbc/^bcTa]=
```

Selbst wenn wie in diesem Fall den Schülern bekannt ist, dass es sich um die Caesar-Verschlüsselung handelt, dauert es für Ungeübte geraume Zeit, diesen Satz zu entschlüsseln. Umso schwieriger wird es, wenn nur Sender und Empfänger der Nachricht wissen, dass es sich um die Caesarverschlüsselung handelt. Es ist sinnvoll, die Schüler einen Satz ohne Computer-Hilfe entschlüsseln zu lassen. Dadurch wird ihnen bewusst, wie auf der einen Seite selbst einfache Verfahren einen gewissen Schutz bieten und somit in früheren Zeiten ihren Sinn gehabt haben[6]. An dieser Stelle lassen sich trotz aller Kürze Ideen der Kryptoanalyse ansprechen wie z.B. das Suchen nach Buchstabenhäufigkeiten.

Kann nun der Computer als Entschlüsselungsautomaten eingesetzt werden, sieht die Sache schon ganz anders aus. Durch mehr oder weniger systematisches Probieren mit der Verschlüsselungszahl kommt man sehr schnell darauf, dass es sich um den Satz

```
ENDLICH IST OSTERN.
```

handelt. Noch deutlicher wird es, wenn die manuelle Veränderung der Verschlüsselungszahl in einer Wiederholungsanweisung automatisch variiert wird und die Lösung praktisch sofort auf dem Bildschirm steht. Das zuvor zumindest scheinbar teilweise geeignete Verfahren wird sofort in seiner Beschränktheit deutlich.

[6] "Dies wird beispielsweise an folgendem Zitat deutlich: „Dine monoalphabetische Substitution mit einem CAESAR-Chiffrierschritt wurde 1915 in der russischen Armee eingeführt, nachdem es sich herausgestellt hatte, daß man den Stäben etwas Komplizierteres nicht zumuten konnte.“ [Bau00, S.51].

Ein weiteres Problem taucht bei folgendem „Versuch“ auf: Die Schüler sollen sich kurze Mitteilungen (eine Art SMS) zusenden. Diese Mitteilungen werden mit der Tastatur in den Computer eingetippt und als Dateien auf einem gemeinsamen Verzeichnis der Schülergruppe hinterlegt. Dabei wird vereinbart, dass als Dateiname eine Kombination von Empfänger- und Absenderadresse verwendet wird. Z.B. meint N311N323, dass vom Computer N323 eine Nachricht an den Computter N311 gesendet worden ist. Diese Nachrichten sind nun von allen Schülerinnen und Schülern einsehbar und sollen mit dem Caesarverfahren verschlüsselt werden. Die Schüler erhalten von mir ein Programm, in dem ihre Prozeduren zur Ver- und Entschlüsselung enthalten sind und das um alle weiteren gewünschten Prozeduren ergänzt worden ist.

Wir erhalten damit folgendes sich (fast) selbst erklärende Hauptprogramm:

```
...
set pfad "server"

puts "CAESAR-Verschluesselungssystem"
puts "Eigene Adresse (z.B. 312 für R312) eingeben:"
gets stdin adresse
puts ""

# set versatz 0
# (Zeile auskommentiert, damit eine fehlende Eingabe "bemerkt" wird.)

set satz ""
set wahl -1

while {$wahl != 0} {
   puts "============================"
   puts "       Client: R$adresse"
   puts "============================"
   puts "            Menue"
   puts "============================"
   puts ""
   puts "Informationen ausgeben:   i"
   puts ""
   puts "Satz eingeben:            1"
   puts "Satz ausgeben:            2"
   puts "Satz laden   :            3"
   puts "Satz speichern:           4"
   puts "Satz verschluesseln:      5"
   puts "Satz entschluesseln:      6"
   puts "SMS Verzeichnis ausgeben: 7"
   puts "Versatz eingeben:         8"
   puts "SMS loeschen:             9"
   puts "Ende:                     0"
   puts ""
   puts ""
   puts "Eingabe:"
   gets stdin wahl
   puts ""

   if {$wahl == "i"} then {info_ausgeben}

   if {$wahl == 1} then {satz_eingabe}
   if {$wahl == 2} then {satz_ausgabe}
   if {$wahl == 3} then {satz_laden}
```

```
45    if {$wahl == 4} then {satz_speichern}
46    if {$wahl == 5} then {satz_verschluesseln}
47    if {$wahl == 6} then {satz_entschluesseln}
48    if {$wahl == 7} then {SMS_verzeichnis_ausgeben}
49    if {$wahl == 8} then {versatz_eingeben}
50    if {$wahl == 9} then {SMS_loeschen}
51    }
```

Das Arbeiten mit diesem Programm gelingt natürlich nur dann, wenn alle Computer ein gemeinsames Verzeichnis auf einem Server ansprechen können. Der Verzeichnisname wird der Variablen `pfad` zugewiesen, hier im konkreten Beispiel mit `server`. Sofern in der konkreten Schulsituation die Computer nicht auf ein gemeinsames Verzeichnis zugreifen können, kann dieses Programm auch per *telnet* oder *ssh* auf einem entsprechend vorbereiteten Computer von mehreren Personen gleichzeitig ausgeführt werden[7].

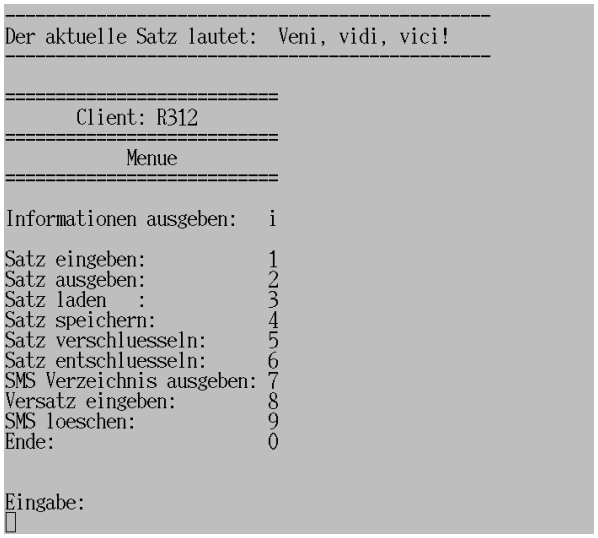

Abbildung 1: Die textbasierte Oberfläche für das Caesar-Verfahren

Mit diesem Versuch haben wir eine Struktur, die den Schülern verdeutlicht, welche Probleme auftreten, wenn viele Partner mit vielen anderen Partnern kommuniziieren wollen. Es treten Probleme auf, die z.B. Militärs bei ihrer Kommunikation nicht kannten und nicht kennen. Beim Austesten des Programmes tritt entsprechend zumeist nach einer kurzen Arbeitsphase eine erhebliche Unruhe auf. Verschlüsselungszahlen werden gesucht, teilweise durch den Raum gerufen oder durch kurze „Besuche“ bei Kommunikationspartnern mitgeteilt. Natürlich wird die schwache Verschlüsselung zum Mitschnüffeln fremder Nachrichten schnellstens geknackt. Anschliessend können bei der Auswertung dieser Phase leicht alle Probleme einer solchen Kommunikation aufgearbeitet werden.

Die hier eingeführte Arbeitsweise kann nun ohne Weiteres mit den weiteren Verfahren immer wieder aufgenommen werden, um die Tauglichkeit von Verfahren zu testen.

2.2 Das Vigenère-Verfahren

Eine wesentliche Verbesserung des Caesar-Verfahrens zu wird *Blaise de Vigenère* (1523-1596) zugeschrieben. Die Idee ist so einfach wie weitreichend: Statt jedes Zeichen gleichartig zu bearbeiten,

[7] Sofern noch nicht geschehen, kann hier oder evtl. auch entsprechend später auf die Problematik der (Nicht-)Verschlüsselung bei der Kommunikation zwischen Client und Server eingegangen werden.

werden die Zeichen unterschiedlich durch ein anderes Zeichen ersetzt. Naheliegend ist, entsprechend eines Verschlüsselungswortes- bzw. -satzes die Zeichen des Klartextes zu verschlüsseln. Dieses Verschlüsselungswort kennen natürlich nur Sender und Empfänger der Nachricht.

Auch bei diesem Verfahren handelt es sich um ein symmetrisches Verfahren, das heisst: Ver- und Entschlüsselung werden nach demselben Verfahren durchgeführt. Die Qualität der Verschlüsselung hängt im Wesentlichen von der Länge des Verschlüsselungsatzes ab. Es versteht sich von selbst, dass die Sicherheit durch ein häufiges Wechseln des Verschlüsselungswortes erhöht wird. Beispielsweise können die ersten Sätze der Seiten eines Literaturwerkes als Codewort verwendet werden: Verwendet wird die Seite mit der Nummer des entsprechenden Tages im Jahr. Beispielsweise wird am 10. Februar der erste Satz der Seite 41 der „Schwarte“ XYZ verwendet.

In unserem Programm müssen wir nur die Ver- und Entschlüsselungsprozeduren ersetzen und erhalten beispielsweise für die Verschlüsselung:

```
...
proc vigenere_verschluesseln {} {
  global satz
  global codewort

  set stelle 0
  set codewortstelle 0
  set verschluesselter_satz ""

  set codewort [string toupper $codewort]
  set laenge [string length $satz]
  set codewortlaenge [string length $codewort]

  while {$stelle < $laenge} {
    set zeichen [string index $satz $stelle]
    scan $zeichen %c asciizahl
    set verschluesselungszeichen [string index $codewort $codewortstelle]
    scan $verschluesselungszeichen %c verschluesselungszahl
    set verschluesselungszahl [expr $verschluesselungszahl -64]

    set asciizahl [expr $asciizahl + $verschluesselungszahl]
    set zeichen [format %c $asciizahl]
    set verschluesselter_satz $verschluesselter_satz$zeichen

    set stelle [expr $stelle + 1]
    set codewortstelle [expr $codewortstelle +1]
    if {$codewortstelle == $codewortlaenge} {
        set codewortstelle 0
        }
    }
  set satz $verschluesselter_satz
  }

proc vigenere_entschluesseln {} {
  global satz
  global codewort

  set stelle 0
  set codewortstelle 0
  set entschluesselter_satz ""

  set codewort [string toupper $codewort]
```

```
43  set laenge [string length $satz]
44  set codewortlaenge [string length $codewort]
45
46  while {$stelle < $laenge} {
47    set zeichen [string index $satz $stelle]
48    scan $zeichen %c asciizahl
49    set entschluesselungszeichen [string index $codewort $codewortstelle]
50    scan $entschluesselungszeichen %c entschluesselungszahl
51    set entschluesselungszahl [expr $entschluesselungszahl -64]
52
53    set asciizahl [expr $asciizahl - $entschluesselungszahl]
54    set zeichen [format %c $asciizahl]
55    set entschluesselter_satz $entschluesselter_satz$zeichen
56
57    set stelle [expr $stelle + 1]
58    set codewortstelle [expr $codewortstelle +1]
59    if {$codewortstelle == $codewortlaenge} {
60        set codewortstelle 0
61        }
62    }
63  set satz $entschluesselter_satz
64  }
```

Es ist offensichtlich, dass eine Entschlüsselung ohne Kenntnis des Verschlüsselungwortes nur noch für Experten möglich sein kann. Interessant ist nun, wie die Schüler die Begrenztheit dieses Verfahrens in einer Kommunikationsstruktur von vielen Teilnehmern mit vielen anderen Ad-Hoc-Teilnehmern erkennen. Wie schon beim Caesar-Verfahren erläutert, sollen sich die Schülerinnen und Schüler verschlüsselte Kurzmitteilungen zusenden. Es zeigt sich, dass oft eine noch hektischere „Nebenkommunikation“ als bei der Anwendung des Caesar-Verfahrens stattfindet. Die durch die einfache Caesar-Verschlüsselung codierten Sätze konnten einige Schüler durch systematisches Probieren decodieren und sie hatten daher keine weiteren Informationen von ihren Partnern verlangt. Dies geht nun nicht mehr und bei ca. 15 Computerarbeitsplätzen im Unterrichtsraum führt dies dann oft zu einer intensiven Kommunikation über alle Kanäle. Werden die Schüler sogar auf zwei Unterrichtsräume verteilt, finden zwangsläufig Wanderungen zwischen den Räumen statt oder es wird versucht, den Lehrer als „Codewort-Übermittler“ einzusetzen.

Bei der Auswertung fällt es den Schülerinnen und Schülern leicht, die Problematik dieses Verfahrens in einer heutigen vernetzten Struktur festzustellen: Dieses Verfahren ist nur sicher anzuwenden, wenn die Codeworte auf einem anderen Kanal als das Computernetz übertragen werden. Zudem muss dieser Kanal noch relativ schnell sein, damit i.A. eine schnelle Kommunikationsaufnahme ermöglicht wird[8].

Für die Militärs der letzten Jahrhunderte und/oder die Kommunikation zwischen ausgewählten Partnern wie beim *Roten Draht* zwischen Moskau und Washington ist dies sicher ein geeignetes System[9], jedoch nicht für die heutige Massenkommunikation in einer vernetzten Welt.

2.3 Das Vigenère-Verfahren mit grafischer Oberfläche

Wie oben erwähnt, können die Routinen für die Ver- und Entschlüsselung nach Vigenère in das beim Caesar-Verfahren beschriebenen Hauptprogramm eingebaut werden. Statt dieses textbasierten Hauptprogrammes kann natürlich auch bei allen Verfahren mit *Tcl/Tk* eine grafische Oberfläche erzeugt werden. Das Hauptprogramm muss dann umgestellt werden, die inhaltlichen Prozeduren, genannt *Fachkonzept*, können weiter benutzt werden. Das Hauptprogramm könnte dann folgendermassen aussehen[10]:

[8] Auf die Möglichkeit, in begrenztem Umfang doch einen Schlüssel über einen derartigen unsicheren Kanal festzulegen, wird an dieser Stelle nicht eingegangen [Hro06, S.226].

[9] Die dabei angewandte zweifelsfrei absolut abhörsichere Variante **Vernamchiffre** (1926) kann hierbei angesprochen werden.

[10] Auf eine syntaktische Beschreibung der grafischen Komponente Tk im Sprachkonzept Tcl/Tk muss hier verzichtet werden. Neben schriftlicher Literatur [Ous95],[HM98],[Web00] sind im Internet aber „unzählige“ Beschreibungen erhältlich.

```
# ******************   Aufbau der grafischen Oberfläche   **************
proc beenden {} {
  destroy .kopfrahmen
  destroy .adressrahmen
  destroy .satzrahmen
  destroy .codewortrahmen
  destroy .sms_verzeichnis
  destroy .menuerahmen
  destroy .info
  destroy .schlussrahmen
  }


beenden
info_anzeigen

set pfad "server"

set adresse 17
set partneradresse 0
set satz "Dies ist eine SMS"
set codewort "Codewort"
set SMS_verzeichnis "Verzeichnis der SMS"

frame .kopfrahmen -pady 30
frame .adressrahmen -pady 20
frame .satzrahmen -pady 10
frame .codewortrahmen -pady 10
frame .menuerahmen -pady 5
frame .schlussrahmen -pady 5

label .kopfrahmen.kopflabel -text "Verschlüsselung nach Vigenere"
.kopfrahmen.kopflabel configure -font "*-arial-*-*-*-*-26-*-*-*-*-*-*-*" \
                                -bg black -fg white -relief raised


label .adressrahmen.adresslabel -text "Meine eigene Adresse ist  R"
entry .adressrahmen.adresse -width 6 -relief sunken -textvariable adresse
label .adressrahmen.partneradresslabel -text "   Die Partner-Adresse ist
R"
entry .adressrahmen.partneradresse -width 6 -relief sunken \
                                   -textvariable partneradresse

entry .satzrahmen.satz -width 50 -relief sunken -textvariable satz
.satzrahmen.satz configure -font "*-arial-*-*-*-*-16-*-*-*-*-*-*-*"

label .codewortrahmen.codewortlabel -pady 10 \
          -text "Das aktuelle Codewort lautet derzeit"
entry .codewortrahmen.codewort -width 20 -relief sunken \
          -textvariable codewort


message .sms_verzeichnis -textvariable SMS_verzeichnis \
                         -width 475 -bg grey -relief sunken

button .menuerahmen.verschluesseln -text Verschlüsseln \
```

```
56        -command {set satz [vigenere_verschluesseln $satz $codewort]}
57  button .menuerahmen.entschluesseln -text Entschlüsseln \
58        -command {set satz [vigenere_entschluesseln $satz $codewort]}
59  button .menuerahmen.satz_laden -text "SMS laden" \
60        -command {set satz [satz_laden $adresse $partneradresse]}
61  button .menuerahmen.satz_speichern -text "SMS speichern" \
62        -command {satz_speichern $satz $adresse $partneradresse
63                  set SMS_verzeichnis [SMS_verzeichnis_ermitteln $adresse]
64                  }
65  button .menuerahmen.sms_verzeichnis -text "SMS Verzeichnis" \
66        -command {set SMS_verzeichnis [SMS_verzeichnis_ermitteln $adresse]}
67  button .menuerahmen.sms_loeschen -text "SMS löschen" \
68        -command {SMS_loeschen $adresse $partneradresse
69                  set SMS_verzeichnis [SMS_verzeichnis_ermitteln $adresse]
70                  }
71
72  button .schlussrahmen.info -text "Info anzeigen"  -width 20 \
73                             -command {info_anzeigen}
74  button .schlussrahmen.schluss -text "Programm beenden"  -width 20 \
75                             -command beenden
76
77
78
79  pack .kopfrahmen
80  pack .kopfrahmen.kopflabel
81  pack .adressrahmen
82  pack .adressrahmen.adresslabel -side left
83  pack .adressrahmen.adresse  -side left
84  pack .adressrahmen.partneradresslabel -side left
85  pack .adressrahmen.partneradresse  -side left
86
87  pack .satzrahmen
88  pack .satzrahmen.satz
89
90  pack .codewortrahmen
91  pack .codewortrahmen.codewortlabel -side left
92  pack .codewortrahmen.codewort -side left
93
94  pack .sms_verzeichnis
95
96  pack .menuerahmen
97  pack .menuerahmen.verschluesseln -side left
98  pack .menuerahmen.entschluesseln -side left
99  pack .menuerahmen.satz_laden -side left
100 pack .menuerahmen.satz_speichern -side left
101 pack .menuerahmen.sms_verzeichnis -side left
102 pack .menuerahmen.sms_loeschen -side left
103
104 pack .schlussrahmen
105 pack .schlussrahmen.info -side left
106 pack .schlussrahmen.schluss -side left
107
108 # ******************   Ende der grafischen Oberfläche   **************
```

Dieses Programm erzeugt eine Oberfläche, auf der der Benutzer mit den „üblichen“ *Buttons* und *Listboxen* mit dem System kommuniziiert. Im Gegensatz zu den textbasierten Programmen, die entweder mit der *Shell*

`tcl` oder `wish` ausgeführt werden müssen, muss die grafische Oberfläche mit der *Shell* `wish` ausgeführt werden.

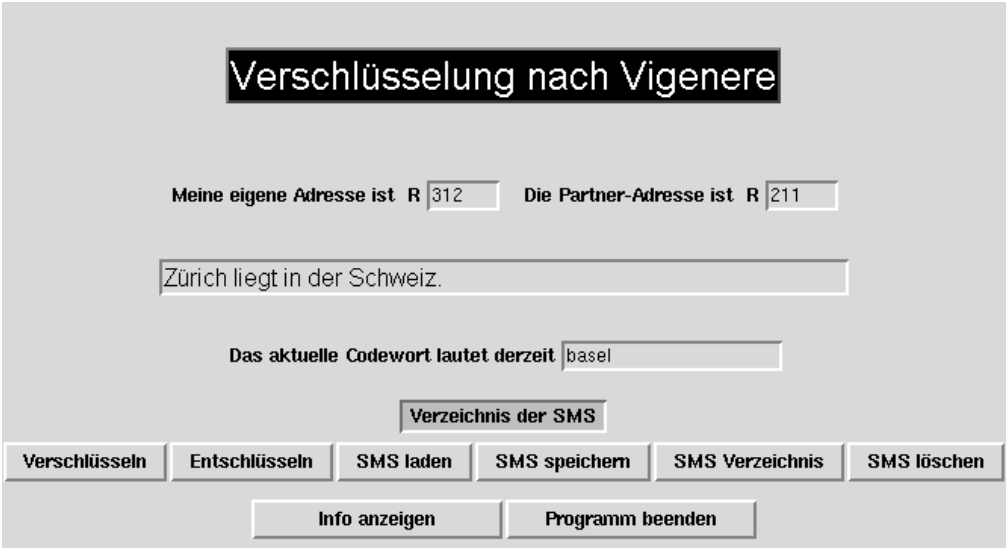

Abbildung 2: Die grafische Oberfläche für das Vigenère-Verfahren

2.4 Public-Key - Verfahren

Die oben erwähnten Verfahren finden sich in abgewandelter Form in mancher Literatur. Beim *Public-Key-Verfahren* versuchen jedoch die meisten Autoren, spezielle Varianten wie beispielsweise das *RSA-Verfahren* mit hohem mathematischen Aufwand vorzustellen [Sei94],[SW06a],[SW06b],[Beh84, S.26]. Die eigentliche Idee dieser Verfahren aus kryptologischer Sicht wird dabei nicht deutlich und sind für Schüler in der Sekundarstufe I, aber oft auch in der Sekundarstufe II nicht nachvollziehbar.

Ich verwende daher ein einfaches Public-Key-Verfahren, damit die Schüler die Prinzipien verstehen können. Dieses Verfahren arbeitet mit Verschlüsselungszahlen als Codeworte. In unserem Programm werden die entsprechenden Ver- und Entschlüsselungs-Prozeduren ersetzt durch.

```
1  ...
2  proc public_key_verschluesseln {} {
3    global satz
4    global schluessel
5
6    set internercode 123
7    set stelle 0
8    set codestelle 0
9    set internercodestelle 0
10   set verschluesselter_satz ""
11
12   set laenge [string length $satz]
13   set codelaenge [string length $schluessel]
14   set internercodelaenge [string length $internercode]
15
16   while {$stelle < $laenge} {
17     set zeichen [string index $satz $stelle]
18     scan $zeichen %c asciizahl
19
20     set codezahlzeichen [string index $schluessel $codestelle]
```

```
21      scan $codezahlzeichen %c codezahl
22      set codezahl [expr $codezahl -48]
23      set internercodezahlzeichen \
24          [string index $internercode $internercodestelle]
25      scan $internercodezahlzeichen %c internercodezahl
26      set internercodezahl [expr $internercodezahl -48]
27
28      set summencodezahl [expr $codezahl + $internercodezahl]
29      if {$summencodezahl > 9} {
30         set summencodezahl [expr $summencodezahl - 10]
31         }
32      set asciizahl [expr $asciizahl + $summencodezahl]
33
34      set zeichen [format %c $asciizahl]
35      set verschluesselter_satz $verschluesselter_satz$zeichen
36
37      set stelle [expr $stelle + 1]
38      set codestelle [expr $codestelle +1]
39      set internercodestelle [expr $internercodestelle +1]
40      if {$codestelle == $codelaenge} {
41          set codestelle 0
42          }
43      if {$internercodestelle == $internercodelaenge} {
44          set internercodestelle 0
45          }
46      }
47    set satz $verschluesselter_satz
48    }
49
50
51  proc public_key_entschluesseln {} {
52    global satz
53    global schluessel
54
55    set stelle 0
56    set codestelle 0
57    set entschluesselter_satz ""
58
59    set laenge [string length $satz]
60    set codelaenge [string length $schluessel]
61
62    while {$stelle < $laenge} {
63      set zeichen [string index $satz $stelle]
64      scan $zeichen %c asciizahl
65
66      set codezahlzeichen [string index $schluessel $codestelle]
67      scan $codezahlzeichen %c codezahl
68      set codezahl [expr $codezahl -48]
69
70      set asciizahl [expr $asciizahl - $codezahl]
71
72      set zeichen [format %c $asciizahl]
73      set entschluesselter_satz $entschluesselter_satz$zeichen
74
75      set stelle [expr $stelle + 1]
76      set codestelle [expr $codestelle +1]
```

```
77      if {$codestelle == $codelaenge} {
78          set codestelle 0
79          }
80      }
81    set satz $entschluesselter_satz
82    }
```

Auch wenn diese Programm nicht wesentlich länger als das Programm zur Vigenère-Verschlüsselung ist, erstellen diese Programmteile die Schüler nicht selber. Für Schüler der Sekundarstufe I sind Programme dieser Länge kaum noch selbst zu realisieren. Das Lesen und Interpretieren derartiger Programme ist oft schon eine Überforderung schwächerer Schüler. Allerdings nutzen manche Schüler bei der Erarbeitung des Verfahrens wie weiter unten beschrieben den Quellcode. Diese Prozeduren sind aber ebenso in das übliche Hauptprogramm (textbasiert oder grafisch) integriert, sodass die Schüler im Verlaufe des Unterrichts darauf problemlos zurückgreifen können. Im Gegensatz zur unterrichtlichen Praxis bei den bisherigen Verfahren sollen die Schüler diese Prozeduren nicht nach unterrichtlichen Überlegungen für ein wie auch verbessertes System diese entwickeln und programmieren, sondern sie sollen die Wirkung des neuen Verfahrens ausprobieren.

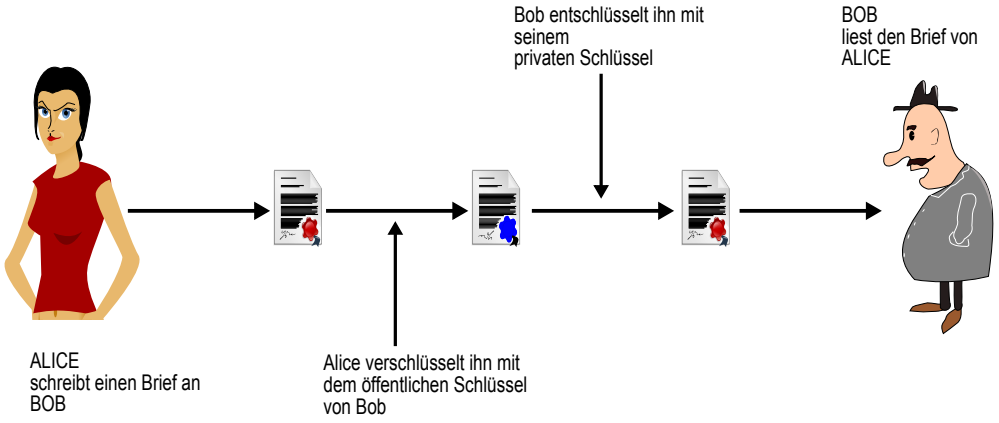

Abbildung 3: Die Idee des *public-key*-Verfahrens

2.5 Unterrichtliche Untersuchung des public-key-Verfahrens

Ich gebe für die Schüler nicht sichtbar einen Satz ein und verschlüssele diesen für die Schüler sichtbar mit einer ihnen dadurch bekannten Verschlüsselungzahl. Dieser chiffrierte Satz wird auf dem gemeinsamen Verzeichnis gespeichert und die Schüler erhalten den Auftrag, diesen Satz zu entschlüsseln. Die Schüler machen sich an die Arbeit und sind zunächst erstaunt, dass eine Entschlüsselung mit der ihnen doch bekannten Verschlüsselungszahl nun so nicht mehr möglich ist. Da das Verfahren bewusst einfach ist, wird es zumeist nach einiger Zeit von einigen pfiffigen Schülern oder Schülerinnen *geknackt*. Die Schülerinnen und Schüler erkennen:

- Es gibt Verfahren, bei denen *zwei* Schlüsseln notwendig sind. Ein Schlüssel wird zum Verschlüsseln der Nachricht verwendet: der *public key*. Ein zweiter Schlüssel, den sinnvoller Weise nur der Empfänger der Nachricht kennt, wird zum Entschlüsseln benutzt: der *private key*.

- Dieses Verfahren verlangt, dass die Systeme des Senders und des Empfängers die Berechnungsvorschrift kennen müssen. Damit die Kommunikation nicht abgehört werden kann, ist es notwendig, dass die Berechnung sehr aufwändig ist[11].
- Ein Problem stellt die Erstellung der Schlüsselpaare dar. Es darf natürlich nicht möglich sein, dass ein Unberechtigter zu einem öffentlichen Schlüssel den zugehörigen privaten Schlüssel ermitteln kann. Das bedeutet, dass es entweder vertrauenswürdige *Schlüsselerzeugungsstellen* geben muss oder das verwendete System kann nach einem Zufallsprinzip für die Benutzer Paare bilden, wobei die Wahrscheinlichkeit für ein Auftreten bisher verwendeter Paare praktisch Null sein muss. Bekannt sind die Programme *PGP* [Kip06, S.345] und die freie Variangte *GnuPG*[12].Hier entwickeln sich oft interessante aktuelle gesellschaftspolitische Diskussionen mit und für die Schüler.

Natürlich wird auch dieses Verfahren praktisch ausgetestet. Die Schüler erhalten dabei von Lehrer eine Liste der *öffentlichen* Schlüssel ihrer Mitschüler[13]. Jedem Einzelnen wird sein *privater* Schlüssel mitgeteilt. Und siehe da: eine Kommunikation in einem offenen Netz ist sinnvoll möglich.

In einer intensiven Diskussion werden die Verfahren noch einmal vergleichsweise gegenübergestellt und bewertet.

2.6 Signieren von Daten

Nachdem die Schülerinnen und Schüler das *Prinzip der Public-Key-Verfahren* verstanden und auch praktiziert haben, kann die *Authenzität* von Nachrichten thematisiert werden [Bau99a],[Bau99b]. Jemand kann unter Täuschung über seine echte *Identität* mit meinem *öffentlichen* Schlüssel zwar eine verschlüsselte Nachrichten zukommen lassen, die nur der Empfänger lesen kann, aber woher weiss dieser, dass sein Gegenüber auch der ist, als der er sich auszugeben versucht. Diese Problematik ist Schülerinnen und Schüler nicht unbekannt, sind sie es doch gewohnt und lieben es, z.B. in Chaträumen eine Anonymität durch eine falsche Identität zu geniessen, sehen aber natürlich ein, dass eine Kommunikation in einer vernetzten Gesellschaft natürlich auch die *Authenzitätfrage* lösen muss.

Naheliegend ist nun, das der Klartext zweimal verschlüsselt wird: Zuerst wird der private Schlüssel des Absenders verwendet, dann der öffentliche Schlüssel des Empfängers. Der Empfänger decodiert nun den Chiffriertext mit seinem privaten Schlüssel und dann mit dem öffentlichen Schlüssel des Absenders und kann dann die Nachricht lesen. Mit diesem Verfahren ist der Text vom Absender also *signiert* worden[14].

Das von uns verwendete Verfahren hat eine besondere Eigenart. Es ist nicht symmetrisch, das heisst: Der zum Signieren zu verwendende private Schlüssel ist nicht identisch mit dem privaten Schlüssel zum Decodieren von Nachrichten[15]. Die Schüler erhalten daher zwei private Schlüssel. In der Praxis verwendete Verfahren sind verständlicherweise symmetrisch. Die Schülerinnen und Schüler können sich nun signierte verschlüsselte Nachrichten zusenden. Das Handling mit den verschiedenen Schlüsseln verlangt aber ein genaues konzentriertes Arbeiten und zeigt daher, in wie weit die Schüler die Verfahren auch tatsächlich verstanden haben und dann praktizieren können.

3 Aufbauende Unterrichtseinheiten

3.1 E-Mail

Das im vorigen Kapitel vorgestellte Programmsystem kann als eine einfache Form eines E-Mail-Systems verstanden werden. Es bietet sich daher an, anschliessend das heutige real verwendete E-Mail-System im Internet zu analysieren. Dabei können z.B. problematische nicht verschlüsselte Protokoll-Mechanismen beim Senden

[11] An dieser Stelle kann ein kurzer Hinweis auf konkrete Realisierungen wie z.B. RSA erfolgen. Eine genauere Analyse verbietet sich meines Erachtens auf jeden Fall im Unterricht in der Sek I, aber eigentlich auch in der Sek II.

[12] Informationen über das System GPG und Programm selber sind im Internet über `http://gnupg.org/index.de.html` zu erhalten.

[13] Die Zertifizierungsstelle ist hier also der Lehrer.

[14] Für das Verständnis ist dabei unerheblich, ob in der Praxis der gesamte Text, nur Teile davon oder Prüfsummen signiert werden. In der Praxis wird aber nur ein Teil, die bekannte *Unterschrift* signiert, da ansonsten Dokumente „willkürlichen" Inhalts nicht authentifiziert werden könnten

[15] Dieser Fachausdruck *symmetrisch* darf nicht mit dem Fachausdruck *symmetrisch* bei den Verfahren allgemein gleichgesetzt werden.

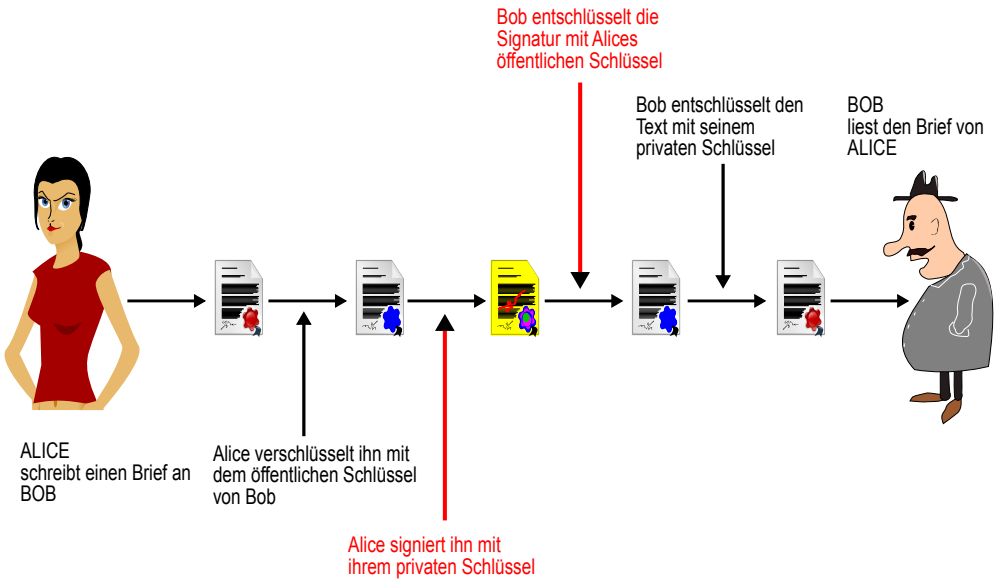

Abbildung 4: Die *Authentisierung* mit dem Public-Key-Verfahren

und Empfangen von E-Mail untersucht werden. Ebenso kann das Verschlüsseln von E-Mails mit Systemem wie PGP und das quelloffene GnuPG besprochen werden. Mit einem geschärften Bewusstsein können dann auch weitere Probleme bei der E-Mail-Benutzung besprochen werden.

Anschliessend sollte die Problematik von Anhängen erarbeitet werden. Vor allem die Möglichkeit der Einbettung von ausführbaren Programmen in vom Benutzer nicht lesbaren (Text-)Dokumenten (vor allem in den Dokumenten eines sachlich nicht gerechtfertigten weit verbreiteten Software-Systems) im Gegensatz zu analysierbaren Dokumenten sollte zusätzlich besprochen werden [Rüg03].

3.2 Betriebssyteme und Netze

Eine weitere Möglichkeit ist die Untersuchung von (aktuellen) Betriebssystemen hinsichtlich der Nutzung von kryptografischen Methoden zur Benutzer- und Datenverwaltung. Unter anderem kann analysiert werden, welche Protokolle für Fernzugriffsmethoden auf diese Systeme und/oder Netze (z.B. ssl, vpn, https) verwendet werden und wie sicher diese sind [ADH+00].

3.3 Online-Banking

Die Gefährdung der Computersysteme beim und durch Online-Banking hat in den letzten Jahren enorm zugenommen. Die Schüler der Sekundarstufe I werden in naher Zukunft auch Bankgeschäfte mit allen Möglichkeiten und Risiken durchführen. Sie haben dagegen oft relativ wenig Informationen über die tatsächliche Funktionsweise der heute üblichen Systeme. Da viele Banken einen Testzugang für ihr Online-Banking ermöglichen, ist ein praktischen Austesten und die Untersuchung teilweise unterschiedlichenr Realisierungen möglich. Bei der Analyse des Online-Bankings kann das Verschlüsseln von Daten auf verschiedenen Ebenen wunderbar betrachtet und untersucht werden [Kou06].

Sowohl bei dieser Thematik als auch beim Thema *Betriebssysteme und Netze* kann dann auch genauer z.B. an Grafiken wie Abbildung 1 und 2 untersucht werden, an welchen Stellen eine dritte Person — zumeist *Claire* genannt — angreifen kann und welche Komponenten der Kommunikation physikalisch und logisch an welchem konkreten Platz zu positionieren sind.

4 Schlussbemerkungen

Praktisch keiner kann sich heute mehr aus der Welt mit Informationssystemen ausklinken. Es ist eher dagegen davon auszugehen, dass immer mehr Aktivitäten aus der „realen“ Welt in die „virtuelle Welt“ ausgelagert werden. Dabei werden an vielen Stellen auch die verschiedensten Verschlüsselungsverfahren praktiziert.

Daher muss eine Schülerin bzw. ein Schüler heute nach Abschluss der Sekundarstufe I ein prinzipielles Verständnis über Verschlüsselung im Allgemeinen und den alternativen Prinzipien besitzen. Das bezieht sich nicht nur auf die Schülerinnen und Schüler, die im Augenblick z.B. das Wahlpflichtfach Informatik in der Sekundarstufe I belegen. Es gehört einfach in einer vernetzten Welt zur Allgemeinbildung dazu.

Die dazu notwendigen Kenntnisse und Kompetenzen können und werden nicht in einem der heutigen üblichen Pflichtfächer vermittelt. Es ist sicher nicht verwegen, zu behaupten, dass auch nur ein kleiner Teil der Nicht-Informatiker auch nur über annähernde Sachkenntnisse über Verschlüsselungsverfahren und deren Anwendung verfügt. Da dies bisher auch nicht zum Kanon einer *Allgemeinbildung* gehört hat, kann das auch nicht verwundern.

Wenn dem aber nicht so ist, dann ergibt sich aus dieser Tatsache bereits der Zwang zu einem Pflichtfach Informatik in der Sekundarstufe I. An diesem Beispiel wird deutlich, dass das u.a. von Wilkens in ihrer Dissertation korrekt beschriebene „allmähliche Verschwinden der informationstechnischen Grundbildung“ nicht dazu führt, dass „für die allgemeinbildenden Inhalte eines Faches Informatik ein Aufgehen in andere Fächer und Lernbereiche als Alternative immer mehr in Betracht zu kommen“ scheint [Wil00, S.63]. Die Kryptologie ist vielmehr mit ein Schulfach begründendes Gebiet der Informatik , also eine *fundamentale Idee der Informatik* [Sch93] und keine „Modeerscheinung“[16].

[16] Dies wird allein schon durch die Jahreszahlen der in der Literaturliste aufgeführten Veröffentlichungen deutlich.

Literatur

[ADH+00] ABROSIMOV, Leonid I. ; DEUTSCHMANN, Jörg ; HORN, Werner ; REIF, Holger ; RESCHKE, Dietrich ; SCHILLER, Jochen ; SEITZ, Jochen ; KRÜGER, Gerhard (Hrsg.) ; RESCHKE, Dietrich (Hrsg.): *Lehr- und Übungsbuch Telematik*. Leipzig : Fachbuchverlag, 2000. – ISBN 3–446–21053–9

[Bat96] BATZER, Peter: Die ENIGMA. In: *LOG IN* 16 (1996), Nr. 5/6, S. 44–51

[Bau99a] BAUMANN, Rüdeger: Digitale Unterschrift, Sichere Rechtsgeschäfte im Internet (Teil 1). In: *LOG IN* 19 (1999), Nr. 2, S. 46–49

[Bau99b] BAUMANN, Rüdeger: Digitale Unterschrift, Sichere Rechtsgeschäfte im Internet (Teil 2). In: *LOG IN* 19 (1999), Nr. 3/4, S. 82–88

[Bau00] BAUER, Friedrich L.: *Entzifferte Geheimnisse : Methoden und Maximen der Kryptologie; mit ... 26 Tabellen*. 3., überarb. und erw. Aufl. Berlin : Springer, 2000. – ISBN 3–540–67931–6

[Beh84] BEHRENS, Rolf: *Zahlentheorie und Datenschutz*. Stuttgart : Landesstelle für Erziehung und Unterricht, 1984 (Materialien zur Lehrerfortbildung, Reihe Mathematik)

[Chr85] CHRISTENSEN, Børge R: *Strukturierte Programmierung mit COMAL 80, 2., verbesserte Auflage*. München : Oldenbourgh-Verlag, 1985

[Hei85] HEINE, Werner: *Die Hacker*. Reinbeck bei Hamburg : Rowohlt Taschenbuch Verlag, 1985

[HM98] HARRISON, Mark ; MCLENNAN, Michael: *Effektiv Tcl/Tk programmieren*. Bonn [u.a.] : Addison-Wesley, 1998 (Professionelle Programmierung). – ISBN 3–8273–1409–7

[Hro06] HROMKOVIČ, Juraj: *Sieben Wunder der Informatik : eine Reise an die Grenze des Machbaren mit Aufgaben und Lösungen*. 1. Aufl. Wiesbaden : Teubner Verlag, 2006 (Lehrbuch Informatik). – ISBN 3–8351–0078–5 ; 978–3–8351–0078–7

[Hro08] HROMKOVIČ, Juraj: *Lehrbuch Informatik : Vorkurs Programmieren, Geschichte und Begriffsbildung, Automatenentwurf*. 1. Aufl. Wiesbaden : Vieweg und Teubner Verlag, 2008 (Studium). – ISBN 978–3–8348–0620–8

[Ihm08] IHM, Maximilian (Hrsg.): *Gaius Suetonius Tranquillus: De vita Caesarum*. Stuttgart : B.G. Teubner, 1908

[Kip06] KIPPENHAHN, Rudolf: *Verschlüsselte Botschaften, 4. Auflage*. Hamburg : Nikol-Verlag, 2006

[Köh03] KÖHLER, Klaus: Sicherheit durch Kryptographie? In: *FIFF Kommunikation* 20 (2003), Nr. 2, S. 39–46

[Kou06] KOUBEK, Jochen: Sicherheit von Online-Bezahldiensten. In: *LOG IN* (2006), Nr. 140, S. 25–29

[Ous95] OUSTERHOUT, John K.: *Tcl und Tk : Entwicklung grafischer Benutzerschnittstellen für das X-Window-System*. 1. Aufl. Bonn : Addison-Wesley Professional Computing, 1995 (Professional computing). – ISBN 3–89319–793–1

[Pas05] PASTERNAK, Arno: Was wir vergessen haben. (2005). – `http://pasternak.in.hagen.de/if/koenigstein/vergessen.htm`

[Rüg03] RÜGER, Hubert: Information Security. In: *FIFF Kommunikation* 20 (2003), Nr. 2, S. 37–38

[Sch93] SCHWILL, Andreas: Fundamentale Ideen der Informatik. In: *Zentralblatt. für Didaktik der Mathematik* (1993), S. 20–31

[Sei94] SEIFFERT, Monika: Verschlüsselungsmethoden (Teil 2). In: *LOG IN* 14 (1994), Nr. 3, S. 33–40

[Str97] STRATHERN, Paul: *Turning & der Computer*. Frankfurt a.M. : Fischer Taschenbuchverlag, 1997

[SW06a] SCHULZ, Ralph-Hardo ; WITTEN, Helmut: RSA & Co. in der Schule (Neue Folge – Teil 1). In: *LOG IN* (2006), Nr. 140, S. 45–54

[SW06b] SCHULZ, Ralph-Hardo ; WITTEN, Helmut: RSA & Co. in der Schule (Neue Folge – Teil 2). In: *LOG IN* (2006), Nr. 143, S. 50–58

[Web00] WEBSTER, Tim ; FRANCIS, Alex (Hrsg.): *Tcl/tk für Dummies : gegen den täglichen Frust mit Tcl/Tk*. Bonn : mitp, 2000. – ISBN 3–8266–2872–1

[Wil00] WILKENS, Ulrike: *Das allmähliche Verschwinden der informationstechnischen Grundbildung*. Aachen : Shaker-Verlag, 2000. – ISBN 3–8265–7125–8