Adam-Kraft-Gymnasium
Schwabach
Kollegstufe 2007/2009 Abiturjahrgang 2009

FACHARBEIT

aus dem Leistungskurs Mathematik

Thema

RSA-Verschlüsselung

Verfasser: Julian von Mendel
Leistungskurs: Mathematik
Kursleiter: Frau Ziegerer
Abgabetermin: 30.01.2009 (12:00 Uhr)

Bewertung:

Facharbeit:	Punkte
mündliche Prüfung:	Punkte
Gesamtwertung:	**Punkte**

. . . . . .

(Unterschrift des Kursleiters)

Inhaltsverzeichnis

1 Vorwort

1.1 Schwäche symmetrischer Verfahren

Symmetrische Verschlüsselung – d.h. die Ver- und Entschlüsselung einer Nachricht mit ein und demselben Schlüssel – steht im Gegensatz zur asymmetrischen Verschlüsselung (auch: **Public-Key-Verfahren**). Symmetrische Verschlüsselungsverfahren haben ein großes Problem: Sie sind nur dann sicher, wenn der Schlüssel sicher übertragen wurde. Das ist in der Praxis nur durch ein direktes Treffen möglich. In Zeiten des Internets, in denen man Daten sicher austauschen möchte, ohne dass räumliche Nähe besteht, z.B. um eine verschlüsselte E-Mail an einen Fremden zu senden, mit dem man sensible Daten teilen möchte, ist dieser Schlüsselaustausch kaum umsetzbar.

1.2 Kryptografie ohne das Problem der Schlüsselübertragung?

Die grundsätzliche Möglichkeit, Informationen ohne Schlüsselaustauch sicher zu übertragen, möchte ich hier anhand eines leicht nachvollziehbaren Beispiels veranschaulichen: Alice möchte Bob eine Schatztruhe schicken, ist jedoch überzeugt, dass die Postbotin Mallory mit allen Mitteln versuchen wird, an den Inhalt der Truhe zu kommen. Alice kann Bob nicht persönlich treffen, und kann nur per Post mit ihm kommunizieren. Wenn Sie die Truhe mit einem Schloss sichert hat sie das Problem, dass Bob den Schlüssel dafür besitzen müsste, um die Truhe öffnen zu können. Folgender Algorithmus umgeht die scheinbar unlösbare Sicherheitslücke des Schlüsselaustausches:

Alice sichert die Truhe mit ihrem (angenommen unknackbaren) Schloss. Sie schickt die Truhe per Post zu Bob, der diese nicht öffnen kann, da er keinen Schlüssel zum Schloss besitzt. Bob verwendet ein gleichwertiges Schloss, das er zusätzlich an der Öffnung der Truhe befestigt, und schickt die Truhe zu Alice zurück. Diese entfernt ihr eigenes Schloss, mit dem Schlüssel, den die ganze Zeit über nur sie hatte, und schickt die Truhe erneut zu Bob. Bob kann die Truhe jetzt mit seinem eigenen Schlüssel öffnen.

Das Beispiel zeigt, dass man ohne Schlüsselaustausch Informationen sicher übertragen kann, es steht nicht in direktem Zusammenhang zum 1977 erstmals beschriebenen RSA-Verfahren[1] (benannt nach den Entwicklern Ronald Rivest, Adi Shamir und Leonard Adleman).

[1] Beim RSA-Verfahren ist **kein** mehrmaliges Hin- und Herschicken der Nachricht notwendig.

2 Begriffserklärungen und mathematische Grundlagen

2.1 Kryptologie/Kryptografie/Kryptoanalyse/Steganografie

Kryptologie beschäftigt sich allgemein mit Informationssicherheit, in der Regel auf Basis technischer oder rein mathematischer Algorithmen. Die Kryptologie wird in die Kryptografie und die Kryptoanalyse unterteilt.

Kryptografie beschäftigt sich mit dem Transformieren von Informationen auf eine umkehrbare Art und Weise.

Kryptoanalyse befasst sich mit dem Brechen von Verschlüsselungsverfahren oder speziellen, verschlüsselten Texten, oder aber dem Nachweis, dass ein Verfahren (un)sicher ist.

Steganografie versucht, die Existenz einer Information zu verheimlichen. Beispiel: Verstecken eines kurzen Textes in einem Bild. Steganografie wird in der Regel mit Kryptografie kombiniert, da im Idealfall die versteckten Daten selbst zufällig wirken (d.h. keine dem Feind bekannte, statistische Häufigkeitsverteilung vorliegt) und es deshalb erstmal nicht möglich ist, zu ermitteln, ob überhaupt eine versteckte Information vorliegt.

Diese Facharbeit befasst sich mit dem asymmetrischen kryptografischen Verfahren RSA, und geht knapp auf kryptoanalytische Aspekte ein.

2.2 Kerckhoffs Maxime

Kerckhofs Maxime besagt, dass die Sicherheit eines Verschlüsselungsverfahren nicht auf der Geheimhaltung des Verfahrens beruhen darf, sondern auf der Geheimhaltung von einem oder mehreren Schlüsseln basieren sollte. Beispiel: Die Cäsar-Verschiebung, die einen Text verschlüsselt, in dem alle Buchstaben in einem Text um z.B. ein Zeichen verschoben werden (A $\rightarrow$ B, Z $\rightarrow$ A, etc.), ist unsicher, sobald der Feind Kenntnis des Verfahrens besitzt. Selbst wenn als Schlüssel die Verschiebungsweite verwendet werden würde, gäbe es zu wenig Möglichkeiten, als dass das Verfahren sicher wäre. Ist das Verfahren gut, ist eine Entschlüsselung ohne Kenntnis des Schlüssels oder ausreichend hohen Rechenkapazitäten nicht möglich.

2.3 Modulo-Operation

Die **Modulo-Operation** liefert den Rest einer Division zweier ganzer Zahlen.

$7 \bmod 2 = 1$	da	$7 = 3 \cdot 2 + 1$
$15 \bmod 15 = 0$	da	$15 = 1 \cdot 15 + 0$
$30 \bmod 15 = 0$	da	$30 = 2 \cdot 15 + 0$
$44 \bmod 15 = 14$	da	$44 = 2 \cdot 15 + 14$

2.4 Einwegverschlüsselung

Ein **Einwegverschlüsselunsgsverfahren** (auch **Hash-Funktion** genannt) erzeugt aus einem beliebig großen Quellwort eine Prüfsumme (oder **Hash**), in der Regel definierter Länge. Mehrere Quellwörter werden auf die gleichen Zielwörter abgebildet. Die Güte eines Einwegverschlüsselungsverfahrens wird durch mehrere Eigenschaften definiert:

- Eine beliebig kleine Änderung des Quellworts soll eine erhebliche/totale Änderung der Prüfsumme verursachen.
- Je weniger Quellwörter das gleiche Zielwort als Ergebnis haben, desto besser. An sich gibt es natürlich unendlich viele Quellwörter mit der gleichen Prüfsumme, aber in der Regel sind Hash-Funktionen auf bestimmte Eingangsdaten ausgelegt, z.B. mit einer Länge, die 500 Buchseiten nicht überschreitet. Ein gut geeignetes Hash-Verfahren weist bei allen möglichen Quellwörtern innerhalb dieses Größenbereichs eine so gut es geht lineare Verteilung an Prüfsummen auf.

Die Modulo-Operation ist ein Einwegverschlüsselungsverfahren, da sich aus einem der Ergebnisse nicht die Eingangsparameter eindeutig zurückrechnen lassen. Andere, komplexere, häufig verwendete Verfahren sind MD5 und SHA-1.

Hash-Funktionen haben diverse Anwendungen, hier eine Auswahl:

- Zur Bestimmung, ob Datenintegrität gegeben ist: Bei der Dokumenterstellung wird dem Dokument eine **Prüfsumme** zugeordnet. Ein Empfänger des Dokuments kann ebenfalls die Prüfsumme des Dokuments berechnen und vergleichen, ob diese mit der vom Absender berechneten übereinstimmt. Falls ja, ist das Dokument mit hoher Wahrscheinlichkeit unverändert (da, je nachdem, wie gut das Einwegverfahren ist, das Erzeugen eines gefälschten Dokuments mit der gleichen Prüfsumme äußerst rechenaufwändig wäre). Die ISBN und die Personalausweisnummer zum Beispiel enthalten kurze Prüfsummen.
- Zur Speicherung von Passwörtern: Statt dem Speichern des Klartextpasswords kann nur die Prüfsumme des Kennworts hinterlegt werden. Sollten die Prüfsummen in falsche Hände geraten, kann aus den Prüfsummen nicht unmittelbar auf das Kennwort geschlossen werden. Bei der Passworteingabe wird erneut die Prüfsumme erzeugt und verglichen. Um bei

gleicher Hash-Funktion bei verschiedenen Computerprogrammen trotzdem nicht die gleichen Prüfsummen zu erhalten (und so zu verhindern, dass z. B. ein Angreifer eine große Liste von Passwort-Prüfsummenzuordnungen erstellt, und in Zukunft alle mit der entsprechenden Hash-Funktion erzeugten Prüfsummen schnell zuordnen kann, **Rainbow-Table** genannt) werden sog. **Salts** verwendet, die bei der Prüfsummenerzeugung an ein Kennwort angehängt werden. Salts sind zufällige Zeichenketten, die das zu verschlüsselnde Passwort sicherer machen. Da eine kleine Änderung der Quellmenge eine signifikante Änderung der Zielmenge zur Folge haben sollte, wird dadurch verhindert, dass sich eine einzelne Datenbank mit Zuordnungen erstellen lässt (die nicht unvertretbare hohe Speicherkapazität beansprucht).

- Zur Identifizierung von Objekten: Unterschiedliche Objekte, die evtl. große Datenmengen beanspruchen, werden über kurze Prüfsummen identifiziert und zugeordnet.

2.5 Eulersche φ-Funktion

Die Eulersche φ-Funktion $\varphi(n)$ gibt an, wie viele positive, ganze Zahlen kleiner gleich n zu n teilerfremd sind, d.h. dass ihr größter gemeinsamer Teiler gleich 1 ist[2]. Beispiel: $\varphi(10) = 4$, da 1, 3, 7 und 9 zu 10 teilerfremd sind. Für Primzahlen gilt, da eine Primzahl per Definition nur durch 1 und sich selbst teilbar ist:

$$\varphi(p) = p - 1 \tag{1}$$

Die φ-Funktion ist multiplikativ[3], d.h. $\varphi(p \cdot q) = \varphi(p) \cdot \varphi(q)$. Demnach gilt für die Primzahlen p, q:

$$\varphi(p \cdot q) = (p - 1) \cdot (q - 1) \tag{2}$$

2.6 Satz von Euler

Unter der Bedingung $ggT(a, b) = 1$ (d. h. a und b sind teilerfremd) mit $a, b \in \mathbb{N}$ gilt[4]:

$$a^{\varphi(b)} \bmod b = 1 \tag{3}$$

Daraus folgt beispielsweise für $a = 3, b = 10$:

$$3^{\varphi(10)} \bmod 10 = 3^4 \bmod 10 = 81 \bmod 10 = 1$$

[2] [2, S.100]
[3] [1, S.129ff]
[4] Nach [1, S.135], [2, S.100]

Mit dem Wissen aus Def. 1 auf der vorherigen Seite, Def. 2 auf der vorherigen Seite und Def. 3 auf der vorherigen Seite lassen sich kompliziert zu berechnend wirkende Rechenaufgaben der Form $a^{(p-1)\cdot(q-1)} \bmod (p \cdot q)$ mit p, q als Primzahlen schnell lösen:

$$23^{192} \bmod 221 = 23^{12\cdot 16} \bmod 221 = 23^{\varphi(13\cdot 17)} \bmod 221 = 23^{\varphi(221)} \bmod 221 = 1$$

Der kleine Satz von Fermat ist ein Spezialfall des Eulerschen Satzes:

$$a^p \bmod p = a \tag{4}$$

mit $a \in \mathbb{N}$ und p als Primzahl. Wenn $a \bmod p \neq 0$ gilt, d.h. dass a kein Vielfaches von p ist, kann man zu

$$a^{p-1} \bmod p = 1 \tag{5}$$

umformen (oder dies direkt aus Def. 3 auf der vorherigen Seite schließen)[5].

2.7 Der euklidische Algorithmus

Zur Bestimmung des größten gemeinsamen Teilers ist das folgende Verfahren auch für große Zahlen geeignet: Von den Zahlen $a, b \in \mathbb{N}$ mit $a > b$ soll der ggT gebildet werden. Für $a = k_0 \cdot b + r_0$ wird k_0, r_0 so berechnet, dass beide Werte ganzzahlig sind. Anschließend das Gleiche für $k_0 = k_1 \cdot r_0 + r_1$, dann $k_1 = k_2 \cdot r_1 + r_2$, usw., bis der Rest $r_n = 0$ ist. k_n ist der resultierende ggT[6].

Beispiel: Es soll $ggT(2344, 1234)$ berechnet werden.

$$\begin{array}{lclclcl}
2344 & = & 1 & \cdot & 1234 & + & 1110 \\
1234 & = & 1 & \cdot & 1110 & + & 124 \\
1110 & = & 8 & \cdot & 124 & + & 118 \\
124 & = & 1 & \cdot & 118 & + & 6 \\
118 & = & 19 & \cdot & 6 & + & 4 \\
6 & = & 1 & \cdot & 4 & + & 2 \\
4 & = & 2 & \cdot & 2 & + & 0
\end{array}$$

Der größte gemeinsame Teiler ist die 2.

2.8 Erweiterter euklidische Algorithmus

Mit dem erweiterten euklidischen Algorithmus lassen sich Zahlen $x, y \in \mathbb{Z}$ bestimmen, die die Gleichung $ggT(a, b) = a \cdot x + b \cdot y$ erfüllen, so dass der größte

[5] [2, S.99]
[6] [2, S.90ff]

gemeinsame Teiler als eine Linearkombination aus a und b dargestellt wird.

Oben anknüpfendes Beispiel:

$$
\begin{aligned}
ggT(a,b) &= 2 = 6 - 1 * 4 = \\
&= 6 - 1 * (118 - 19 * 6) = \\
&= 20 * 6 - 118 = \\
&= 20 * (124 - 1 * 118) - 118 = \\
&= 20 * 124 - 21 * 118 = \\
&= 20 * 124 - 21 * (1110 - 8 * 124) = \\
&= 188 * 124 - 21 * 1110 = \\
&= 188 * (1234 - 1 * 1110) - 21 * 1110 = \\
&= 188 * 1234 - 209 * 1110 = \\
&= 188 * 1234 - 209 * (2344 - 1 * 1234) = \\
&= 397 * 1234 - 209 * 2344 = \\
&= a \cdot (-209) + b \cdot (397)
\end{aligned}
$$

Die Gleichungen sind alle nur Umformungen der Ergebnisse aus dem einfachen euklidischen Algorithmus. Es werden von unten nach oben die Terme durchgegangen, aufgelöst und ineinander eingesetzt, um r_x jeweils durch eine Linearkombination $k_{x-1} - k_x \cdot r_{x-1}$ darzustellen. Ziel ist es nicht auszurechnen, sondern zusammenzufassen[7]. In diesem Beispiel ergibt sich $x = -209$ und $y = 397$.

Gilt $ggT(a,b) = 1$, ist x das modulare Inverse von a modulo n, was wir uns zur Vereinfachung der RSA-Gleichungsberechnung ohne Computer zu Nutze machen können.

[7] [4, S.104ff]

3 Das RSA-Verfahren

3.1 Vorgehensweise zur Nachrichtübertragung

Asymmetrische Verfahren basieren darauf, dass beide Kommunikationspartner einen öffentlichen und einen privaten Schlüssel verwenden. Der öffentliche Schlüssel muss sich mittels einer Einwegfunktion aus dem privaten errechnen lassen und darf der ganzen Welt bekannt sein. Der private Schlüssel muss unter allen Umständen geheim gehalten werden. Mit dem öffentlichen Schlüssel des Empfängers und der Klartextnachricht lässt sich dann die verschlüsselte Nachricht errechnen, die der Empfänger mit seinem privaten Schlüssel dechiffrieren kann. Sowohl die verschlüsselte Nachricht, als auch die öffentlichen Schlüssel beider Kommunikatinspartner dürfen in fremde Hände geraten – ohne den privaten Schlüsseln kann man damit nichts anfangen[8]. [7]

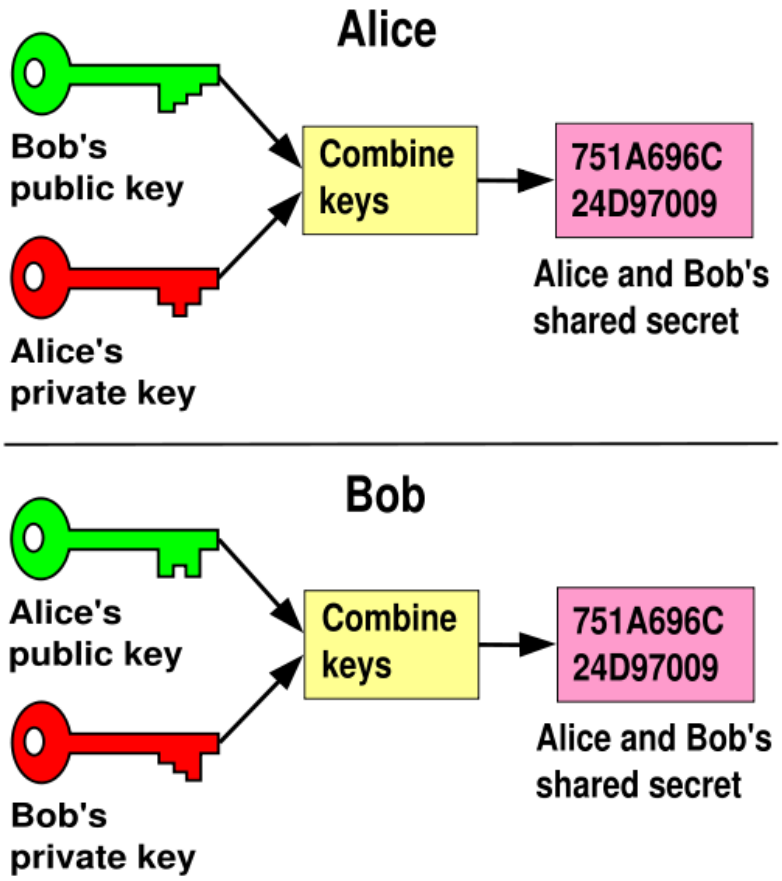

Abbildung 1: Das Konzept asymmetrischer Verschlüsselung (aus [9])

[8] Ausnahmen siehe Kapitel 5.1 auf Seite 17.

3.2 Schlüsselerzeugung

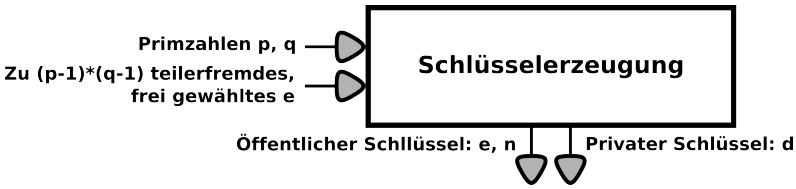

Abbildung 2: Ein- und Ausgangswerte bei der Schlüsselerzeugung

Der Schlüssel wird in drei einfachen Schritten von allen Kommunikationspartnern unabhängig erzeugt:

- Es werden zwei Primzahlen p, q gewählt, bei praktischer Verwendung sollten beide Primzahlen möglichst groß sein und den in Kapitel 5.3 auf Seite 17 näher erläuterten Ansprüchen genügen.
- $n = p \cdot q$ und $\varphi(n) = (p-1) \cdot (q-1)$ wird berechnet.
- Zwei natürliche Zahlen e, d werden so gewählt, dass $ggT(e, \varphi(n)) = 1 \wedge e \cdot d \bmod \varphi(n) = 1$ (d.h. $d = \frac{1+k \cdot \varphi(n)}{e}$) gilt[9].

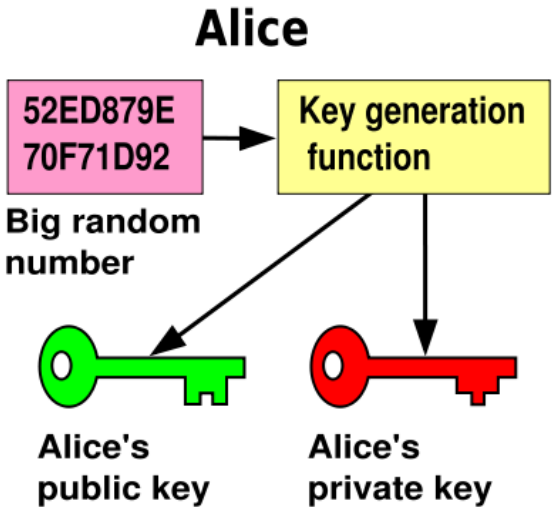

Abbildung 3: Schlüsselerzeugung (aus [9])

e und n bilden den öffentlichen Schlüssel, der allen Kommunikationspartnern bekannt sein muss. Die Zahl d ist der private Schlüssel. p, q und $\varphi(n)$ können/sollten vernichtet werden[10].

Beispiel: Wir wählen $p = 17$ und $q = 11$, demnach gilt

[9] Die Variable e hat hier **nichts** mit der eulerschen Zahl zu tun; e ist über die gesamte Literatur hinweg der übliche Variablenname.

[10] [5, S.17]

$$n = p \cdot q = 17 \cdot 11 = 187$$
$$\varphi(n) = (p-1) \cdot (q-1) = 160$$

Mit frei gewähltem, zu 160 teilerfremden $e = 19$ können wir den erweiterten euklidischen Algorithmus anwenden, um d zu bestimmen:

$$\begin{aligned} 160 &= 8 \cdot 19 + 8 \\ 19 &= 2 \cdot 8 + 3 \\ 8 &= 2 \cdot 3 + 2 \\ 3 &= 1 \cdot 2 + 1 \end{aligned}$$

$$\begin{aligned} 1 &= 3 - (1 \cdot 2) = \\ &= 3 - (8 - 2 \cdot 3) = \\ &= 3 \cdot 3 - 8 = \\ &= 3 \cdot (19 - 2 \cdot 8) - 8 = \\ &= 3 \cdot 19 - 7 \cdot 8 = \\ &= 3 \cdot 19 - 7 \cdot (160 - 8 \cdot 19) = \\ &= 59 \cdot 19 - 7 \cdot 160 \end{aligned}$$

Es ergibt sich $d = 59$.

3.3 Ver- und Entschlüsselung

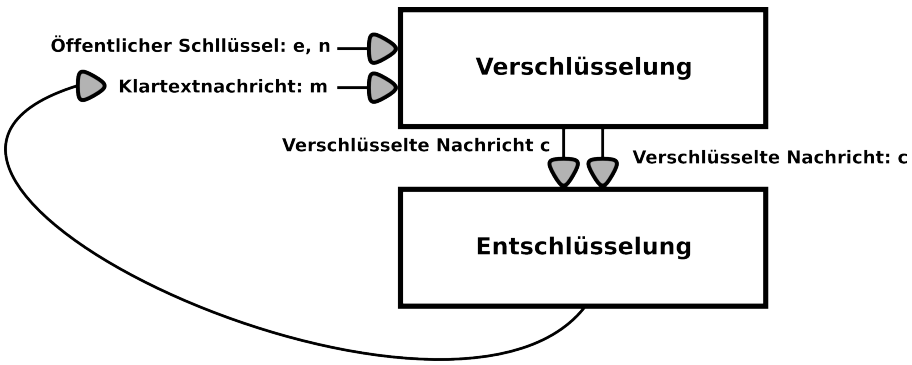

Abbildung 4: Ein- und Ausgangswerte bei der Ver- und Entschlüsselung

Zur Verschlüsselung einer Nachricht wird jetzt der öffentliche Schlüssel der Gegenseite e, n verwendet. Die Klartextnachricht $m \in \mathbb{N}$ kann wie folgt in die verschlüsselte Nachricht c umgerechnet werden[11]:

[11] Siehe beispielsweise [7], [5, S.17] oder [2, S.96]

$$c = m^e \bmod n$$

Das Entschlüsseln verläuft ähnlich einfach:

$$m = c^d \bmod n$$

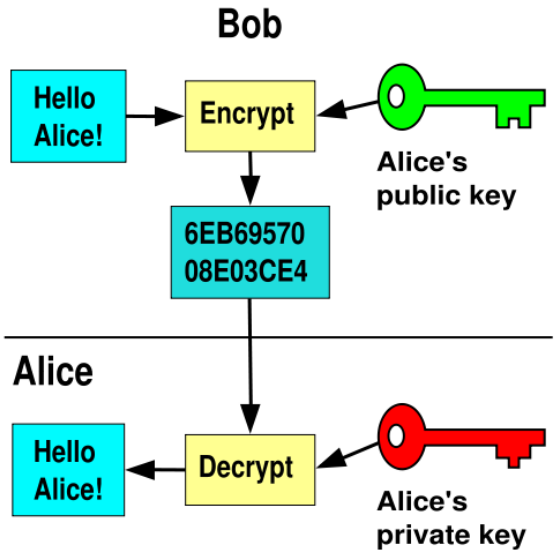

Abbildung 5: Das Ver- und Entschlüsseln einer Nachricht (aus [9])

Beispiel: Wir verschlüsseln die Zahl $m = 16$...

$$\begin{aligned}
c &= m^e \bmod n = \\
&= 16^{19} \bmod 187 = \\
&= ((16^4 \bmod 187)^4 \cdot (16^3 \bmod 187)) \bmod 187 = \\
&= ((65536 \bmod 187)^4 \cdot (4096 \bmod 187)) \bmod 187 = \\
&= (86^4 \cdot 169) \bmod 187 = \\
&= 152
\end{aligned}$$

...und entschlüsselen wieder:

$$
\begin{aligned}
m &= c^d \bmod n = \\
&= 152^{59} \bmod 187 = \\
&= ((((152^4 \bmod 187)^4 \bmod 187)^3 \bmod 187) \cdot 152) \bmod 187 = \\
&= (((137^4 \bmod 187)^3 \bmod 187) \cdot 152) \bmod 187 = \\
&= ((86^3 \bmod 187) \cdot 152) \bmod 187 = \\
&= (69 \cdot 152) \bmod 187 = \\
&= 16
\end{aligned}
$$

Wir erhalten das korrekte Ergebnis $m = 16$.

3.4 Übertragen von ganzen Texten

Zur Übertragung von Nachrichten ist es notwendig, alle Zeichen in Zahlen umzusetzen[12]. Als Ergebnis erhält man mehrere Zahlen, die einzeln verschlüsselt werden können. Jede Nachricht m muss kleiner n sein, was bei typischen Werten von n mit mehr als 100 Stellen natürlich kein Problem darstellt, wenn man nur 8-Bit-Werte (d.h. $m < 256$) verschlüsselt. Aus Effizienzgründen lassen sich mehrere der zu verschlüsselnden Zahlen in eine geringere Zahl von Blöcken zusammenfassen[13].

3.5 Signieren

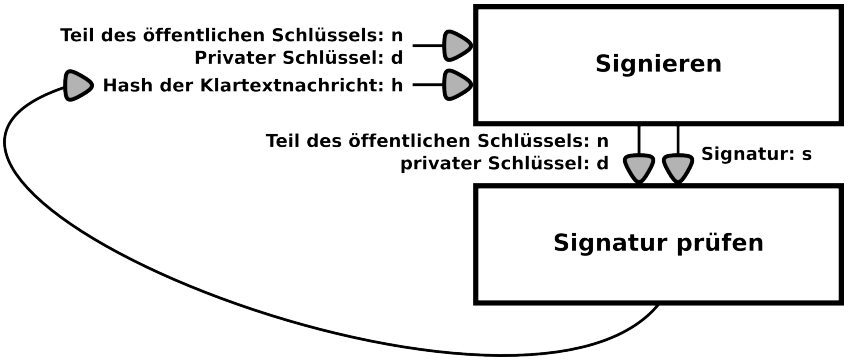

Abbildung 6: Ein- und Ausgangswerte bei dem Signieren

RSA kann zum Signieren verwendet werden. Das gibt dem Empfänger einer Nachricht die Möglichkeit, eindeutig zu überprüfen, ob eine Nachricht von einer

[12] Hierfür existiert der ASCII-Standard, den Computer verwenden (können), um Zeichen 8-Bit-Zahlen zuzuordnen.

[13] [1, S.163ff]

Person versendet wurde, die im Besitz des privaten und öffentlichen Schlüssels des Kommunikationspartners ist – wurde vom Empfänger also in der Vergangenheit einmal die Zuordnung zwischen öffentlichem Schlüssel und einer bestimmten Person hergestellt, kann er bei allen signierten und von ihm verifizierten Nachrichten sicher sein, dass er mit der Person kommuniziert, mit der er kommunizieren möchte[14].

Die Signatur der Variable h wird einfach berechnet...

$$s = h^d \bmod n$$

...und verifiziert:

$$h = s^e \bmod n$$

Beispiel: Die Signatur von $h = 4$ lautet:

$$\begin{aligned}
s &= h^d \bmod n = \\
&= 4^{59} \bmod 187 = \\
&= ((((4^8 \bmod 187)^2 \bmod 187)^3 \bmod 187) \cdot (4^8 \bmod 187) \cdot 4^3) \bmod 187 = \\
&= ((((65535 \bmod 187)^2 \bmod 187)^3 \bmod 187) \cdot (65535 \bmod 187) \cdot 64) \bmod 187 = \\
&= ((((86^2 \bmod 187)^3 \bmod 187) \cdot 86 \cdot 64) \bmod 187 = \\
&= ((((7396 \bmod 187)^3 \bmod 187) \cdot 5504) \bmod 187 = \\
&= (86 \cdot 5504) \bmod 187 = \\
&= 47
\end{aligned}$$

Zur Überprüfung wird wieder h bestimmt und mit dem tatsächlich empfangenen h verglichen.

$$\begin{aligned}
h &= s^e \bmod n = \\
&= 47^{19} \bmod 187 = \\
&= (((47^4 \bmod 187)^4 \bmod 187) \cdot (47^3 \bmod 187)) \bmod 187 = \\
&= (((4879681 \bmod 187)^4 \bmod 187) \cdot (103823 \bmod 187)) \bmod 187 = \\
&= ((103^4 \bmod 187) \cdot 38) \bmod 187 = \\
&= ((112550881 \bmod 187) \cdot 38) \bmod 187 = \\
&= (69 \cdot 38) \bmod 187 = \\
&= 2622 \bmod 187 = \\
&= 4
\end{aligned}$$

[14] Außer der private Schlüssel des Nachrichtensenders ist in fremde Hände gefallen.

4 Anwendungsmöglichkeiten

4.1 Hybride Verschlüsselung

RSA ist in der Praxis für die meisten Anwendungen (im Rahmen der momentanen technischen Möglichkeiten) ungeeignet, weil es aufwendig anzuwenden ist. (Auch für einen Computer, der häufig Megabyte bis Terabyte an Daten verschlüsseln muss, mit erheblich längeren Schlüsseln als in dem verwendeten Beispiel.)

Symmetrische Verschlüsselung kann an sich eine hohe Sicherheit aufweisen, wenn sichergestellt ist, dass der Schlüssel nicht in falsche Hände gerät, und ist je nach Verfahren schnell anzuwenden. Asymmetrische Verschlüsselung ist mit längeren Nachrichten sehr aufwendig zu berechnen, vermeidet aber das Problem der Schlüsselübertragung.

Hybride Verschlüsselung ist ein Verfahren, das symmetrische und asymmetrische Verschlüsselung kombiniert: Der Schlüssel einer mit einem symmetrischen Verfahren gesicherten Nachricht wird asymmetrisch verschlüsselt übertragen. So erreichen gängige Verschlüsselungsverfahren (wie z.B. das sehr verbreitete SSL, genutzt für Online-Banking, sichere Kommunikation mit anderen Computersystemen im Allgemeinen, PGP, usw.) eine kurze Berechnungszeit bei gleichzeitig hoher Sicherheit. Erstmals umgesetzt hat die hybride Verschlüsselung 1991 Phil Zimmermann[15], der Erfinder von PGP, da die Verschlüsselung einer längeren Textnachricht mit RSA auf damaligen Computersystemen mehrere Minuten benötigte, und sich somit für den Endanwender als unpraktisch darstellte.

4.2 Signieren

Eine weitere Anwendung von RSA ist das Signieren von Nachrichten. Durch die Signatur von z.B. E-Mails mit RSA kann der Empfänger sicherstellen, dass die Nachricht von einer Person ausging, die im Besitz eines bestimmten privaten Schlüssels ist. Beim Versand von rechtskräftigen Dokumenten per E-Mail ist in Deutschland eine qualifizierte elektronische Signatur Pflicht. Dies ist eine weitverbreitete Anwendung von Signaturen auf Basis von Public-Key-Verfahren.

Um die Signatur kurz zu halten wird zum Signieren einer ganzen Nachricht selbige durch ein Einwegverschlüsselungsverfahren[16] auf eine definierte Länge gebracht, und hierüber die Signatur gebildet[17].

[15] [3, S.359]
[16] Siehe Kapitel 2.4 auf Seite 5.
[17] [5, S.18]

4.3 Schlüsselaustausch und Identitätsverifikation in der Praxis

Für den Austausch von öffentlichen Schlüsseln zwischen Computer-Anwendern werden sogenannte **Keyserver** verwendet, denen jeder Benutzer seinen öffentlichen Schlüssel zur Verfügung stellen kann[18]. Zur Verifikation der Identität des Schlüsselerzeugers können andere Benutzer dann dem Keyserver mitteilen, dass sie die Zuordnung Schlüssel $\leftrightarrow$ Person bestätigen können[19]. Bei Kommunikation per SSL (erwähnt in Kapitel 4.1 auf der vorherigen Seite), mit dem u.a. die Datenübertragung zu einem Online-Banking-Server gesichert wird, werden die öffentlichen Schlüssel direkt zwischen zwei Computern ausgetauscht, und die Identität des Servers durch eine allgemein anerkannte Zertifizierungsstelle bestätigt.

[18] [6, S.95]
[19] [6, S.100]

5 Mögliche Attacken

5.1 Angriffspunkte

Die meisten Sicherheitsprobleme bei den verschiedenen RSA-Implementationen sind nicht theoretischer Natur, sondern verursacht durch umsetzungsbedingte Details[20].

Im Allgemeinen wird das Brechen von RSA mit dem Problem große Zahlen $n = p \cdot q$ schnell zu faktorisieren gleichgesetzt. Eine andere Möglichkeit RSA zu brechen ist es, m direkt aus $m^e \bmod n = c$ abzuleiten. Wurzeln modulo n zu ziehen ist bei ausreichend großem n jedoch enorm aufwendig, und ein besserer Weg ist nicht bekannt[21].

Trotzdem ist unklar, ob die Sicherheit von RSA in Zukunft auf einen Schlag zerstört werden könnte – weil ein effizienter Faktorisierungsalgorithmus entwickelt wird, Quantencomputer ausreichende Kapazität erreichen[22] oder sich die Geschwindigkeit momentaner Computer vervielfacht (wovon auszugehen ist).

5.2 Primfaktorzerlegung

Zum Beispiel mit dem Faktorisierungsalgorithmus von Fermat, beschrieben in [1, S.68] und implementiert im beiliegenden Softwarebeispiel in der Datei *util.c*, oder aber ausgefeilteren Algorithmen, wie dem quadratischen Zahlensieb, lässt sich die Zahl $n = p \cdot q$ faktorisieren. Dies ist bei gut gewählten, sehr großen Zahlen p, q enorm rechenaufwendig, worauf die Sicherheit des RSA-Algorithmus basiert.

5.3 Die richtige Wahl von p, q

1995 haben zwei amerikanische Studenten eine weit verbreitete Variante von RSA gebrochen – weil die Primzahlen p, q schlecht gewählt waren.

Die Zahl n ist mit dem Algorithmus von Fermat sehr schnell faktorisierbar, wenn p, q nahe beeinander liegen, d.h. $|p - q|$ einen kleinen Wert hat. Wenn für n etwa r Ziffern angestrebt werden, hat sich für p eine Länge zwischen $4r/10$ und $45r/100$ Stellen als gut geeignet erwiesen. q sollte dann einen Wert nahe $10^r/p$ haben.[23]

$p - 1$, $q - 1$, $p + 1$ und $q + 1$ sollten weiterhin auch große Primzahlen als Faktoren haben, da ansonsten verschiedene Faktorisierungsalgorithmen $p \cdot q$ erheblich schneller faktorisieren können.

[20] Weitere Informationen: `http://en.wikipedia.org/wiki/RSA#Timing_attacks`
[21] [1, S.168]
[22] [3, S.398]
[23] [1, S.168]

6 Eigene RSA-Implementation

6.1 Aufbau

Zur Veranschaulichung des Verfahrens wurde ein Gerät entwickelt, das den geheimen RSA-Schlüssel nach der Eingabe von p, q und e berechnet und damit eine vom Benutzer eingebene Klartextnachricht m verschlüsselt. Weiterhin kann das Gerät sowohl bei Kenntnis des privaten Schlüssels, als auch bei Unkenntnis dessen, eine verschlüsselte Nachricht c wieder in eine Klartextnachricht umrechnen. In letzterem Fall wird dazu aus dem öffentlichen Schlüssel durch Faktorisierung p und q ermittelt, und dann zuerst der private Schlüssel errechnet.

6.2 Elektronik

Die Elektronik der Verschlüsselungseinheit besteht aus einem fertig erstandenen Prozessormodul[24] mit ATMEGA128-Prozessor[25] und 1,5″-TFT, das mit einem Tastenfeld erweitert und in ein Gehäuse eingebaut wurde.

6.3 Software

Die über 1000 Zeilen lange Software ist bis auf drei Funktionen aus [10], [12] und [11] komplett selbstgeschrieben. Sie ist in der Programmiersprache C entwickelt worden und unterstützt für p, q, e und m nur Werte kleiner 128. Dass in der Praxis Zahlen mit etwa 200 Stellen verwendet werden, und dann natürlich auch das Brechen der Verschlüsselung keine Sache weniger Sekunden mehr ist, schränkt den Wert dieser Umsetzung auf den eines reinen Beispiels zur Vorgehensweise ein. Es wird auf Eingabefehler (Werte zu groß, p, q keine Primzahlen, e nicht teilerfremd zu $\varphi(n)$) hingewiesen und beim Brechen der Verschlüsselung erkannt, wenn der öffentliche Schlüssel nicht den Anforderungen eines RSA-Schlüssels genügt. Beim Erzeugen eines neuen Schlüssels wird ein valides e vorgeschlagen, aber die Eingabe eines anderen Werts erlaubt. Eine Berechnung von e auf Basis von d und n ist nicht vorgesehen, eine Berechnung von n anhand von e und d ist unmöglich.

Der gesamte nicht-technikbezogene Quelltext findet sich in den Dateien *rsa.c* und *util.c*, in denen direkt der beschriebene Algorithmus zur Schlüsselerzeugung, dem Ver- und Entschlüsseln, dem Signieren und dem Brechen implementiert ist.

[24] Mehr Informationen siehe `http://display3000.com`
[25] Datenblatt auf beiliegender CD)

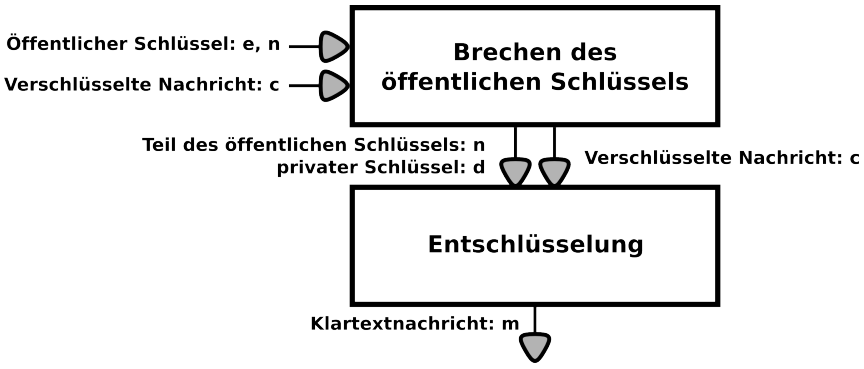

Abbildung 7: Vorgehensweise zum Entschlüsseln einer Nachricht, ohne den privaten Schlüssel zu kennen

6.4 Ergebnis

Abbildung 8: Foto des fertig aufgebauten Geräts

7 Anhänge

7.1 Datenblatt ATMEGA128

... Weitere 393 Seiten auf der beiliegenden CD.

Literatur

[1] The Mathemathics of Ciphers, Coutinho, A K Peters, Ltd., 1999, ISBN 1-56881-082-2

[2] Mathematik für Informatiker: Diskrete Mathematik und Lineare Algebra, Teschl, Teschl, eXamen.press, 2007, ISBN 1614-5216

[3] Geheime Botschaften: Die Kunst der Verschlüsselung von der Antike bis in die Zeiten des Internet, Singh, dtv, 2008, ISBN 978-3-423-33071-8

[4] Kryptologie, Eine Einführung in die Wissenschaft vom Verschlüsseln, Verbergen und Verheimlichen, Beutelspacher, Vieweg Verlag, 2002, ISBN 3-528-58990-6

[5] Moderne Verfahren der Kryptographie, Von RSA zu Zero-Knowledge, Beutelspacher, Schwenk, Wolfenstetter, Vieweg Verlag, 2006, ISBN 3-8348-0083-X

[6] Praktische Kryptographie unter Linux, Lars Packschies, Open Source Press, 2005, ISBN 3-937514-06-6

[7] Kryptographie – Das Hüten von Geheimnissen, Jahnke, Spektrum 1/08, S.27–30

[8] `http://www.cacr.math.uwaterloo.ca/hac/about/chap9.pdf` (22.01.2009 22:33)

[9] `http://de.wikipedia.org/wiki/Asymmetrisches_Kryptosystem` (15.08.2008 09:45)

[10] `http://en.wikipedia.org/wiki/Euclidean_algorithm` (05.08.2008 01:35)

[11] `http://www.math.uu.se/~astrombe/stfz/modpow.c` (09.08.2008 21:10)

[12] `http://snippets.dzone.com/posts/show/4256` (10.08.2008 00:28)

Eine Kopie der Quellen aus dem Internet liegt zusammen mit dem entwickelten Programmcode und Datenblättern auf CD bei.

Abbildungsverzeichnis

Erklärung

Hiermit erkläre ich, dass ich die Facharbeit ohne fremde Hilfe angefertigt habe, und nur die im Literaturverzeichnis angeführten Quellen und Hilfsmittel verwendet habe.

Schwabach, den 29.01.2009

. . . . . .
(Unterschrift)