

# **Informatikseminar WS 08/09**

## **Thema: RSA-Verschlüsselung**



**Eingereicht von: Daniel Wischer**  
**Triftheide 9**  
**22547 Hamburg**  
**[inf6501@fh-wedel.de](mailto:inf6501@fh-wedel.de)**

**Erarbeitet im: 6. Semester**

**Abgegeben am: 7. Dezember 2008**

Inhaltsverzeichnis

1. Einleitung und Geschichte der RSA-Verschlüsselung..... 3

2. Asymmetrische vs. Symmetrische Verschlüsselung..... 3

3. Mathematische Grundlagen: Einwegfunktionen..... 4

4. Bestandteile der RSA-Verschlüsselung und Erstellung der Schlüssel

    4.1 Bestandteile des RSA-Verfahrens..... 4

    4.2 Berechnung der RSA-Bestandteile..... 5

    4.3 Vollständiges Beispiel der Schlüsselberechnung..... 8

5. Ablauf der Kommunikation mit dem RSA-Verfahren

    5.1 Verschlüsselung einer Nachricht..... 10

    5.2 Entschlüsselung einer Nachricht..... 11

    5.3 Beispiel einer Kommunikation..... 11

6. Sicherheit und Korrektheit der RSA-Verschlüsselung

    6.1 Korrektheit der RSA-Verschlüsselung..... 12

    6.2 Sicherheit der RSA-Verschlüsselung..... 14

7. Digitale Signatur mit dem RSA-Verfahren..... 15

8. Praktische Anwendungen des RSA-Verfahrens..... 17

9. Literaturverzeichnis..... 18

## **1. Einleitung und Geschichte der RSA-Verschlüsselung**

Das RSA-Verfahren wurde im Jahr 1977 von den Mathematikern Ron Rivest, Adi Shamir und Len Adleman entwickelt. Es galt damals als das erste asymmetrische Verschlüsselungsverfahren das entwickelt wurde. In Wirklichkeit wurde schon im Jahr 1970 ein asymmetrisches Verschlüsselungsverfahren vom britischen Geheimdienst entwickelt, aus Sicherheitsgründen aber nie veröffentlicht.

Dadurch konnte das RSA-Verfahren 1983 patentiert werden. Dieses Patent ist am 21. September 2008 ausgelaufen. Dadurch sind ab diesem Datum alle Implementierungen des RSA-Verfahrens frei von Lizenzgebühren. Zur Anwendung kam das RSA-Verfahren z.B. beim PGP-Verfahren (Pretty Good Privacy), wurde aber mittlerweile durch andere Algorithmen abgelöst.

## **2. Asymmetrische vs. Symmetrische Verschlüsselung**

Bevor die ersten asymmetrischen Verfahren entwickelt wurden, gab es nur symmetrische Verschlüsselungsverfahren. Bei den symmetrischen Verfahren gibt es nur einen Schlüssel, der sowohl für die Verschlüsselung als auch für die Entschlüsselung verwendet wird. Dies ist zwar günstig für die Laufzeit der Verschlüsselung, allerdings entsteht dadurch das Problem, dass der Schlüssel beiden Teilnehmern der verschlüsselten Übertragung bekannt sein muss. Es muss demnach zuerst ein sicherer Übertragungsweg für den Schlüssel gefunden werden um eine symmetrische Verschlüsselung nutzen zu können. Dies ist für wenige Teilnehmer noch möglich, jedoch erhöht sich der Verteilungsaufwand mit steigender Teilnehmerzahl enorm.

Für  $n$  Teilnehmer müssen  $n \cdot (n-1) / 2$  Schlüssel übertragen werden.

Das ergibt bei einer Teilnehmerzahl von  $n=100$  schon 4950 Schlüssel, die sicher übertragen werden müssen.

Im Gegensatz dazu wurden die asymmetrischen Verschlüsselungsverfahren entwickelt, sogenannte „Public-Key Systeme“. Bei diesen Verfahren gibt es einen öffentlichen und einen privaten Schlüssel. Dabei wird der öffentliche Schlüssel zur Verschlüsselung der Nachricht verwendet und der private Schlüssel zum Entschlüsseln der Nachricht.

Dies bringt den Vorteil, dass nur noch die öffentlichen Schlüssel irgendwo öffentlich gespeichert werden müssen und dass dafür keine sicheren Übertragungswege genutzt werden müssen.

Allerdings geht dies meist zulasten der Laufzeit, da in der Regel aufwändigere Algorithmen für die Ver- und Entschlüsselung genutzt werden müssen.

#### **4. Mathematische Grundlagen: Einwegfunktionen**

Für die asymmetrische Verschlüsselung werden Funktionen benötigt, bei denen die Berechnung des Funktionswertes einfach ist, die Umkehrung allerdings schwer berechenbar ist.

Diese Eigenschaften bieten Einwegfunktionen.

Als Beispiel nehmen wir hier ein Telefonbuch. Zu einem bekannten Namen lässt sich schnell die zugehörige Telefonnummer finden. Ist allerdings nur die Telefonnummer bekannt, ist es schwierig und langsam den zugehörigen Namen zu finden.

Ein Spezialfall der Einwegfunktionen sind so genannte Trapdoor-Einwegfunktionen. Bei diesen Einwegfunktionen gibt es, wie der Name schon erwarten lässt, eine Falltür durch die sich die Umkehrfunktion berechnen lässt, wenn man die Falltür kennt.

Das Potenzieren mod  $n$  ist eine solche Trapdoor-Einwegfunktion.

#### **4. Bestandteile der RSA-Verschlüsselung und Berechnung der Schlüssel**

##### 4.1 Bestandteile der RSA-Verschlüsselung

Wie bei Public-Key-Systemen üblich, besitzt die RSA-Verschlüsselung einen öffentlichen und einen privaten Schlüssel. Zusätzlich zu diesen beiden Werten werden noch andere Werte berechnet, die hier kurz erläutert werden:

- $p, q$ : Zwei zufällig gewählte, möglichst große Primzahlen
- $n$ : Der RSA-Modul. Bestandteil des öffentlichen Schlüssels
- $e$ : Encrypt: Bestandteil des öffentlichen Schlüssels
- $d$ : Decrypt: Bestandteil des privaten Schlüssels
- $\Phi(n)$ : Eulerzahl zum RSA-Modul

Der öffentliche Schlüssel wird nun aus dem Paar  $e$  und  $n$  und der private Schlüssel aus dem Paar  $d$  und  $n$  gebildet.

Public-Key:         $(e, n)$   
Private-Key:        $(d, n)$

## 4.2 Berechnung der RSA-Bestandteile

Hier folgt zuerst eine Erklärung der Schritte zum Berechnen der RSA-Bestandteile. Im Anschluss finden Sie ein anschauliches Beispiel.

### **1.Schritt: Wählen der Primzahlen**

Zuerst wählt man für die Berechnung der Schlüssel zwei zufällige, möglichst große Primzahlen. Dabei wächst die Sicherheit der Verschlüsselung mit der Größe der gewählten Primzahlen. Heutzutage wählt man Primzahlen mit mindestens 100-350 Dezimalstellen.

Eine 1024-stellige Binärzahl entspricht dabei einer Dezimalzahl mit 350 Dezimalstellen.

### **2.Schritt: Berechnen des RSA-Moduls**

Zum Berechnen des RSA-Moduls werden die im Schritt 1 gewählten Primzahlen multipliziert. Das Ergebnis ist der RSA-Modul, der Bestandteil des öffentlichen und des privaten Schlüssels ist. Außerdem bestimmt er die maximale Größe der zu verschlüsselnden Nachricht, denn die Nachricht muss numerisch echt kleiner als der RSA-Modul sein, ansonsten funktioniert die RSA-Verschlüsselung nicht mehr.

Dies lässt sich allerdings immer erreichen, indem die Nachricht in kleinere Teile zerlegt wird.

### **3.Schritt: Berechnung der Eulerzahl zum RSA-Modul**

Als nächstes muss die Eulerzahl zum RSA-Modul berechnet werden. Die Eulerzahl einer Zahl  $x$  ist dabei die Anzahl der teilerfremden Zahlen die kleiner sind als  $x$ .

Beispiel:  $\Phi(8) = 4$ , weil 1, 3, 5 und 7 teilerfremd zu 8 sind

Da der RSA-Modul das Produkt von 2 Primzahlen ist, ist die Berechnung der Eulerzahl von  $n$  sehr einfach. Bei Primzahlen gilt, dass alle Zahlen außer der 1 teilerfremd zur Primzahl sind.

Daraus ergibt sich die Eulerzahl einer Primzahl zu  $\Phi(p) = (p - 1)$

Dies führt wiederum dazu, dass die Eulerzahl des RSA-Moduls das Produkt der Eulerzahlen der beiden Primzahlen  $p$  und  $q$  ist.

$$\Phi(n) = (p - 1) * (q - 1)$$

#### 4.Schritt: Berechnen des öffentlichen Schlüssels e

Der öffentliche Schlüssel e wird berechnet, indem man eine zu  $\Phi(n)$  teilerfremde Zahl wählt. Das bedeutet das der  $ggT(\Phi(n), e) = 1$  sein muss.

Wir wählen also eine beliebige Zahl, von der wir denken, dass sie teilerfremd zu  $\Phi(n)$  ist und berechnen mit dem euklidischen Algorithmus ob der  $ggT = 1$  ist. Ist dies der Fall, haben wir gut gewählt und unsere Zahl e gefunden. Sollten wir einen  $ggT > 1$  erhalten, teilen wir unsere gewählte Zahl e durch den  $ggT$  den wir errechnet haben (  $e' = e / ggT(\Phi(n), e)$  ). Nun erhalten wir eine garantiert zu  $\Phi(n)$  teilerfremde Zahl. Sollte die Zahl e durch die Division zu klein geworden sein um noch ausreichende Sicherheit zu gewährleisten, wählen wir eine neue teilerfremde Zahl und durchlaufen erneut den Algorithmus, da das Berechnen des  $ggT$  durch den euklidischen Algorithmus effizient lösbar ist. Wichtig bei diesem Schritt ist, beim Berechnen des  $ggT$  sich die Koeffizienten im Laufe der Berechnung zu speichern, da diese in einem späteren Schritt zu einer deutlich leichteren Berechnung führen.

Beispiel:

Berechnung des  $ggT(24,11)$

- 1.Schritt:  $24 = 2 * 11$  Rest 2
- 2.Schritt:  $11 = 5 * 2$  Rest 1
- 3.Schritt:  $2 = 2 * 1$  Rest 0

a	b	q
24	11	2
11	2	5
2	1	2
1	0	

Bei der Berechnung des  $ggT$  werden die Koeffizienten in der Spalte q der Tabelle eingetragen. a und b sind dabei die beiden Werte, dessen  $ggT$  in diesem Schritt berechnet wird.

**5.Schritt: Berechnen des privaten Schlüssels d**

Der private Schlüssel der RSA-Verschlüsselung muss folgende Gleichung erfüllen:

$$e * d \text{ MOD } \Phi(n) = 1$$

Das bedeutet, das d das modular Inverse Element zu  $\Phi(n)$  ist. An diesem Punkt benötigen wir die aus Schritt 4 berechneten Koeffizienten wieder, da sich mit dem Erweiterten Euklidischen Algorithmus auch effizient das modular Inverse Element berechnen lässt.

Beispiel:

Nehmen wir also das Beispiel aus Schritt 4 und erweitern es um  $\Phi(n) = 24$  Wir wollen demnach das inverse Element zu 11 bezüglich 24 berechnen. Wir können jetzt durch Rückwärtsrechnen mit dem Erweiterten Euklidischen Algorithmus das modular inverse berechnen.

Es soll in jeder Zeile gelten:  $ggT(a,b) = s*a + t*b$   
 Daraus folgt, dass die erste Zeile die Werte  $s = 1$  und  $t = 0$  erhält.  
 Die folgenden Zeilen lassen sich nun wie folgt berechnen:  
 $s = t_{alt}$   
 $t = s_{alt} - q * t_{alt}$

Erweitert man nun die Tabelle aus dem Beispiel aus Schritt 4 um die Spalten s und t und trägt in die unterste Zeile die besagten Werte  $s = 1$  und  $t = 0$  ein, lassen sich die restlichen Zeilen einfach und effizient berechnen.

<b>a</b>	<b>b</b>	<b>q</b>	<b>s</b>	<b>t</b>
24	11	2	-5	<b>11</b>
11	2	5	1	-5
2	1	2	0	1
1	0		1	0

In der oberen rechten Ecke der Tabelle finden wir nun das modular Inverse Element zu b bezüglich a.

Nach diesem Schritt sind alle Bestandteile der RSA-Verschlüsselung berechnet.

4.3 Vollständiges Beispiel der Schlüsselberechnung

**1. Wählen der Primzahlen**

Zur Veranschaulichung der Berechnung wählen wir kleine Primzahlen. Dies ist in der Praxis aus Sicherheitsgründen nicht sinnvoll.

$$p = \mathbf{13}$$

$$q = \mathbf{23}$$

**2. Berechnen des RSA-Moduls**

Der RSA-Modul ergibt sich aus den zuvor gewählten Primzahlen:

$$n = p * q = 13 * 23 = \mathbf{299}$$

**3. Berechnen der Eulerzahl zu n**

$$\Phi(n) = (p - 1) * (q - 1) = 12 * 22 = \mathbf{264}$$

**4. Berechnen von e**

Wir wählen e = 5 und testen mit dem Euklidischen Algorithmus ob der ggT = 1

a	b	q
264	5	52
5	4	1
4	1	4
1	0	

Da der ggT(264,5) = 1 ist, haben wir unseren öffentlichen Schlüssel gefunden. Wir merken uns für die weitere Berechnung die berechnete Tabelle.

**Öffentlicher Schlüssel: (299,5)**

## 5. Berechnen von d

Mit dem Erweiterten Euklidischen Algorithmus berechnen wir das inverse Element zu 5 bezüglich 264

<b>a</b>	<b>b</b>	<b>q</b>	<b>s</b>	<b>t</b>
264	5	52	-1	<b>53</b>
5	4	1	1	-1
4	1	4	0	1
1	0		1	0

**Privater Schlüssel: (299, 53)**

## 6. Zusammenfassung

Zu diesem Zeitpunkt sind alle Bestandteile der RSA-Verschlüsselung berechnet.

$$p = 13$$

$$q = 23$$

$$n = 299$$

$$\Phi(n) = 264$$

$$e = 5$$

$$d = 53$$

Public Key: (299, 5)

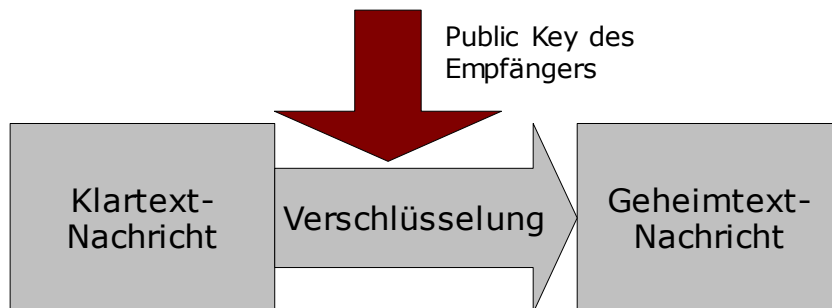
Private Key: (299, 53)

## 5. Ablauf der Kommunikation mit dem RSA-Verfahren

### 5.1 Verschlüsselung einer Nachricht

Ein Teilnehmer am RSA-Verfahren veröffentlicht nun bei einer vertrauenswürdigen Schlüsselstelle seinen Public-Key, den jeder der Zugriff auf die Schlüsselstelle hat, abrufen kann.

Um nun diesem Teilnehmer eine verschlüsselte Nachricht zu senden, nimmt der Sender den Public-Key des Empfängers und verschlüsselt die Nachricht mit eben diesem Public-Key. Sobald der Sender die Nachricht mit dem Public-Key verschlüsselt hat, hat auch er selber keine Chance mehr, die Nachricht zu entschlüsseln um Korrekturen vorzunehmen oder ähnliches. Einmal verschlüsselt ist nur noch der Empfänger in der Lage die Nachricht zu entschlüsseln.



Die Verschlüsselung erfolgt dabei mit der Funktion:

$$\text{Geheimtext} = \text{Klartext}^e \text{ MOD } n$$

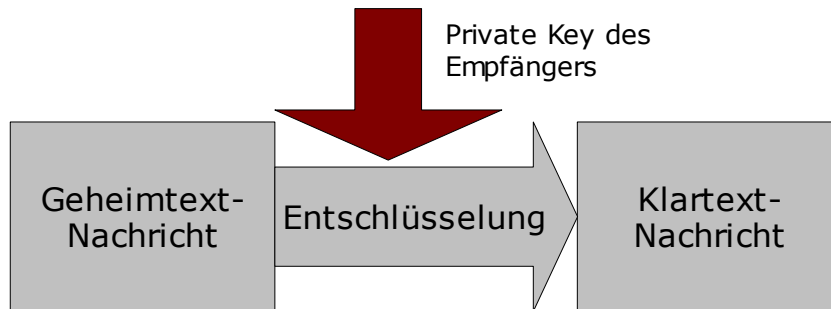
Bei dieser Verschlüsselung sind 2 Randbedingungen zu beachten:

1. Der Klartext und der Geheimtext wird beim Verschlüsseln als Zahl interpretiert. Dies lässt sich allerdings immer erreichen, indem zum Beispiel die Ascii-Werte verschlüsselt werden oder die Zeichenkette binär interpretiert werden.

2. Der Klartext muss vom Zahlenwert immer echt kleiner als  $n$  sein, da sich sonst der Klartext nicht mehr aus dem Geheimtext berechnen lässt. Diese Bedingung lässt sich allerdings immer durch entsprechende Blockung des Textes erzielen (z.B. wortweise oder zeichenweise Verschlüsseln)

5.2 Entschlüsselung einer Nachricht

Der Empfänger der verschlüsselten Nachricht, kann die Nachricht nun mit seinem Private-Key entschlüsseln.



Zur Entschlüsselung benutzt er die gleiche Formel wie bei der Verschlüsselung, nur tauscht er den Public-Key gegen seinen Private-Key aus:

$$\text{Klartext} = \text{Geheimtext}^d \text{ MOD } n$$

Zur Entschlüsselung wird demnach die Umkehrfunktion der Verschlüsselung genutzt. Warum dies funktioniert wird in einem späteren Abschnitt erklärt.

5.3 Beispiel einer Verschlüsselung

Setzen wir nun das Beispiel aus Punkt 4.3 fort und generieren eine Nachricht, die verschlüsselt werden soll.

Klartext:

H	A	L	L	O
72	65	76	76	79

Nun verschlüsseln wir jeden Buchstaben mit der oben angegebenen Verschlüsselungsformel und dem Public-Key des Empfängers:

72	65	76	76	79
$72^5 \text{ MOD } 299$	$65^5 \text{ MOD } 299$	$76^5 \text{ MOD } 299$	$76^5 \text{ MOD } 299$	$79^5 \text{ MOD } 299$
128	195	293	293	157

Es wird also die Nachricht (bzw. 5 kleine Einzelnachrichten) versendet:  
128,195,293,293,157

Der Empfänger wiederum kann nun die Nachricht mit seinem Private-Key wieder entschlüsseln:

128	195	293	293	157
$128^{53} \text{ MOD } 299$	$195^{53} \text{ MOD } 299$	$293^{53} \text{ MOD } 299$	$293^{53} \text{ MOD } 299$	$157^{53} \text{ MOD } 299$
72	65	76	76	79
H	A	L	L	O

## 6. Sicherheit und Korrektheit der RSA-Verschlüsselung

### 6.1 Korrektheit der RSA-Verschlüsselung

Um die Korrektheit der RSA-Verschlüsselung zu zeigen, muss bewiesen werden, dass aus dem verschlüsselten Klartext mithilfe des privaten Schlüssels wieder der Klartext entschlüsselt werden kann.

Es muss also die Verknüpfung der Verschlüsselung und der Entschlüsselung wieder die Nachricht  $m$  ergeben.

Behauptung:  $(m^e)^d \text{ mod } n = m$

Beim Berechnen der Schlüssel wurden  $e$  und  $d$  so definiert, dass folgende Bedingung gilt:

$$e \cdot d \text{ mod } (p-1)(q-1) = 1$$

Es lässt sich außerdem eine Zahl  $j$  um das mod aufzulösen, sodass gilt:

$$e \cdot d = 1 + j \cdot (p-1)(q-1)$$

Nun setzen wir dies in unsere Behauptung ein und stellen folgende Gleichung auf:

$$(m^e)^d = m^{ed} = m^{1+j \cdot (p-1)(q-1)} = m \cdot m^{j \cdot (p-1)(q-1)} = m \cdot (m^{(p-1)})^{(q-1)j}$$

Aus dieser Gleichung lässt sich dann folgende Kongruenz ableiten:

$$(m^e)^d \equiv m * (m^{(p-1)})^{(q-1)j} \equiv m \pmod{p}$$

Nun lassen sich 2 Fälle unterscheiden um die Behauptung zu beweisen (hier wird mod p gerechnet. Somit gilt dieser Teilbeweis nur für p. Analog gilt dieser Beweis auch für q):

### **Beweis durch Fallunterscheidung**

1. **p ist ein Teiler von m:** Wenn p ein Teiler von m ist, ist die Kongruenz auf beiden Seiten 0, da  $m \pmod{p}$  in der Restklasse 0 ist. Dies ist darin begründet, dass eine Zahl  $n \pmod{y}$  immer in der gleichen Restklasse liegt wie  $n^x \pmod{y}$ .

2. **p ist kein Teiler von m:** Wenn p kein Teiler von m ist, lässt sich die Kongruenz der obigen Gleichung aus dem kleinen Fermatschen Satz ableiten. Der kleine Fermatsche Satz sagt:  $a^{(x-1)} \pmod{x} = 1$   
Deswegen lässt sich  $m^{(p-1)} = 1$  setzen und somit gilt  
 $m * 1^{(q-1)j} \equiv m \pmod{p}$

Man sieht also, dass:

$$(m^e)^d = m \pmod{n}, \text{ da } p \text{ und } q \text{ verschiedene Primzahlen sind.}$$

Dies gilt, da  $x = m \pmod{p} \Leftrightarrow x = m \pmod{(y * p)}$  und da p und q Primfaktoren von n sind und oben bewiesen wurde, dass es sowohl für p als auch für q gilt.

Nun lässt sich durch Ersetzen von  $m^e$  durch den Geheimtext c das RSA-Verfahren beweisen:

$$m = c^d \pmod{n}$$

6.2 Sicherheit des RSA-Verfahrens

Die Sicherheit des RSA-Verfahrens besteht darin, dass der geheime Schlüssel  $d$  nicht aus den öffentlichen Schlüsseln und dem Geheimtext in polynomialer Zeit berechnet werden kann.

Eine Strategie um die RSA-Verschlüsselung zu brechen ist, den Modul  $n$ , der öffentlich bekannt ist, in seine Primfaktoren zu zerlegen und somit  $p$  und  $q$  zu erhalten.

Diese Vorgehensweise wird in der Mathematik als das Faktorisierungsproblem bezeichnet. Für die Faktorisierung von großen Zahlen wurde bisher kein effizientes Verfahren gefunden. Dies bedeutet allerdings nicht, dass es kein effizientes Verfahren gibt. Daher kann die RSA-Verschlüsselung auch nicht als sicher garantiert werden, es ist nur mit heutigen Methoden nicht zu entschlüsseln.

Da das Faktorisierungsproblem ein recht verbreitetes mathematisches Problem ist, an dem viele Wissenschaftler forschen, würde ein effizienter Algorithmus schnell bekannt werden und ein Nutzen für die Entschlüsselung von RSA-Nachrichten wäre nicht vorhanden.

Sollte ein Angreifer den Modul  $n$  faktorisiert haben, kann er auf folgende Weise den geheimen Schlüssel  $d$  berechnen.

Zuerst wird die Eulersche Funktion von  $n$  berechnet. Dies ist mit den beiden Primfaktoren  $p$  und  $q$  einfach möglich, da  $\Phi(n) = (p-1)(q-1)$  ist.

Nun kann der Angreifer wie in der Schlüsselerstellung mit dem öffentlichen Schlüssel  $e$  und  $\Phi(n)$  mit dem Erweiterten Euklidischen Algorithmus den Schlüssel  $d$  berechnen.

Beispiel:

Öffentlicher Schlüssel: (299,5)

Der Angreifer faktorisiert 299 ->  $p=13$   $q=23$

Berechnung der  $\Phi(n) = 12 * 22 = 264$

<b>a</b>	<b>b</b>	<b>q</b>	<b>s</b>	<b>t</b>
264	5	52	-1	<b>53</b>
5	4	1	1	-1
4	1	4	0	1
1	0		1	0

Berechneter Schlüssel  $d = 53$

Wie im vorigen Beispiel gesehen, ist das der richtige Schlüssel.

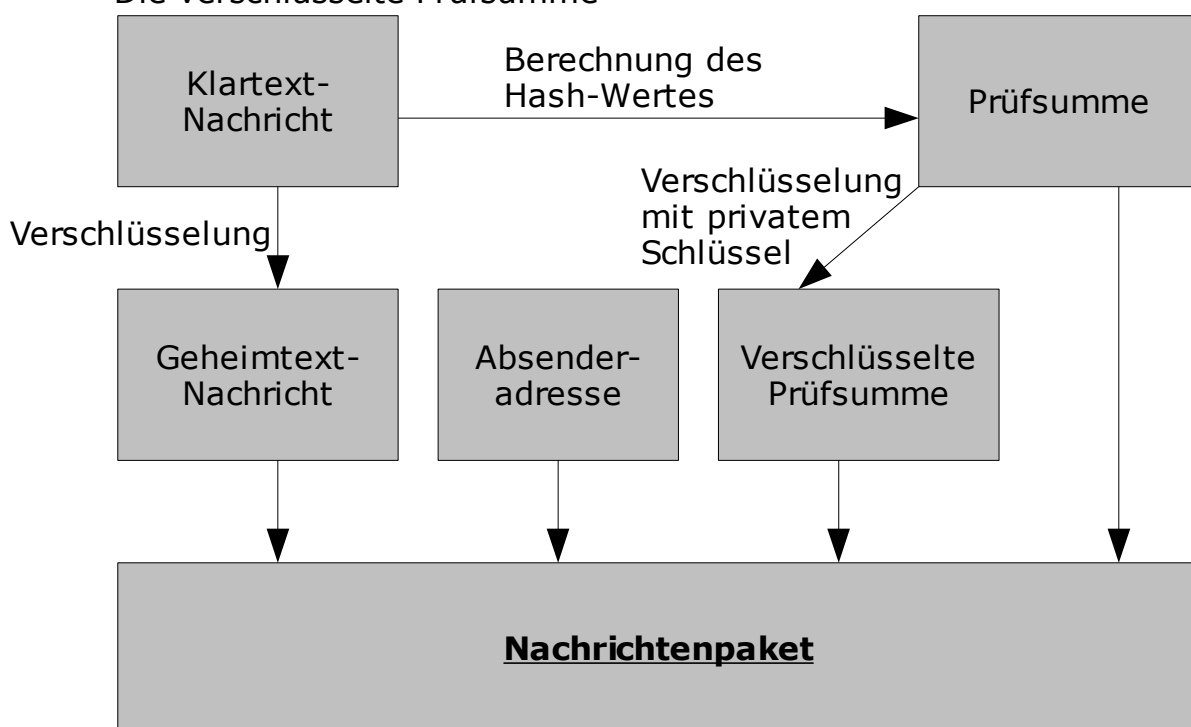
### 7. Digitale Signatur mit dem RSA-Verfahren

Bisher konnte man Nachrichten an Teilnehmer des RSA-Verfahrens verschicken, indem man mit dem öffentlichen Schlüssel des Empfängers verschlüsselt hat. Nun kann zwar nur der Empfänger die Nachricht lesen, er kann jedoch nicht verifizieren, ob die Nachricht tatsächlich vom genannten Absender stammt.

Dieses Problem lässt sich mit dem RSA-Verfahren einfach und effizient lösen. Die Voraussetzung hierfür ist, dass nun beide Teilnehmer einen RSA-Schlüssel für ihre Adresse besitzen müssen.

Für die Digitale Signatur einer Nachricht wird das RSA-Verfahren folgendermaßen angewendet:

1. Absender X schreibt die Nachricht die er verschicken möchte und verschlüsselt sie mit dem öffentlichen Schlüssel des Empfängers Y (Standard RSA Vorgehensweise).
2. Nun berechnet X eine Prüfsumme der geschriebenen Nachricht mit einem Hash Algorithmus (Hierbei sind viele mögliche Hash-Funktionen denkbar) und verschlüsselt diese Prüfsumme mit seinem **privaten** Schlüssel d.
3. Als komplette Nachricht schickt X nun folgendes an Y:
  - Den verschlüsselten Geheimtext
  - Die unverschlüsselte Prüfsumme
  - Die Absenderadresse von X
  - Die verschlüsselte Prüfsumme



4. Empfänger Y holt nun den öffentlichen Schlüssel von X mithilfe seiner Absenderadresse von der Schlüsselstelle und entschlüsselt mit dem öffentlichen Schlüssel von X die Prüfsumme. Nun kann Y die beiden Prüfsummen vergleichen.

Sind beide Prüfsummen identisch kann Y sicher sein, dass die Nachricht tatsächlich von X stammt, da nur X mit seinem privaten Schlüssel verschlüsseln kann.

Als letztes kann Y die eigentliche Nachricht mit dem privaten Schlüssel von Y entschlüsseln.

Dieses Verfahren beruht auf der Austauschbarkeit der Exponenten beim Verschlüsseln mit dem RSA-Verfahren, denn

$$(m^e)^d \Leftrightarrow (m^d)^e$$

Man erhält hierdurch nicht nur eine Verifizierung des Absenders, sondern gleichzeitig eine Fehlerüberprüfung der Nachricht, denn sollten Übertragungsfehler auftreten, würde die Prüfsumme nicht mehr stimmen und der Fehler könnte festgestellt, allerdings nicht korrigiert werden.

#### Beispiel:

Als Basis für dieses Beispiel dient unser zuvor gezeigtes Beispiel zur Kommunikation aus Kapitel 5.3, welches nun um eine digitale Signatur erweitert wird.

Daten des Empfängers Y:

Public Key: (299, 5)

Private Key: (299, 53)

Daten des Senders X:

Public Key: (3233, 17)

Private Key: (3233, 2753)

Wie im Beispiel zuvor versenden wir die Nachricht HALLO:

$m = 7265767679$

$g = 128195293293157$

Zur Bildung der Prüfsumme verwenden wir hier der Einfachheit halber die Quersumme der Nachricht:

Prüfsumme = 62

Diese Prüfsumme wird nun mit dem privaten Schlüssel des Absenders verschlüsselt:  $P_v = 62^{2753} \bmod 3233 = 428$

Die Nachricht besteht nun aus folgenden Bestandteilen:

$g = 128195293293157$

$P = 62$

$P_v = 428$

Absender = X@fh-wedel.de

Empfänger Y entschlüsselt nun die Prüfsumme mit dem öffentlichen Schlüssel von X:  $P = 428^{17} \bmod 3233 = 62$

Diese wird nun mit der unverschlüsselten Prüfsumme verglichen. Da diese übereinstimmen kann Y sicher sein, dass die Nachricht von X stammt und die Nachricht wie gewohnt entschlüsseln.

(Die Entschlüsselung der Nachricht wird hier nicht noch einmal gezeigt, da diese aus Beispiel 5.3 bekannt ist.)

## 8. Praktische Anwendungen des RSA-Verfahrens

In der Praxis wird die RSA-Verschlüsselung meist nicht für das Verschlüsseln großer Datenmengen genutzt. Wie in Punkt 2 bereits erwähnt, sind die asymmetrischen Verschlüsselungsverfahren nicht so effizient wie die symmetrischen Verfahren. Deswegen ist das Haupteinsatzgebiet der RSA-Verschlüsselung das sichere Verschicken des eigentlichen Schlüssels für das symmetrische Verfahren.

Dadurch lässt sich die Effizienz der symmetrischen Verfahren in Verbindung mit dem geringen Aufwand zur Schlüsselverwaltung der asymmetrischen Verfahren nutzen.

Weitere Einsatzgebiete sind z.B.:

- PGP (Pretty Good Privacy)
- EMV (Europay Mastercard Visa)
- S-HTTP

X.Y Literaturverzeichnis

- Johannes Buchmann - Einführung in die Kryptographie (Springer Verlag 2001)
- Kurt-Ulrich Witt - Algebraische Grundlagen der Informatik (Vieweg & Sohn Verlag 2001)
- Wilfried Dankmeier - Grundkurs Codierung (Vieweg & Sohn Verlag 2006)
- RSA-Paper  
<http://people.csail.mit.edu/rivest/Rsapaper.pdf>
- RSA-Erklärung der Universität Wuppertal  
<http://www.matheprisma.uni-wuppertal.de/Module/RSA/index.htm>
- Onlinevorlesung des Hasso-Plattner-Institut Prof. Meinel  
Sicherheit in der Informationstechnik (nichtmehr vorhanden)  
<http://www.tele-task.de>