Schriftliche Zusammenfassung zum Vortrag

„Die Grundlagen der RSA-Verschlüsselung“

von Daniel Metzsch

Freie Universität Berlin
Fachbereich für Mathematik & Informatik
Institut für Mathematik II

Seminar über Algebra und Zahlentheorie (LV 19132)
Univ.-Prof. Dr. Volker Schulze

im Sommersemester 2007

Inhaltsverzeichnis

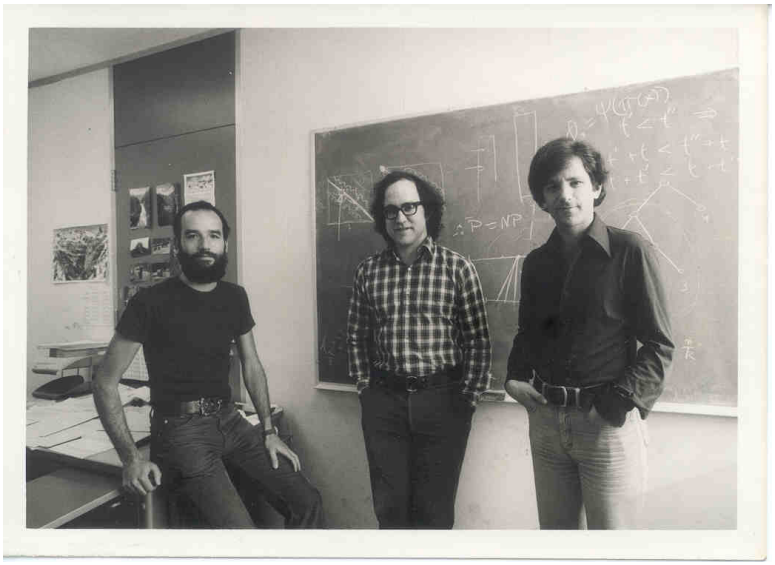
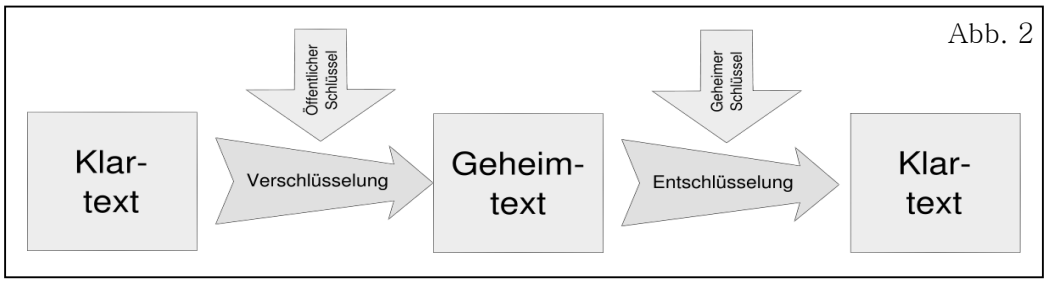


Abb. 1: Adi **S**hamir, Ron **R**ivest und Len **A**dlemen (von links nach rechts)

1. Idee der Public-Key-Verschlüsselung

Man unterscheidet in der Kryptographie zwei große Teilbereiche zur Ver- und Entschlüsselung, die symmetrische und asymmetrische Kryptographie. Bei der symmetrischen Kryptographie benötigen je zwei Teilnehmer einen Schlüssel, der vor allen anderen Teilnehmern im Netzwerk geheim gehalten werden muss. Die symmetrische Kryptographie geht davon aus, dass nicht nur der Empfänger einer Nachricht den passenden Schlüssel besitzt, sondern auch der Sender dieser Nachricht, um den entsprechenden Geheimtext zu produzieren. Gegensätzlich dazu entwickelten W. Diffie und M. Hellmann in ihrer Publikation aus dem Jahr 1976 das Konzept der asymmetrischen- oder Public-Key-Kryptographie. Dabei hat jeder Teilnehmer zwei Schlüssel, einen öffentlichen und einen privaten. Ein potentieller Angreifer hat also auch bei diesem Verfahren von Vornherein mehr Informationen zur Verfügung. Man muss demnach natürlich fordern, dass es dem Angreifer nicht möglich ist, aus dem öffentlichen Schlüssel eines Teilnehmers dessen privaten Schlüssel zu berechnen. Ver- und Entschlüsselung funktionieren gemäß der Abbildung 2. Ein bekanntes Beispiel, das RSA-Verfahren, wird im Folgenden diskutiert.

Abb. 2



2. Das RSA-Verfahren

a. Einführung und Historisches

Das, nach seinen Erfindern Ron Rivest, Adi Shamir und Len Adleman benannte, RSA-Verfahren wurde 1978 entwickelt und publiziert, war das erste Public-Key-Verschlüsselungsverfahren und hat bis heute an seiner Bedeutung nicht verloren. Zunächst noch einige Definitionen und Sätze:

Definition 1: Eine positive ganze Zahl $p>1$ heißt Primzahl, wenn sie sich nur durch p und 1 teilen lässt.

Definition 2: Für zwei ganze Zahlen $m,n \neq 0$ gibt es stets eine größten gemeinsamen Teiler (in Zeichen: ggT(m,n)), d.h. die größte natürliche Zahl, die sowohl m als auch n ohne Rest teilt.

Definition 3: Zwei Zahlen n und m heißen teilerfremd, wenn sie positive ganze Zahlen sind und wenn 1 die einzige Zahl ist, die sowohl n also auch m teilt, anders ausgedrückt: Der größte gemeinsame Teiler von n und m ist 1: $ggT(m,n)=1$.

Definition 4: Für zwei positive Zahlen m und n bezeichnen wir den Rest r bei der Division von m durch n mit $r=m \bmod(n)$, d.h. es ist $0 \leq r < m$, und es gibt eine Zahl k (der ganze Anteil bei der Division von m durch n) mit $m=k \cdot n+r$

Definition 5: $\varphi : IN \to IN$ ist eine Abbildung mit $\varphi(n) = \sum_{\substack{1 \le i < n \\ ggT(i,n)=1}} 1$ und heißt Eulersche φ-Funktion. Außerdem ist φ multiplikativ: $\varphi(x \cdot y) = \varphi(x) \cdot \varphi(y)$ für teilfremde x,y.

Definition 6: Seien $n \in IN, a \in IZ_n$. Dann heißt a Primitivwurzel mod(n), falls Ord(a)= $\varphi(n)$, d.h. (a mod(n)) erzeugt die Gruppe der primen Restklassen, wobei (a mod(n)) prime Restklasse mod(n) heißt $\Leftrightarrow$ ggT(a,n)=1.

Satz 1 (Satz von Euler): Es sei $a \in IZ, n \in IN$. Dann gilt, wenn ggT(a,n)=1, $a^{\varphi(n)} \equiv 1 \bmod(n)$. Ersetzt man die Zahl n durch eine Primzahl p, folgt $a^{\varphi(p)} \equiv a^{p-1} \equiv 1 \bmod(p)$ und damit der **Corollar 1 (Kleiner Satz von Fermat)**. Beweise: [3], S. 124f..

Satz 2 (Chinesischer Restsatz): Seien $m_1, m_2, \ldots, m_n \in IN$ paarweise teilerfremde Zahlen und $a_1, a_2, \ldots, a_n \in IZ$. Dann existiert ein $x \in IZ$, das alle Kongruenzen $x \equiv a_1 \bmod(m_1), \ldots, x \equiv a_n \bmod(m_n)$ erfüllt. x ist $\bmod(m_1, m_2, \ldots, m_n)$ eindeutig bestimmt und mit x auch jeder andere Repräsentant in seiner Restklasse eine Lösung $\bmod(m_1, m_2, \ldots, m_n)$. Beweis: [6] auf S. 18.

Satz 3 (Erweiterter euklidischer Algorithmus, kurz EEA): Seien $a \in IN_0, b \in IN$. Der größte gemeinsame Teiler ggT(a, b) lässt sich als Linearkombination von a und b darstellen: $ggT(a,b) = u \cdot a + v \cdot b \mid u, v \in IZ$. Die Darstellung muss dabei nicht eindeutig sein. Der konstruktive Beweis findet sich z.B. in [6] auf S. 3.

Wir diskutieren im Folgenden wie die Schlüssel erzeugt werden, wie die Ver- und Entschlüsselung funktionieren, einige Aspekte zur Sicherheit und Effizienz des RSA-Verfahrens sowie die sinnvolle Wahl nötiger Parameter.

b. SCHLÜSSELERZEUGUNG

Der erste Schritt besteht darin, zwei Primzahlen p,q zu wählen, sodass die Faktorisierung ihres Produktes n schwierig ist. Das bedeutet insbesondere, dass die Primfaktoren groß genug sein müssen, sodass ein Ausprobieren aller Primzahlen als in Frage kommende Teiler quasi unmöglich ist. Man wählt heute typsicherweise Zahlen der Größenordnung von 512 Bits, d.h. Zahlen der Größe 2^{512}. Der Moduln hat damit die Größe 1024 Bits. Halten wir also fest: $n = p \cdot q$ (I). Weiter wird eine natürliche Zahl e gewählt mit $1 < e < \varphi(n) = (p-1)(q-1) \wedge ggT(e, \varphi(n)) = 1$, wobei φ die eulersche φ-Funktion bezeichnet. Daraus berechnet man eine natürliche Zahl d mit $1 < d < \varphi(n) = (p-1)(q-1) \wedge \; d \cdot e \equiv 1 \bmod(\varphi(n))$. Die Zahl d lässt sich mit dem erweiterten euklidischen Algorithmus (EEA) einfach berechnen. Außerdem existiert sie natürlich, weil $ggT(e, \varphi(n)) = 1$. Betrachten wir das kurz an einem Beispiel:

__Beispiel 1__: Wir wählen p=11 und q=17. Damit ist n=187 und damit ist $\varphi(n) = 160$. e sei gewählt durch e=7. Damit ist nach dem EEA d=23:

Wir wenden den euklidischen Algorithmus auf 160 und 7 an:

$$\left.\begin{aligned} 160 &= 22 \cdot 7 + 6 \\ 7 &= 1 \cdot 6 + 1 \\ 6 &= 6 \cdot 1 + 0 \end{aligned}\right\} \Rightarrow ggT(160,7) = 1 = 7 - 1 \cdot 6 = 7 - 1 \cdot (160 - 22 \cdot 7) = 23 \cdot 7 - 1 \cdot 160$$

Es ist also $1 = 23 \cdot 7 - 1 \cdot 160$ bzw. $1 \equiv 23 \cdot 7 - 1 \cdot 160 \equiv 23 \cdot 7 \bmod(160)$.
$\Rightarrow 1 \equiv 23 \cdot 7 \bmod(160) \Rightarrow d = 23.$

c. VERSCHLÜSSELUNG

Überlegen wir uns nun, wie wir einen Klartext m mit dem RSA-Verfahren verschlüsseln können. m wird verschlüsselt zu $c = m^e \bmod(n)$ (II). Somit kann nun jeder, der den öffentlichen Schlüssel (n,e) kennt, den Klartext m verschlüsseln. Wollen wir nun einen Text verschlüsseln, müssen wir uns sog. Blockchiffren bedienen. Nehmen wir dazu an, das verwendete Alphabet Ω hat N Zeichen und die Zeichen die Zahlen 0,1,2,...,N-1. Setze $k := \lfloor \log_N n \rfloor$. So wird ein Block $m_1,m_2,...,m_k$, $m_i \in \Omega, 1 \le i \le k$ in die Zahl $m = \sum_{i=1}^{k} m_i N^{k-i}$ (III) verwandelt. m wird nun gemäß (II) verschlüsselt. c kann dann zur Basis N geschrieben werden. Die N-adische Entwicklung der Zahl hat dann aber auch die Länge (k+1). Damit ist $c = \sum_{i=0}^{k} c_i N^{k-i}, c_i \in \Omega, 0 \le i \le k$ (IV).

Beispiel 1 (Forts.): *Wir verwenden $\Omega = \{0,a,b,c\}$ mit a=1, b=2, c=3. Unser Alphabet besteht demnach aus N=4 Zeichen mit n=187 und $k=\lfloor \log_4 187 \rfloor = 3$. Das ist die Länge der Klartextblöcke. Wollen wir also den Block „abb" verschlüsseln, entspricht dies zunächst dem String „122" und wg. (III) der Zahl $m=1\cdot 4^2 + 2\cdot 4^1 + 2\cdot 4^0 = 26$. Durch Verschlüsseln erhalten wir* $c = 26^7 \bmod(187) = 104$. *Zur Basis 4 umgeschrieben ergibt sich* $c=1\cdot 4^3 + 2\cdot 4^2 + 2\cdot 4^1 + 0\cdot 4^0 = 104$, *also „abb0".*

d. ENTSCHLÜSSELUNG

Die Tatsache, dass das RSA-Verfahren überhaupt funktioniert, beruht auf diesem Satz 4, den wir im Anschluss beweisen:

Satz 4: Sei (n,e) ein öffentlicher Schlüssel und d der entsprechende private Schlüssel im RSA-Verfahren. Dann gilt $(m^e)^d \bmod(n) = m \quad \forall m \in \mathbb{N} : 0 \le m < n$.

Beweis: Wir wissen nach Definition von d, dass $ed \equiv 1 \bmod(\varphi(n))$. Nun können wir den Modulo-Operator so umschreiben, dass eine ganze Zahl u existiert: $u \cdot \varphi(n) + 1 = e \cdot d$. $\Rightarrow (m^e)^d = m^{ed} = m^{u\cdot\varphi(n)+1}$. Das Ziel ist es nun zunächst, den Satz von Euler anzuwenden. Wir unterscheiden daher zwei Fälle:

1. Fall (ggT(m,n)=1): $m^{u\cdot\varphi(n)+1} \equiv m \cdot m^{u\cdot\varphi(n)} \equiv m \cdot (m^{\varphi(n)})^u \equiv m \cdot 1^u \equiv m \bmod(n)$ (Satz 1).

2. Fall ($ggT(m,n) \neq 1$): Dabei können wiederum zwei Fälle auftreten: m ist ein Vielfaches von n oder m besitzt genau einen Primteiler gemeinsam mit n.

(a) $n \mid m$: Dann ist $m \equiv 0 \bmod(n) \wedge m^{u\cdot\varphi(n)+1} \equiv 0^{u\cdot\varphi(n)+1} \equiv 0 \equiv m \bmod(n)$.

(b) $p \mid m$, $q \nmid m$: Dann ist $m \equiv 0 \bmod(p) \Rightarrow m^{u\cdot\varphi(n)+1} \equiv 0^{u\cdot\varphi(n)+1} \equiv 0 \equiv m \bmod(p)$ und damit $m^{u\cdot\varphi(n)+1} \equiv m^{u\cdot(q-1)(p-1)+1} \equiv (m^{(q-1)})^{u(p-1)} \cdot m \equiv 1^{u(p-1)} \cdot m \equiv m \bmod(q)$ (die vorletzte Kongruenz wg. Cor. 1). Insgesamt gilt $m^{u\cdot\varphi(n)+1} \equiv m^{ed} \equiv m \bmod(n)$. □

Wurde also c gemäß Beispiel 1 berechnet, kann m wegen (II) mittels $m=c^d \bmod(n)$ ermittelt werden. Sehen wir uns das Resultat wieder an unserem Beispiel an:

Beispiel 1 (Forts.): *Wir hatten n=187, e=7 und d=23 gewählt bzw. berechnet. Unser verschlüsselter String war c=104. Wir entschlüsseln dies gemäß der Erkenntnisse aus Satz 3:* $104^{23} \bmod(187) \equiv 26$. *Eine praktische Methode zur Berechnung dieser großen Kongruenz findet man unter [5].*

e. SICHERHEIT DES GEHEIMEN SCHLÜSSELS

Nun haben wir die mathematische Korrektheit des RSA-Algorithmus hinreichend überprüft. Bleibt die Frage zu beantworten, ob das RSA-Verfahren auch die in 1. beschriebenen Public-Key-Eigenschaften erfüllt. Es darf also praktisch nicht möglich sein, aus dem öffentlichen Schlüssel eines Teilnehmers dessen privaten Schlüssel zu berechnen. Wir zeigen im folgenden Satz 5, dass es genauso schwierig ist, die Primfaktoren p,q aus n zu berechnen, wie die Bestimmung des geheimen Schlüssels d aus dem öffentlichen Schlüssel (n,e).

Satz 5[1]: Gegeben sei eine Zahl n, die das Produkt zweier Primzahlen p,q ist. Für jeden Angreifer A sind folg. Dinge äquivalent:

(i) A kann bei Eingabe von n die Primfaktoren p und q von n bestimmen.
(ii) A kann bei Eingabe von n und einer zu $\varphi(n)$ teilerfremden Zahl e eine Zahl d berechnen mit $ed \equiv 1 \bmod(\varphi(n))$.

Beweis:

"(i) $\Rightarrow$ (ii)": Dies folgt mittels des erweiterten euklidischen Algorithmus genauso wie bei der Schlüsselerzeugung des RSA-Verfahrens (s. Satz 3 bzw. Beispiel 1).

"(ii) $\Rightarrow$ (i)": Angenommen der Algorithmus kann bei der Eingabe von n und e die Zahl d bestimmen, für die gilt $ed \equiv 1 \bmod(\varphi(n))$. Aus $ed \equiv 1 \bmod(\varphi(n))$ folgt $ed-1=k\varphi(n)$ für ein $k \in IZ$. Nehmen wir weiter an, der Angreifer kann eine Zahl $a \in IZ_n$ und eine natürliche Zahl s finden mit den Eigenschaften (*) $a^{2s} \equiv 1 \bmod(n)$, (**) $a^s \not\equiv \pm 1 \bmod(n)$. Dann kann man aus (*) schließen, dass $n \mid (a^{2s}-1)$ und wegen der 3. binomischen Formel $n \mid (a^s-1)(a^s+1)$. Aus (**) folgt im Gegensatz dazu, dass n weder (a^s-1) noch (a^s+1) teilt. n teilt aber das Produkt und deswegen ist auch in jedem der Faktoren genau einer der Primfaktoren enthalten. Bestimmt man also den $ggT(n,(a^s-1))$, so findet man einen Primfaktor von n. □
Man kann zeigen, dass sich a und s effektiv finden lassen. Der Algorithmus ist aber probabilistisch, man hat also unter Umständen einige Fehlversuche. Wir sehen in dem nächsten Satz 6, dass in jeder Iteration die Wahrscheinlichkeit dafür, dass ein Primteiler von n gefunden wird, mindestens 0,5 beträgt:

Satz 6: Die Anzahl der zu n primen Zahlen a in der Menge {1,2,...,n-1}, für die a^k mod(p) u. mod(q) eine verschiedene Ordnung haben, ist wenigstens (p-1)(q-1)/2.

Beweis: findet sich in [2] auf S. 120. Der Beweis beruht auf Satz 2, einigen Erkenntnissen über Ordnung von Elementen endlicher Gruppen und Primitivwurzeln.□

Die Erfolgswahrscheinlichkeit des Faktorisierungsalgorithmus ist also wenigstens ½. Nach r Iterationen hat man mit Wahrscheinlichkeit größer $1-(\frac{1}{2})^r$ einen Faktor gefunden. Wie wählen wir nun p,q,d und e sinnvoll? Um die Faktorisierung möglichst schwer zu machen, werden q und p etwa gleichgroß gewählt. Wie eingangs bereits erwähnt, werden die Zahlen in Größenordnung von 512 Bit gewählt, damit ist n eine 1024-Bit-Zahl. Weiterhin sind bestimmte Faktorisierungsalgorithmen bekannt, die p und q bestimmen sollen. Um es diesen möglichst schwer zu machen, sollten p und q zufällig und gleichverteilt gewählt werden, z.B. durch einen Zufallszahlengenerator. Der öffentliche Schlüssel e wird

[1] Um die Public-Key-Eigenschaft vollständig zu zeigen, fehlt hier noch, dass es genauso schwierig ist die Primfaktoren p,q aus n zu berechnen, wie aus n die Zahl $\varphi(n)$ zu berechnen. Dazu mehr im Vortrag oder in [3] auf S. 127 f..

so gewählt, dass die Verschlüsselung effizient möglich ist, die Sicherheit des Algorithmus aber nicht gefährdet ist. e muss dabei mindestens 3 sein, da e=2 wegen $\varphi(n)$ gerade und ggT$(e,\varphi(n))=1$, ausgeschlossen ist. Die Verwendung von e=3 ist aber wegen der sog. Low-Exponent-Attacke problematisch. Näheres dazu in [2] auf S. 122 f.. Dies kann umgangen werden, indem man e groß, aber trotzdem noch effizient genug wählt, z.B. ist $e=2^{16}+1$ typisch.

f. Effizienz des RSA-Algorithmus

Die Verschlüsselung beim RSA-Algorithmus muss gut durchdacht sein. Betrachtet wird eine Exponentiation modulo n. Die Verschlüsselung ist um so effizienter, je kleiner der Exponent ist. Jedoch haben obige Betrachtungen eben gezeigt, dass er auch nicht zu klein sein darf. Bei der Entschlüsselung ergibt sich dasselbe Problem. d hat nun aber dieselbe Größenordnung wie n. Die Verwendung kleiner Schlüsselexponenten ist demnach unsicher. Bei einer Bitlänge von mindestens 512 Bits des RSA-Moduln kann das Entschlüsseln also sehr lange dauern. Unter Verwendung von Satz 2 kann das Verfahren beschleunigt werden. Wollen wir also unter Verwendung des privaten Schlüssels d den Text c entschlüsseln, gilt zunächst $m_p = c^d \bmod(p), m_q = c^d \bmod(q)$ und damit die simultane Kongruenz $m \equiv m_p \bmod(p), m = m_q \bmod(q)$. m ist dabei der eingangs verschlüsselte Klartext. Diese Kongruenz kann mittels des EEA gelöst werden, indem zwei ganze Zahlen y_p und y_q mit $y_p p + y_q q = 1$ berechnet werden. Somit ist dann abschließend die Gleichung $m = (m_q y_p p + m_p y_q q) \bmod(n)$ zu betrachten. Wir betrachten also zum letzten Mal unser Beispiel:

> ***Beispiel 1 (Forts.):*** *Wir beschleunigen unser Verfahren wie gerade eben besprochen:* $m_p = 104^{23} \bmod(11) = 4$ *und* $m_q = 104^{23} \bmod(17) = 9$, $y_p = -3$ *und* $y_q = 2$ *gem. Lösung der diophantischen Gleichung* $11y_p + 17y_q = 1$. *Eingesetzt ergibt das alles* $m = (-9 \cdot 3 \cdot 11 + 4 \cdot 2 \cdot 17) \bmod(187) = -161 \bmod(187) = 26$.

Betrachtet man die benötigte Zeit, so kann durch unser neu gewonnenes Verfahren eine Beschleunigung um den Faktor 4 (s. [2] auf S. 124) erfolgen.

3. Literatur

a. Textnachweise

- Schulze: Skript zur Vorlesung „Einführung in die Algebra und Zahlentheorie“ WS 2002/2003, Freie Universität Berlin **[1]**
- Buchmann: „Einführung in die Kryptographie“ (Kopien) **[2]**
- Beutelspacher, Neumann, Schwarzpaul: „Kryptografie in Theorie und Praxis“ Wiesbaden 2005, Vieweg-Verlag, 1. Auflage **[3]**
- Bartholomé, Rung, Kern: „Zahlentheorie für Einsteiger“ Wiesbaden 2003, Vieweg-Verlag, 4. Auflage **[4]**
- http://www.mathematik.de/mde/information/wasistmathematik/rsa/rsa_potenzen.html **[5]**
- Wolfart: „Einführung in die Zahlentheorie und Algebra“ Wiesbaden 1996, Vieweg-Verlag, 1. Auflage **[6]**

b. Bildnachweise

- Abbildung 1: http://theory.lcs.mit.edu/~rivest/rsa-photo.jpeg
- Abbildung 2: http://de.wikipedia.org/wiki/RSA-Kryptosystem