# Die mathematischen Grundlagen asymmetrischer Kryptographieverfahren

Florian Forster

30. Januar 2003

**Zusammenfassung**

Es gibt nur sehr wenige Methoden, welche für asymmetrische Kryptographie verwendet werden können. Die vorliegende Arbeit klärt zunächst wichtige Begriffe der Kryptographie und gibt einen kurzen Überblick über symmetrische Verfahren. Nach der Vorstellung einiger Grundbausteine werden einige wesentliche asymmetrische Verfahren vorgestellt. Zuletzt werden Anwendungsgebiete besprochen, die über das reine Verschlüsseln hinaus gehen.

# Inhaltsverzeichnis

# 1 Einleitung

Information ist in der heutigen Gesellschaft zu einem sehr wertvollen Gut geworden. Und mit dem Wert wächst auch die Notwendigkeit des Schutzes. Mit Hilfe von Kryptographie kann man einen entsprechenden Schutz gewährleisten und noch ein wenig mehr. Die Ziele der Kryptographie sind:

- Dritte sollen den Inhalt von Daten nicht erfahren. Im Idealfall kann ein Dritter eine mit kryptographischen Methoden bearbeitete Nachricht nicht von einem Zufallstext unterscheiden. Auch statistische Auswertungen auf den „Zufallstext“ und andere sogenannte „Attacken“ sollen keine Rückschlüsse auf die Ursprungsnachricht zulassen.

- Die Integrität, also die „Unverändertheit“, von Daten soll gewährleistet sein. Daten, die von einer Zufallsfolge von Buchstaben nicht zu unterscheiden ist, so zu manipulieren, dass sie nach dem Umkehren der kryptographischen Methode wieder einen Sinn ergeben, ist verständlicherweise sehr schwierig.

Allerdings ist Kryptographie kein Allheilmittel für die Kommunikation in unsicheren Datennetzen wie dem Internet. Zwar kann man, wie oben beschrieben, verhindern, dass Daten auf dem Weg zum Empfänger mitgelesen oder verändert werden, aber es werden auch neue Probleme geschaffen. Das wichtigste ist das Problem der Identität des Kommunikationspartners, das sich leider durch Mathematik nicht lösen lässt. In Kapitel 9 wird der Ansatz vorgestellt, mit dem die Software PGP bzw. GPG versuchen, diesem Problem zu begegnen.

# 2 Begriffe und Definitionen

**Symmetrische Kryptographie** sind kryptographische Methoden, die zum Ver- und Entschlüsseln den selben Schlüssel[1] verwenden.

**Asymmetrische Kryptographie** sind kryptographische Methoden, bei denen jeder Partei ein „private key“ und ein „public key“ zugeordnet wird.

**Klartext** (in der Fachliteratur auch oft „plain text“ genannt) ist eine (geheime) Nachricht, die über einen unsicheren (abhörbaren) Kanal zu einem Kommunikationspartner übertragen werden soll.

**Chiffre-Text** (oder „cipher text“) ist scheinbar eine zufällige Buchstabenfolge, die sich aber mit den nötigen Kenntnissen (Verschlüsselungsverfahren und Schlüssel) wieder in den Klartext übertragen lässt.

**Schlüssel** ist die geheime Komponente, die beim Ver- und Entschlüsseln angegeben werden muss. Da die Verfahren selbst oft bekannt sind, wird auf diese Art und Weise ein Faktor eingeführt, der zum Entschlüsseln bekannt sein muss.

**Verschlüsseln** ist der Prozess, in dem der „plain text“ nach einem eindeutigen, mathematischen Verfahren in den „cipher text“ umgeformt wird.

**Entschlüsseln** ist die Umkehrung des Verschlüsselns. Mit Hilfe eines Schlüssels gewinnt man aus dem Chiffre-Text wieder den Klartext.

**Schlüsselaustauschproblem** wird das Problem genannt, wenn man sich über einen unsicheren Kanal auf einen gemeinsamen geheimen Schlüssel einigen muss, ohne dass Dritte ebenfalls in den Besitz des Schlüssels kommen können.

[1] Auch „key“ oder „shared secret“ genannt

**Kryptoanalyse** ist die Wissenschaft, kryptographische Methoden zu untersuchen und möglicherweise zu vereinfachen.

**private key** ist die geheime Hälfte eines Schlüsselpaares und wird zum Entschlüsseln und Signieren verwendet.

**public key** ist der öffentliche Schlüssel eines Schlüsselpaares und kann sowohl zum Verschlüsseln als auch zum Verfizieren von Signaturen verwendet werden. Er wird üblicherweise auf einem Keyserver abgelegt und ist grundsätzlich jedermann zugänglich.

**Signatur** ist eine „elektronische Unterschrift“. Mit Hilfe asymmetrischer Kryptographie kann man aus der Nachricht und dem private key eine Prüfsumme berechnen, die mit dem public key verifiziert werden kann.

# 3 Überblick über symmetrische Algorithmen

Symmetrische Chiffren verwenden einen Schlüssel um die Nachricht zu verschlüsseln *und* entschlüsseln. Da der Schlüssel bei diesen Methoden allen teilnehmenden Personen bekannt sein muss, wird er in diesem Zusammenhang auch oft „shared secret“ genannt.

## 3.1 Einfache kryptographische Methoden

Ein sehr gutes (und deswegen auch fast immer angeführtes) Beispiel für eine einfache Methode ist der sogenannte „Cäsar-Chiffre“. Hier wird jeder Buchstabe einfach eine bestimmte Anzahl von Stellen im Alphabet weitergerückt. Wenn man also beispielsweise um drei Positionen verrückt ($n = 3$), wird aus

jedem A ein D, aus jedem B ein E, etc. Die Zahl, um wieviele Stellen verschoben wurde, ist in diesem Fall der Schlüssel. Wie sehr einfach ersichtlich ist, gibt es insgesamt 26 verschiedene Schlüssel, was ein Lösen dieser Verschlüsselung zum Kinderspiel macht. So einfach, dass ein Cäsar-Chiffre mit $n = 13$, genannt „ROT-13“[2], im Internet ab und zu verwendet wird, um unwichtige Texte, wie die Pointen von Witzen, vor all zu voreiligen Augen zu verbergen. Die Zahl 13 wurde gewählt, weil hier zum Ver- und Entschlüsseln der selbe Algorithmus verwendet werden kann: Rotiert man zwei mal um je 13 Stellen, hat man insgesamt um 26 Stellen rotiert. Da dies die Anzahl der Buchstaben im Alphabet ist, wird jeder Buchstabe wieder auf sich selbst abgebildet und der plain text zurück gewonnen.

## 3.2 Moderne symmetrische Verfahren

Moderne symmetrische Algorithmen sind ziemlich komplex und eine genaue Betrachtung der Funktionsweise würde den Rahmen dieser Arbeit sprengen. Deshalb seien hier nur die drei wichtigsten Verfahren genannt:

Die ersten beiden repräsentieren die große Gruppe der „Blockchiffren“, bei denen der Klartext in Blöcke fester Größe (oft 56 Bit bzw. 7 Buchstaben) zerlegt und diese Blöcke dann verschlüsselt werden. Diese Art der Algorithmen findet sich im EDV-Bereich fast ausschließlich.

Der dritte Algorithmus steht stellvertretend für die Gruppe der Strom-Chiffren. Wie der Name schon suggeriert, sind diese Algorithmen in der Lage, einen Datenstrom zu verschlüsseln. Das ist vor allem bei Mobilfunk-Techniken von Bedeutung.

**DES** steht für „**D**ata **E**ncryption **S**tandard“. Er wurde von IBM entwickelt

[2] „ROT“ steht für „rotieren“ und „13“ für die Anzahl der Stellen, um die rotiert wird.

und vom NIST[3] zum Standard erhoben. Seine 56-Bit Schlüssel sind für den heutigen Stand der Technik ein wenig kurz, weswegen häufig der 3DES[4] eingesetzt wird.

**AES** steht für „**A**dvanced **E**ncryption **S**tandard“ und ist als Nachfolger des DES gedacht. Für den AES gab es einen Wettbewerb mehrerer Algorithmen. Voraussetzungen für die Teilnahme waren unter anderem die Freiheit von Patenten, mögliche Schlüssellängen von 128, 192 und 256 Bit und eine möglichst einfache und effektive Implementierung in Hardware. In der Endausscheidung konnte sich der Algorithmus **Rijndael**[5] durchsetzen und ist jetzt der AES. Zwar sind in letzter Zeit Angriffe auf den AES bekannt geworden ([7]), zumeist mit Hilfe der „Linearen Kryptoanalyse“[6], aber praktische Bedeutung hat dies kaum. Und für den Notfall existieren mit den anderen AES-Finalisten weitere vielversprechende Alternativen, auf die zurückgegriffen werden kann.

**RC4** [7] ist einer der zahlreichen Algorithmen des Crypto-Genies Ron Rivest

[3] „National Institute of Standards and Technology“, die Normierungs-Behörde der US-Amerikanischen Regierung.

[4] Tripple-DES – Die doppelte Anwendung des DES: Der Klartext wird zunächst mit einem ersten Schlüssel verschlüsselt, anschließend mit einem zweiten Schlüssel **ent**schlüsselt und danach noch einmal mit dem ersten Schlüssel verschlüsselt. Der Grund für das Entschlüsseln als zweiten Schritt ist ein rein praktischer: Hat ein Hersteller den Tripple-DES in Hardware implementiert, kann man den einfachen DES realisieren, indem man den selben Schlüssel zweimal verwendet – das Ver- und Entschlüsseln heben sich dann gegenseitig auf. Theoretisch sind auch andere Varianten denkbar, aber nicht üblich.

[5] Der Name vereint die beiden Nachnamen, „Rijmen“ und „Daemen“, der Erfinder

[6] Die Lineare Kryptoanalyse versucht, vereinfacht gesagt, den Algorithmus als polynomes Gleichungssystem darzustellen und dadurch die Zahl der in Frage kommenden Schlüssel zu begrenzen.

[7] Steht für „**R**ivest **C**ipher number **4**“

und einer der einfachsten Stromchiffren. Die Einfachheit und gleichzeitige Sicherheit des Agorithmus ist extrem beeindruckend. Das Herzstück des Algorithmus besteht lediglich aus vier Zeilen Programmcode und kann in [1], Seite 158 nachgelesen werden.

### 3.3 Hybridverfahren

Im Vergleich zu den im folgenden beschriebenen asymmetrischen Verfahren sind symmetrische Verfahren in aller Regel mindestens um den Faktor 2 schneller und kommen mit kürzeren Schlüssellängen aus. Diese Vorteile sind der Grund, warum oft sogenannte Hybridverfahren zum Einsatz kommen: Mit einem asymmetrischen Verfahren wird das shared secret übermittelt und die eigentliche Nachricht wird „herkömmlich“ mit einem symmetrischen Verfahren verschlüsselt. Mit der Verwendung von asymmetrischer Kryptographie für den Austausch des Schlüssels wird das „Schlüsselaustauschproblem“ gelöst. Eine andere Lösung besteht in der Anwendung des Diffie-Hellman-Verfahrens zur Schlüsselgenerierung, welches in Kapitel 5 beschrieben wird und auf dem Problem des „diskreten Logarithmus“ beruht.

## 4 Grundbausteine asymmetrischer Verfahren

Die folgenden zwei mathematischen Sätze sind außerhalb der Kryptographie eher unbedeutend und daher wenig bekannt:

### 4.1 Die Eulersche-$\phi$-Funktion

Die Eulersche-$\phi$-Funktion macht eine Aussage darüber, wie viele teilerfremde Zahlen zu einer Zahl $n$ im Intervall $[1; n-1]$ existieren. Die Zahl 1 wird dabei

immer als teilerfremde Zahl gezählt. Ist $n$ eine Primzahl, so macht man sich leicht klar, dass $\phi(n) = n - 1$ gelten muss.

Ist $n$ das Produkt zweier Primzahlen $p$ und $q$, so gilt $\phi(n) = \phi(pq) = (p-1)(q-1)$. Das heisst, die Berechnung der Eulerschen-$\phi$-Funktion ist für das Produkt zweier Primzahlen genau dann trivial, wenn beide Faktoren bekannt sind.

Eine weitere für die Kryptographie wichtige Aussage der $\phi$-Funktion ist, dass folgender Zusammenhang gilt, wenn $\phi(n)$ und $e$ teilerfremd sind:

$$p^e \equiv p^{e \bmod \phi(n)} \pmod{n} \tag{1}$$

Ein Beweis dieses Zusammenhangs ist leider im Rahmen dieser Facharbeit nicht möglich. Interessierte seien auf [5] verwiesen.

## 4.2 Der erweiterte euklidische Algorithmus

Mit Hilfe des euklidischen Algorithmus kann man zu zwei Zahlen den größten gemeinsamen Teiler (ggT) effizient berechnen. Modifiziert man den Algorithmus ein wenig, kann man ihn dazu verwenden, zu einer Zahl $e$ das inverse Element $d$ in einer Modulo-Restgruppe zu finden. Das heisst, $d$ ist die Zahl, welche mit $e$ multipliziert anschließend um $n$ reduziert 1 ergibt. Mathematisch lässt sich das Problem wie folgt beschreiben:

$$e \cdot d \equiv 1 \pmod{n} \tag{2}$$

Das inverse Element ist nicht einfach $d = \frac{1}{e}$, da wir es mit Modulo-Multiplikationen zu tun haben. Es gibt aber immer Zahlen, die, wenn sie um $n$ reduziert werden, 1 ergeben. Eine solche Zahl, die zudem noch durch $e$ teilbar ist, ergibt bei der Division durch $e$ das gesuchte inverse Element $d$.

Die genaue Anwendung und Funktionsweise des erweiterten euklidischen Algorithmus würden den Rahmen dieser Arbeit sprengen, deswegen sei an

dieser Stelle auf [1], Seite 97 und [4] verwiesen, wo der erweiterte euklidische Algorithmus vorgestellt wird.

# 5 Schlüsselaustausch nach Diffie-Hellman

Diffie und Hellman entwickelten ein Verfahren, um das Schlüsselaustauschproblem zu umgehen. Mit Hilfe des Verfahrens einigt man sich über einen unsicheren Kanal auf einen Schlüssel, ohne dass Dritte ebenfalls an den Schlüssel kommen können. Zur Verschlüsselung ist das Verfahren nicht geeignet – hier müssen herkömmliche Algorithmen verwendet werden – aber nachdem der Schlüssel ausschliesslich den beiden Kommunikationspartnern bekannt ist, ist das kein Problem, da jetzt problemlos auch symmetrische Algorithmen angewendet werden können.

Die Sicherheit des Verfahrens beruht auf dem Problem des „diskreten Logarithmus“. Bei folgender Gleichung ist es sehr schwer, bei gegebenen $p$, $w$ und $n$ den Exponenten $x$ zu bestimmen:

$$p^x \equiv w \pmod{n} \tag{3}$$

Eine weitere wichtige Tatsache ist, dass sich folgende Schritte auch auf Modulo-Exponentationen übertragen lassen:

$$(a^b)^c = a^{b \cdot c} = (a^c)^b \Longrightarrow (a^b)^c = a^{b \cdot c} = (a^c)^b \pmod{n} \tag{4}$$

Diffie und Hellman nutzen diese beiden Tatsachen aus, um zwei Parteien über einen unsicheren, öffentlichen Kanal einen gemeinsamen, geheimen Schlüssel zu berechnen. Zunächst einigen sich die Kommunikationspartner auf zwei Zahlen $p$ und $n$. Zusätzlich werden je noch eine geheime Zufallszahl $x_1$ bzw. $x_2$ benötigt, die jede Partei lokal erzeugt. Diese Zahlen $x_1$ und $x_2$

werden nicht ausgetauscht. Die Partner berechnen nun

$$p^{x_i} \bmod n = w_i \tag{5}$$

und tauschen die Ergebnisse $w_1$ bzw. $w_2$ aus. Nun müssen beide Parteien nur noch das erhaltene Ergebnis mit ihrem eigenen geheimen $x$ potenzieren (und um $n$ reduzieren) und erhalten so das selbe Ergebnis.

Einer dritten Person ist das Ergebnis nicht bekannt, selbst wenn sie die gesamte Kommunikation mitverfolgt hat, da die Zahlen $p$ und $n$ sowie das Ergebnis $w$ keinen Rückschluss auf die verwendeten Werte für $x$ zulassen.

Für die Zahlen $p$ und $n$ müssen ein paar Voraussetzungen erfüllt sein, damit dieses Verfahren auch sicher funktioniert. $n$ muss eine Primzahl sein und $p$ sollte idealerweise ein Generator der Gruppe $Z(p, \cdot)$ sein ([1], Seite 105). Für Anwender ist dieses Detail eher unwichtig, da $p$ und $n$ häufig fest vorgegeben sind und nur der Wert für $x$ jedesmal neu berechnet wird.

# 6 Das Merkle-Hellman-Verfahren

Das Merkle-Hellman-Verfahren beruht auf dem „Untersummen-Problem“, auch als „Rucksack-Problem“ bekannt. Der Name stammt von einer sehr anschaulichen Aufgabenstellung: Es sind eine Reihe Gewichte bekannt, von denen sich einige in einem Rucksack befinden. Das Gesamtgewicht des Rucksacks ist bekannt und es soll nun bestimmt werden, welche Einzelgewichte (Elemente) sich in dem Rucksack befinden. Die Lösung dieses Problems ist nicht trivial, da weder gewährleistet ist, dass überhaupt eine Lösung existiert, noch dass eine gefundene Lösung eindeutig ist.

Sind die möglichen Gewichte beispielsweise $\{2, 3, 5, 9\}$ und das Gesamtgewicht 6, dann gibt es keine Lösung. Ist das Gesamtgewicht hingegen 5, dann gibt es zwei Lösungen.

## 6.1 Generierung des public key

Wenn man das Untersummen-Problem zum Verschlüsseln nutzen möchte, muss man zunächst sicherstellen, dass stets eine eindeutige Lösung existiert. Zu diesem Zweck bedient man sich des folgenden Tricks: Jedes neue Element muss größer sein, als die Summe aller bisheriger Elemente. Mathematisch ausgedrückt:

$$g_{i+1} > g_i + g_{i-1} + g_{i-2} + \ldots + g_2 + g_1 \tag{6}$$

Damit ist eine Lösung des Problems einfach und das Verfahren ähnelt dem Übertragen einer Dezimalzahl in das Binärsystem: Ist das Gesamtgewicht größer oder gleich dem schwersten Einzelgewicht, so muss sich dieses Gewicht im Rucksack befinden, da alle anderen Gewichte zusammen leichter als das schwerste Gewicht, demzufolge auch leichter als das Gesamtgewicht sind. Nun wird das schwerste Gewicht gegebenenfalls vom Gesamtgewicht abgezogen und das Verfahren für das nächstschwerere Gewicht wiederholt.

Das Problem ist, dass es für einen Angreifer ebenso einfach ist wie für den Empfänger, an die Nachricht heran zu kommen. Schliesslich kann er das selbe Prinzip verwenden. Deswegen wird aus dieser Menge $\{g_1, g_2, \ldots, g_i\}$ noch eine neue Menge $\{G_1, G_2, \ldots, G_i\}$ gebildet, bei der dieses Prinzip nicht mehr funktioniert. Eine algebraische Summe von Elementen der neuen Menge kann so umgewandelt werden, so dass sie sich aus Elementen der ursprünglichen Menge zusammensetzt.

Um $\{G_1, G_2, \ldots, G_i\}$ zu bilden wird noch eine beliebige Zahl $n$ benötigt, welche lediglich größer als die Summe aller $g_i$ sein muss. Ausserdem braucht man noch eine zu $n$ teilerfremde Zahl $e$. Nun berechnet man für jedes $g_i$ das zugehörige $G_i$ nach folgendem Algorithmus:

$$(g_i \cdot e) \bmod n = G_i \tag{7}$$

Die so gewonnene Menge $\{G_1, G_2, \ldots, G_i\}$ ist der public key.

## 6.2 Generierung des private key

Den private key bilden zusammen die ursprüngliche Menge $\{g_1, g_2, \ldots, g_{i-1}, g_i\}$, die Zahl $n$ und das multiplikative Element $d$ zur Zahl $e$. Zur Berechnung von $d$ wird der euklidische Algorithmus, wie er in 4.2 vorgestellt wurde, verwendet. Damit lässt sich in der folgenden Gleichung ein passendes $d$ finden:

$$e \cdot d \equiv 1 \pmod{n}$$

## 6.3 Ver- und Entschlüsseln

Um Daten zu verschlüsseln, müssen diese zunächst in das Binärsystem übertragen und in Blöcke mit je $i$ Elementen aufgeteilt werden, wobei $i$ die Anzahl der Elemente der Gruppe $\{G_1, G_2, \ldots, G_{i-1}, G_i\}$ ist. Die so erhaltene Menge sei $\{b_1, b_2, \ldots, b_{i-1}, b_i\}$, wobei jedes $b_i$ für ein Bit, also entweder 0 oder 1 steht. Werden die beiden Mengen als mehrdimensionale Vektoren interpretiert, so ist das Skalarprodukt der beiden der Cipher-Text $C$:

$$\begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_i \end{pmatrix} \circ \begin{pmatrix} G_1 \\ G_2 \\ \vdots \\ G_i \end{pmatrix} = (b_1 G_1 + b_2 G_2 + \ldots + b_i G_i) = C \tag{8}$$

Um „den Rucksack wieder auszupacken“ benötigt der Empfänger lediglich $d$ und $n$. Mit folgender Rechnung erhält er eine Summe $C'$, die sich aus Elementen der Menge $\{g_1, g_2, \ldots, g_{i-1}, g_i\}$ zusammensetzt. Da in dieser Menge jedes nächst größere Element größer ist als die Summe aller vorherigen Elemente, kann leicht auf die einzelnen Elemente geschlossen werden.

$$C' = (C \cdot d) \bmod n \tag{9}$$

Nun kann der Empfänger das in 6.1 beschriebene Verfahren nutzen, um aus $C'$ wieder $\{b_1, b_2, \ldots, b_i\}$ des Plain-Textes zu bestimmen.

# 7 Das RSA-Verfahren

Der RSA ist ein asymmetrisches Kryptographieverfahren, das von Ron **R**ivest, Adi **S**hamir und Leonard **A**dleman entwickelt wurde. Seine Sicherheit beruht auf dem Faktorisierungsproblem, also das Zerlegen einer Zahl in seine Primfaktoren. Der RSA erfreut sich größter Beliebtheit, weil er durch seine einfache Anwendung besticht. Das sollte aber nicht über die Komplexität der mathematischen Hintergründe hinwegtäuschen.

## 7.1 Generierung des public key

Um den public key zu generieren, wählt man zunächst zwei große Primzahlen zufällig aus. Da man sehr große Primzahlen braucht, um eine möglichst sichere Verschlüsselung zu erreichen, ist das gar nicht so einfach, wie man zunächst vermuten mag. Glücklicherweise gibt es Möglichkeiten, wie etwa das „Kriterium von Pocklington“, um aus kleinen Primzahlen und einem Zufallswert, neue größere Primzahlen zu errechnen.

Die beiden Primzahlen nennt man üblicherweise $p$ und $q$, deren Produkt $n$ den ersten Teil des public key darstellt.

$$n = p \cdot q \tag{10}$$

Der zweite Teil besteht lediglich aus der Zufallszahl $e$. Üblicherweise wird für $e$ einer der Werte 3, 17 oder 65537 verwendet, da diese wenige Einsen in ihrer Binärdarstellung aufweisen und Computer so leicht mit ihnen potenzieren können ([1], Seite 109). Für $e$ gibt es eine weitere Vorraussetzung, die

in 7.2 erklärt wird, wegen der es sich empfiehlt, für $e$ ebenfalls eine Primzahl zu wählen.

Um nun eine Nachricht, üblichweise $P$ (für „Plain-Text“) genannt, zu verschlüsseln, potenziert man mit $e$ und reduziert um $n$. So erhält man den Cipher-Text $C$:

$$C = P^e \bmod n \tag{11}$$

Das Anwenden der Modulofunktion verhindert, dass ein Dritter, der $e$ kennt[8] die Potenzierung rückgängig machen und damit $P$ erhalten kann. Das ist das Problem des „diskreten Algorithmus“, das schon in Kapitel 5 verwendet wurde.

## 7.2 Generierung des private key

Mit Hilfe der Eulerschen-$\phi$-Funktion (1) lässt sich folgender Zusammenhang feststellen:

$$P^e \equiv P^{e \bmod \phi(n)} = P^{e \bmod \phi(pq)} \pmod n \tag{12}$$

Jetzt stellt man folgende Überlegung an: Wenn es für $e$ ein inverses Element $d$ gibt, für das gilt $(d \cdot e) \bmod \phi(n) = 1$, dann kann man den Cipher-Text $C = P^{e \bmod \phi(n)} \bmod n$ mit $d$ potenzieren und erhält $P^1 = P$:

$$C^d \equiv (P^e)^d = P^{e \cdot d} \equiv P^{e \cdot d \bmod \phi(n)} = P^1 = P \pmod n \tag{13}$$

Allerdings ist nicht garantiert, dass ein inverses Element überhaupt existiert. Damit dies der Fall ist, müssen $\phi(n)$ und $e$ teilerfremd sein. Das inverse Element $d$ ist das einzige (ausser der bekannten Zahl $n$), das zum Entschlüsseln gebraucht wird. Um $d$ berechnen zu können, muss man aber $\phi(n)$ berechnen können. Das ist aber nur möglich, wenn $p$ und $q$ bekannt sind.

[8] $e$ ist Teil des public key und daher frei zugänglich.

Wenn $\phi(n)$ bekannt ist, muss immer noch folgende Gleichung gelöst werden, was analog zu (2) mit Hilfe des erweiterten Euklidischen Algorithmus einfach zu bewerkstelligen ist:

$$e \cdot d \equiv 1 \pmod{\phi(n)} \tag{14}$$

## 7.3 Ver- und Entschlüsseln

Werden sämtliche Zwischenschritte der Gleichungen (11) und (13) ignoriert, so tritt die anfangs gepriesene Einfachheit des Verfahrens an den Tag. Das Verfahren lässt sich auf die folgenden Gleichungen reduzieren:

$$P^e \equiv C \pmod{n} \tag{15}$$

$$C^d \equiv P \pmod{n} \tag{16}$$

## 7.4 Das Problem mit den großen Zahlen

Obwohl das Verfahren an sich so einfach ist, wie in (15) und (16) beschrieben, macht die Größe der Zahlen durchaus Probleme. Selbst wenn man $p$ und $q$ extrem klein wählt (unter 100) ist $d \cdot e > n$, da $d \cdot e = s \cdot n + 1; s \in \mathbb{N}$. Das heißt: $d \cdot e$ ist um eins grösser als ein Vielfaches von $n$, also mindestens $n+1$. Das Problem ist also, mit einer hundertstelligen Zahl **zu potenzieren** – eine Aufgabe, die auch für moderne Computer eine Herausforderung darstellt. Man verwendet daher einen Trick, der unter anderem in [2] beschrieben wird:

Um beispielsweise $7^{55} \pmod{143}$ zu berechnen, stellt man den Exponenten 55 zunächst als Summe von Potenzen von 2 dar:

$$55 = 2^0 + 2^1 + 2^2 + 2^4 + 2^5 = 1 + 2 + 4 + 16 + 32 \tag{17}$$

Damit kann man die Potenz umformen zu:

$$7^{55} = 7^1 \cdot 7^2 \cdot 7^4 \cdot 7^{16} \cdot 7^{32} \tag{18}$$

Nun berechnet man zunächst die kleinen Faktoren mit Hilfe der Modulo-Multiplikation:

$$7^4 \bmod 143 = (7 \cdot 7 \cdot 7 \cdot 7) \bmod 143 = 113 \tag{19}$$

Jetzt kann man die größeren Potenzen mit Hilfe der Ergebnisse bei den kleineren Potenzen nach dem gleichen Schema berechnen:

$$7^{16} \bmod 143 = (7^4 \cdot 7^4 \cdot 7^4 \cdot 7^4) \bmod 143 = (113 \cdot 113 \cdot 113 \cdot 113) \bmod 143 = 48 \tag{20}$$

$$7^{32} \bmod 143 = (7^{16} \cdot 7^{16}) \bmod 143 = (48 \cdot 48) \bmod 143 = 16 \tag{21}$$

Die Modulo-Multiplikation verhindert, dass die Zahlen zu groß werden und reduziert zugleich noch die Anzahl der notwendigen Rechenschritte.

# 8 Digitale Signaturen

Bisher wurde nur eines der grundlegenden Probleme behandelt:

- Durch die Verschlüsselung ist der Inhalt der Daten für Dritte unzugänglich.

Die beiden anderen Probleme,

- Schutz der Daten vor Manipulation,
- Sicherstellen der Identität des Kommunikationspartners,

wurden noch nicht behandelt.

Das Problem der Identität ist im Internet auf Grund der freien Zugänglichkeit der Verfahren nicht lösbar. Der Ansatz der Software PGP bzw. GPG werden im Kapitel 9 erläutert.

Für den Schutz der Daten vor Manipulation können die genannten asymmetrischen kryptographischen Verfahren herangezogen werden. Grundsätzlich ist dies ein sehr ähnliches Problem wie die Verschlüsselung, nur die Richtung ist diesmal umgekehrt: Die Authentität der Daten muss leicht nachprüfbar sein, das Fälschen aber so schwer wie möglich. Die Lösung des Problems besteht darin, den Text mit dem private key zu **verschlüsseln**. Der Cipher-Text kann von jedem, der im Besitz des dazugehörigen public key ist, wieder in den Plain-Text zurückgerechnet werden. Grundsätzlich besteht nach wie vor die Gefahr, dass Unbefugte, die Signatur verändern. Die Schwierigkeit besteht darin, die Signatur so zu verändern, dass der enthaltene Plain-Text einen Sinn ergibt. Bedenkt man, das wichtige Dokumente oft erwartet werden, ist die nötige Zeit[9] eher nicht ausreichend, um eine unbemerkte Manipulation vorzunehmen.

# 9 Public-Key-Infrastrukturen

Die E-Mail-Verschlüsselungs-Software „PGP“[10], die 1991 geschrieben wurde und inzwischen der Quasi-Standard für E-Mail-Verschlüsselung ist und deren freies Pendant „GPG“[11] verwenden ein Verfahren namens „Web of Trust“ um die Identität einer Person sicherzustellen. Das Verfahren geht davon aus, dass jeder Nutzer ein paar anderen „vertraut“. Vertrauen wird definiert als die Sicherheit, dass ein public key tatsächlich zu der Person gehört, die quasi auf dem „Namensschild“ steht. Es gibt auch Institutionen, die nach Vorlage des Personalausweises ein solches Vertrauen „aussprechen“; der Heise-Verlag

[9] Viele Kryptologen geben sich mit 30 Jahren als „nötige Zeit“ nicht zufrieden...

[10] **P**retty **G**ood **P**rivacy

[11] **G**NU **P**rivacy **G**uard

hat sich in den letzten Jahren als eine solche Institution engagiert[12]. Ein Nutzer, der einem anderen vertraut, signiert dessen public key mit seinem eigenen private key. Digitale Signaturen wurden im Kapitel 8 behandelt.

Bekommt nun ein Nutzer, der einigen anderen sein Vertrauen ausgesprochen hat, einen ihm unbekannten public key, kann er überprüfen, ob dieser vielleicht mit einem Schlüssel seines Vertrauens signiert wurde. Überlicherweise wird bei einem solchen „trust check“ über mehrere Ebenen rekursiv nach einem „Pfad“ gesucht. Wird auf diese Weise ein Weg zu dem in Frage stehenden Schlüssel gefunden, so kann man zumindest mit begründeter Zuversicht davon ausgehen, dass die Identität der Person mit der auf dem Schlüssel übereinstimmt.

# 10 Schlussgedanken

Kryptographie ist schon lange kein Thema mehr, das nur für das Militär, Regierungen oder internationale Konzerne interessant ist. Mit dem AES steht der „beste“ zivile Algorithmus zur freien[13] Verfügung[14] und auch das RSA-Verfahren ist für Privatpersonen kostenfrei. Die Einfachheit, mit der im Internet unbemerkt Daten mitgelesen, analysiert oder sogar verändert werden können, ist beispielslos und sollte alamieren. Dass es das nicht tut, zeigt die statistische Auswertung aller öffentlich bekannten public keys[15]: Gerade einmal 1.649.308 public keys[16] sind weltweit bekannt (Stand April 2002).

Allerdings zeichnet sich langsam auch eine wachsende Akzeptanz der

[12] http://www.heise.de/ct/pgpCA/

[13] „free“, as in „speech“, not „free“, as in „beer“ – Linus Torvalds

[14] Bert-Jaap Koops „Crypto Law Survey“ gibt Auskunft über die Import- und Exportbestimmungen zahlreicher Länder – http://rechten.uvt.nl/koops/cryptolaw/

[15] http://dtype.org/keyanalyze/

[16] Nur 158.707 entsprechen obligatorischen Sicherheitsansprüchen!

Kryptographie ab, so übermittelt einer der größten deutschen E-Mail-Dienste „web.de“ Daten im Normalfall inzwischen verschlüsselt.

Aus rechtlicher Sicht ist Deutschland eine Oase der Kryptographie: Es ist eines der wenigen Länder, in denen Kryptographie nicht nur uneingeschränkt erlaubt, sondern deren Entwicklung sogar noch gefördert wird. Deshalb ist es wenig verwunderlich, dass viele der Entwickler freier Kryptographieprojekte, wie etwa OpenSSL[17] oder GnuPG[18] aus Deutschland stammen.

Die Entwicklung der Kryptographie wird auch in naher Zukunft interessant bleiben, wie einige Beispiele belegen können:

- Am 10. April 2002 wurde [7] veröffentlicht, eine Arbeit von Nicolas Courois und Josef Pieprzyk, in der eine XSL-Attacke auf den AES vorgestellt wird. Etwas verständlicher hat es Bruce Schneider in seinem Cryptogram ([8]) zusammengefasst.

- Adi Shamir[19] veröffentlichte schon 1999 Pläne für einen optoelektronischen Apparat, namens „TWINKLE“, mit dem das Faktorisieren einer 512 Bit langen Zahl in weniger als einem Jahr realisierbar sein soll.

- Am 23.01.2003 veröffentlichten Asi Shamir und Eran Tromer eine Arbeit, in der eine neue Hardware, genannt „TWIRL“, vorgestellt wird ([6]). Damit soll es möglich sein, eine 1024 Bit große Zahl binnen eines Jahres zu faktorisieren. Der Preis von etwa 10 Millionen US-Dollar sollte Hobby-Kryptoanalytiker allerdings abschrecken.

[17] http://www.openssl.org/
[18] GPG wurde schon im Abschnitt 9 erwähnt
[19] Das „S“ in „RSA“

# Literatur


[1] Schmeh, Klaus: *Kryptographie und Public-Key-Infrastrukturen im Internet.* Heidelberg: dpunkt.verlag 2001

[2] Hellman, Martin E.: *Die Mathematik von Public-Key-Verfahren.* Spektrum der Wissenschaft – Dossier Kryptographie, 2001, Seite 32

[3] Dr.-Ing. Huber, Klaus und Dr. Tönsing, Friedrich: *Kryptologische Verfahren.* Telekom Unterrichtsblätter, 51. Jahrgang, 09/1998, Seite 436

[4] http://www.iti.fh-flensburg.de/lang/algorithmen/code/krypto/euklid.htm erstellt am 15.11.2002, angesehen am 28.12.2002

[5] Lochbihler, Andreas: http://home.t-online.de/home/andreas.lochbihler/F/f.htm erstellt am 05.12.2001, angesehen am 28.12.2002

[6] Shamir, Adi und Tromer, Eran: http://psifertex.com/download/twirl.pdf erstellt am 23.01.2003, angesehen am 25.01.2003

[7] Courois, Nicolas und Pieprzyk, Josef: http://eprint.iacr.org/2002/044/ erstellt am 10.04.2002, angesehen am 23.01.2003

[8] Schneider, Bruce: http://www.counterpane.com/crypto-gram-0209.html erstellt am 15.09.2002, angesehen am 23.01.2003

# Erklärung

Ich erkläre hiermit, dass ich die Facharbeit ohne fremde Hilfe angefertigt und nur die im Literaturverzeichnis angeführten Quellen und Hilfsmittel benutzt habe.