Anhang IV zur Vorlesung Kryptologie: Public-Key Kryptographie

von
Peter Hellekalek

Fakultät für Mathematik, Universität Wien, und
Fachbereich Mathematik, Universität Salzburg

Tel: +43-(0)662-8044-5310
Fax: +43-(0)662-8044-137
e-mail: peter.hellekalek@sbg.ac.at
web: http://random.mat.sbg.ac.at/

Wien, 24. April 2009

Inhaltsverzeichnis

Kapitel 1

Asymmetrische Kryptographie

▷ **Inhalt**
In diesem Kapitel stellen wir Verfahren vor, die sich radikal von den bekannten Chiffren unterscheiden.

▷ **Ziel**
Wir diskutieren die bekanntesten Konzepte der asymetrischen Kryptographie (Englisch: "public-key cryptography").

▷ **Stichwörter**
Die Stichwörter zu diesem Kapitel lauten

- asymmetrische Verschlüsselung
- digitale Signatur
- RSA
- Primzahlen

▷ **Literatur**

A. J. Menezes, P. C. van Oorschot, and S.A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, Boca Raton, 1997.

Stinson, D. R.[Sti06] *Cryptography. 3rd edition.* Chapman and Hall/CRC Press, Boca Raton 2006.

W. Trappe and L. Washington[TW06] *Introduction to Cryptography with Coding Theory.* Pearson Prentice Hall, 2nd edition, 2006.

sowie die folgende Monographie:
Salomaa, A. [Sal97] *Public Key Cryptography.* Springer-Verlag, New York, 1997.

1.1 Einleitung

Mit den Methoden der klassischen (dh. symmetrischen) Kryptographie können die vier Grundaufgaben der Kryptographie nur unzureichend gelöst werden:

1. *Vertraulichkeit*
   Die Nachricht wird verschlüsselt, um die Vertraulichkeit des Inhalts zu gewährleisten. Der Inhalt der Nachricht soll geheim bleiben.
2. *Authentifikation*
   Sie ermöglichst den Identitätsnachweis des Senders der Nachricht.
3. *Integrität*
   Die Inhalte der Nachrichten bleiben bei der Übertragung unverändert.
4. *Verbindlichkeit*
   Die Nachricht ist verbindlich. Der Sender kann nicht abstreiten, eine bestimmte Nachricht zu einer bestimmten Zeit abgeschickt zu haben.

Die Inhalte von Nachrichten lassen sich durch moderne symmetrische Verschlüsselungsverfahren wie DES und AES sehr gut geheim halten, aber die Übermittlung der Schlüssel ist umständlich und natürlich riskant.

Auch die Verwaltung der Schlüssel ist aufwändig: Sollen in einem Rechnernetz mit n Computern je zwei Computer über einen gemeinsamen geheimen Schlüssel verschlüsselt kommunizieren können, dann sind dafür $\binom{n}{2} = O(n^2)$ Schlüssel erforderlich. Alle diese Schlüssel müssen auf sicherem Weg übermittelt werden.

In der Praxis ist also schon die erste Grundaufgabe der Kryptographie ein nicht geringes Problem, wenn wir nur symmetrische Kryptographie zur Verfügung haben. Speziell die zweite und die vierte Aufgabe sind mit symmetrischer Verschlüsselung nur recht umständlich zu lösen.

Ein Hinweis auf die praktischen Dimensionen: bei der Fluglinie Lufthansa mußten diese Aufgaben vor kurzem (im Jahr 2001) für etwa 60000 Computerarbeitsplätze gelöst werden. In derart großen Dimensionen wäre die klassische Kryptographie überfordert.

Allerdings: vor mehr als einem Vierteljahrhundert wurde die Kryptographie durch ein revolutionär neues Konzept total verändert. Im Jahr 1976 präsentierten Whitfield Diffie und Martin E. Hellman in ihrer berühmten Arbeit „New Directions in Cryptography“ [DH76] das Konzept der *asymmetrischen* oder *Public Key*-Verschlüsselung.

Wir geben die ursprüngliche Definition von Diffie und Hellman wieder, angepaßt an unsere Notation.

1.1 Definition (Asymmetrisches oder Public Key-Verschlüsselungssystem)
Unter einem asymmetrischen oder Public Key-Verschlüsselungssystem verstehen wir ein Verschlüsselungssystem $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ mit $\mathcal{P} = \mathcal{C} =: \mathcal{M}$, für das gilt:

1. Für $k \in \mathcal{K}$ ist die Funktion $e_k \in \mathcal{E}$ die inverse Abbildung zu $d_k \in \mathcal{D}$,

$$d_k \circ e_k \;=\; e_k \circ d_k \;=\; \text{Identität}.$$

2. Für $k \in \mathcal{K}$ und $m \in \mathcal{M}$ sind die Werte $e_k(m)$ und $d_k(m)$ leicht zu berechnen.

3. Für fast alle $k \in \mathcal{K}$ ist jeder leicht zu berechnende und zur Funktion d_k äquivalente Algorithmus nur mit Hilfe von e_k praktisch nicht zu berechnen.

4. Für jedes $k \in \mathcal{K}$ ist es praktisch möglich, „inverse“ Paare von Funktionen (e_k, d_k) laut Punkt 1 zu finden.

1.2 Bemerkung
Bei einem Public Key-Verfahren wählt jeder Benutzer einen Schlüssel $k \in \mathcal{K}$ aus und erzeugt die *Verschlüsselungsfunktion* e_k und die dazu gehörige *Entschlüsselungsfunktion* d_k. Nach Punkt 4 von Definition 1.1 ist dies möglich.

Wegen der dritten Forderung in Definition 1.1 darf die Verschlüsselungsfunktion e_k eines Benutzers öffentlich gemacht werden, ohne dass dadurch ein Angreifer die Entschlüsselungsfunktion d_k finden kann.

Man nennt e_k den *öffentlichen Schlüssel* des Benutzers und d_k den *geheimen Schlüssel* des Benutzers (Englisch: public key und private key).

1.3 Bemerkung (Kryptographisches Protokoll)
Unter einem kryptographischen Protokoll versteht man eine endliche Folge von Schritten, um ein kryptographisches Problem zu lösen. Es gilt:

1. Mindestens zwei Personen sind am Protokoll beteiligt (Alice, Bob, ...).

2. Der nächste Schritt des Protokolls kann erst getan werden, wenn der vorhergehende beendet wurde.

3. Jeder am Protokoll Beteiligte muß das Protokoll kennen, insbesondere jeden der nächsten Schritte.

4. Jeder am Protokoll Beteiligte hält sich an das Protokoll.

5. Jeder Schritt des Protokolls ist eindeutig festgelegt.

6. Das Protokoll muß vollständig sein. Das heißt, jede mögliche Situation muß berücksichtigt sein.

7. Das Protokoll verwendet kryptographische Methoden.

1.4 Bemerkung (Protokoll einer Public Key-Verschlüsselung)
Sei $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ ein asymmetrisches Verfahren. Sei d_{Alice} der geheime und e_{Alice} der öffentliche Schlüssel des Benutzers Alice. Analog gelte diese Bezeichnung für Bob, Eve, ...

Die zu lösende Aufgabe ist jene der *Vertraulichkeit* der Nachricht. Alice möchte an Bob eine Nachricht m schicken, die so verschlüsselt werden soll, dass nur Bob die Nachricht lesen kann und niemand anderer.

1. Alice besorgt sich den öffentlichen Schlüssel e_{Bob} von Bob.
2. Alice verschlüsselt den Klartext, also die Nachricht m, mit dem öffentlichen Schlüssel e_{Bob} von Bob und erhält den Geheimtext c,
$$c = e_{\text{Bob}}(m).$$
3. Alice versendet den Geheimtext c über den unsicheren Kanal an Bob. (Dort liegt Eve auf der Lauer und führt einiges im Schilde)
4. Bob entschlüsselt den Geheimtext c mit seinem privaten Schlüssel,
$$d_{\text{Bob}}(c) = d_{\text{Bob}}(e_{\text{Bob}}(m)) = m.$$

Es ist das Grundproblem der Public Key-Verschlüsselung sicherzustellen, dass der öffentliche Schlüssel von e_{Bob} zu Bob gehört und zu niemandem anderen. Es könnte sich sonst Eve (im Internet) als Bob ausgeben und mit Alice vertrauliche Nachrichten austauschen, die Alice eigentlich an Bob schicken möchte. Um genau dieses Problem zu lösen, ist eine sogenannte Public Key-Infrastruktur (PKI) notwendig. Zum Beispiel wird in der Signaturverordnung zum österreichischen Signaturgesetz festgelegt, wie diese PKI auszusehen hat. Weitere Informationen erhalten Sie von der Telekom Control Kommission, der österreichischen Aufsichtsstelle für digitale Signaturen, unter

`http://www.signatur.rtr.at`

Informationen zu einem lokalen Zertifizierungsdienst finden Sie zum Beispiel unter `https://a-cert.argedaten.at/`

Der öffentliche Schlüssel eines Benutzers ist auf der Homepage des Benutzers oder auch von einem *public key server* wie zum Beispiel `pgp.mit.edu` verfügbar. Diese Server müssen gegenüber Hackerattacken besonders gut geschützt sein.

An Hand eines "Fingerabdruckes" (Englisch: fingerprint) des öffentlichen Schlüssels von Bob kann Alice sicherstellen, dass es sich wirklich um den öffentlichen Schlüssel von Bob handelt. Bob und Alice können den Fingerabdruck zum Beispiel per Telephon überprüfen. Ein Hinweis: diese Art von Fingerabdrücken sind nichts anderes als Hashwerte.

Wegen Eigenschaft 3 in Definition 1.1 kann kein Angreifer aus dem öffentlichen Schlüssel e_{Bob} von Bob den privaten Schlüssel d_{Bob} berechnen. Daher kann nur Bob und niemand sonst den Geheimtext c entschlüsseln, den ihm Alice geschickt hat, denn nur Bob verfügt über die passende Umkehrfunktion zu e_{Bob}:

$$d_{\text{Bob}}(c) \;=\; d_{\text{Bob}}(e_{\text{Bob}}(m)) \;=\; m.$$

Jeder Angreifer, wie zum Beispiel Eve, scheitert an dieser Stelle, denn er verfügt nicht über die passende Umkehrfunktion d_{Bob} zur Funktion e_{Bob}. Einzig für die Funktion d_{Bob} gilt die gewünschte Beziehung $d_{\text{Bob}} \circ e_{\text{Bob}} = \text{Identität}$.

Die Erzeugung von digitalen Signaturen ist in Zusammenhang mit dem elektronischen Handel und Geldverkehr im Internet von großer Bedeutung.

Mit Hilfe der digitalen Signatur sollen neben anderen Aufgaben zwei Probleme gelöst werden:

1. Alice und Bob sollen gegenüber Nachrichten geschützt sein, die von Eve unter dem Namen von Alice an Bob geschickt werden. Diese Art von Fälschung soll zuverlässig erkannt werden können.
2. Alice soll gegen gefälschte Nachrichten von Bob geschützt sein, von denen er behauptet, Alice hätte diese Nachrichten an ihn geschickt.

1.5 Bemerkung (Protokoll zur digitalen Signatur)
Sei $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ ein asymmetrisches Verfahren. Sei d_{Alice} der geheime und e_{Alice} der öffentliche Schlüssel des Benutzers Alice, analog für Bob, Eve, ...

Die Aufgabe lautet wie folgt: Alice möchte an Bob eine signierte Nachricht m schicken. Die digitale Signatur soll sicherstellen (i) dass die Nachricht nur von Alice stammen kann und nicht von jemandem anderen und (ii) dass die Nachricht bei der Übertragung nicht verändert wurde (*Datenintegrität*). Bob soll die digitale Signatur von Alice und die Integrität der Nachricht einfach überprüfen können.

1. Alice signiert den Klartext m mit ihrem privaten Schlüssel. Dies ist nichts anderes als eine Verschlüsselung mit der Funktion d_{Alice}. Alice erhält
$$c = d_{\text{Alice}}(m).$$
2. Alice versendet das Paar(m, c) über den unsicheren Kanal an Bob.
3. Bob besorgt sich den öffentlichen Schlüssel e_{Alice} von Alice.
4. Bob überprüft die digitale Signatur von Alice. Dazu wendet er e_{Alice} auf c an und kontrolliert, ob die Gleichung
$$m = e_{\text{Alice}}(c) \tag{1.1}$$
gilt. Wenn dies der Fall ist, dann betrachtet Bob das Dokument m als von Alice signiert. Wenn $m \neq e_{\text{Alice}}(c)$, dann wurde entweder der Klartext m bei der Übertragung verändert oder es stimmt die Signatur nicht.

Durch die Anwendung des öffentlichen Schlüssels e_{Alice} auf die erhaltene Nachricht c überprüft Bob die Signatur von Alice. Wegen der Eigenschaft 3 in Definition 1.1 erzeugt nur die Funktion e_{Alice} aus der empfangenen Nachricht c den ursprünglichen Klartext m und keine andere Funktion,
$$e_{\text{Alice}}(c) = e_{\text{Alice}}(d_{\text{Alice}}(m)) = m.$$
Jeder Angreifer, wie zum Beispiel Eve, scheitert an dieser Stelle, denn $e_{\text{Eve}} \circ d_{\text{Alice}} \neq \text{Identität}$. Einzig für die Funktion e_{Alice} gilt die gewünschte Beziehung $e_{\text{Alice}} \circ d_{\text{Alice}} = \text{Identität}$.

1.6 Bemerkung
Das hier vorgestellte Verfahren zur digitalen Signatur ist ein *theoretisches Konzept*, das in dieser (Roh-) Form noch nicht für die Praxis geeignet ist. Für große Datenmengen ist das Public Key-Verfahren zu langsam.

In der Praxis erzeugt Alice zuerst den *Hashwert* $h(m)$ des Klartextes m, signiert dann den Hashwert mittels ihres privaten Schlüssels,

$$\gamma = d_{\text{Alice}}(h(m)),$$

und versendet dann das Paar (m, γ) an Bob.

Bob empfängt ein Paar (m', γ). Wir gehen hier davon aus, dass Eve eventuell m bei der Übertragung verstümmelt hat, nicht aber γ. (Der Fall, dass auch γ verstümmelt wurde, verläuft analog.)

Bob wendet e_{Alice} auf γ an,

$$e_{\text{Alice}}(\gamma) = e_{\text{Alice}}(d_{\text{Alice}}(h(m))) = h(m).$$

Im nächsten Schritt erzeugt Bob selbst den Hashwert der empfangenen Nachricht m' und erhält den Wert $h(m')$. Wenn die Gleichung

$$h(m) \;=\; h(m')$$

gilt, dann akzeptiert Bob das Dokument m als ein von Alice digital signiertes Dokument, mit allen Rechtsfolgen für Alice (z.B. Steuererklärung, Kaufvertrag, ...).

Stimmen die beiden Hashwerte $h(m)$ und $h(m')$ nicht überein, dann muß Bob davon ausgehen, dass (i) jemand das Dokument m bei der Übertragung verändert hat oder (ii) eine technische Störung vorliegt, die entweder m oder γ verändert hat oder (iii) das Dokument nicht von Alice signiert wurde. Er wird also das empfangene Dokument m' ablehnen und sich mit Alice in Verbindung setzen. Bob ist also gewarnt.

1.7 Bemerkung
Hashfunktionen ordnen einer Datei beliebiger Größe einen Wert fester Grösse zu. In der Praxis sind Hashwerte 128, 160 oder 256 Bits lang. Kryptographische Hashfunktionen sind so konstruiert, dass es praktisch unmöglich ist, zu einem gegebenen Hashwert H ein Dokument m mit Hashwert $h(m) = H$ zu finden und dass es äußerst unwahrscheinlich ist, dass zwei verschiedene Dokumente $m \neq m'$ den gleichen Hashwert besitzen. Hashfunktionen sind wichtiger Teil der digitalen Signatur. Wir verweisen auf Schneier [Sch96] und Menezes et al. [MvOV97] für Details zu Hashfunktionen.

1.8 Bemerkung (Protokoll zur digitalen Signatur mit Verschlüsselung)
Sei $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ ein asymmetrisches Verfahren. Sei d_{Alice} der geheime und e_{Alice} der öffentliche Schlüssel des Benutzers Alice, analog für Bob, Eve, ...

Die Aufgabe lautet wie folgt: Alice möchte an Bob eine signierte Nachricht m schicken und diese zusätzlich verschlüsseln.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: PGPfreeware 6.5.8 for non-commercial use <http://www.pgp.com>

mQGiBDoQ8UkRBAD5HgWVsgJt00HzZl65tkjG73H4pT1h4PE9NHEZOCnojgKY5T1n
G2gDMiPeKXAG22PMbHWG6eDGco/SYo/cQyRQvUBtLjlopowYh+o4iGIlASgBcN7x
/6Kgb/+vrS0EZ/rGmB+ORELaU9DkHnuebeYMf9DZTz2U/H2hUcfAgP8xZQCg/wDe
rPTEjQNaXjvWZ0dk0tIxeTkD/RUxxu71VGS4Cldbxm4+X/9qFYJu5VQOugKMCnYI
P6fNglK2D6V6sGsb6vr9O2SmWkFaJ/UdiV3gf6QJ7z92C5ftFK6rlBOKCO1AopVL
... (und so weiter)
PZeAEIyJAEYEGBECAAYFAjoQ8UoACgkQoTwz1Sp6LJDdfACfXnAMrVQbalJaNvxT
ncGO9JqhltUAoK8sglIB45ajkvv23HkFxZgUmoH+
=171C
-----END PGP PUBLIC KEY BLOCK-----
```

Abbildung 1.1: Teil eines öffentlichen Schlüssels zu PGP

1. Alice signiert den Klartext m mit ihrem privaten Schlüssel d_{Alice} und verschlüsselt dann mit dem öffentlichen Schlüssel e_{Bob} von Bob. Alice erhält

$$c = e_{\text{Bob}}(d_{\text{Alice}}(m)).$$

2. Alice versendet c über den unsicheren Kanal an Bob.

3. Bob entschlüsselt c mit seinem geheimen Schlüssel d_{Bob} und erhält die signierte Nachricht:

$$d_{\text{Bob}}(c) = d_{\text{Bob}}\left(e_{\text{Bob}}(d_{\text{Alice}}(m))\right) = d_{\text{Alice}}(m).$$

4. Bob besorgt sich den öffentlichen Schlüssel e_{Alice} von Alice.

5. Bob überprüft die digitale Signatur von Alice. Dazu wendet er e_{Alice} auf c an:

$$m = e_{\text{Alice}}(d_{\text{Alice}}(m)). \tag{1.2}$$

Wesentlich ist die Reihenfolge, in der signiert und verschlüsselt wird. Die Verschlüsselung der signierten Nachricht dient hier nicht nur der Vertraulichkeit, sondern sie verhindet auch, dass die Signatur von Alice vom Dokument m abgetrennt werden kann. Würde Alice zuerst verschlüsseln und dann signieren, so könnte Eve die digitale Signatur von Alice leicht abtrennen. Wäre nämlich

$$c = d_{\text{Alice}}(e_{\text{Bob}}(m)),$$

dann muss Eve nur $e_{\text{Alice}}(c) = e_{\text{Bob}}(m)$ bilden und kann dann die für sie unlesbare, aber von ihr signierbare Nachricht $e_{\text{Bob}}(m)$ mit ihrer Signatur versehen und an Bob schicken:

$$c' = d_{\text{Eve}}(e_{\text{Bob}}(m)).$$

Es ist klar, dass derartige Möglichkeiten der Abtrennung der digitalen Signatur unerwünscht sind.

1.2 Das RSA-Verfahren

Nachdem Diffie und Hellman ihr revolutionäres Konzept der asymmetrischen Verschlüsselung vorgestellt hatten, begann eine intensive Suche nach geeigneten Algorithmen für die Implementierung.

Im Jahr 1977 stellten Ron Rivest, Adi Shamir und Leonard Adleman das sogenannte RSA-Verfahren vor und ließen es als US Patent No. 4 405 829 patentieren, obwohl Pohlig und Hellman mehr als ein Jahr vor ihnen ein vergleichbares Verfahren präsentiert hatten (siehe `http://www.cyberlaw.com/rsa.html`). Das RSA-Patent ist am 20.9.2000 abgelaufen. Der Unterschied zwischen RSA und Pohlig-Hellman liegt darin, dass bei RSA der Modul $n = p \cdot q$ eine zusammengesetzte Zahl ist, anstelle von $n = p$ bei Pohlig-Hellman.

1.9 Bemerkung (Schlüsselerzeugung)
Jeder Benutzer des RSA-Verfahrens erzeugt seinen öffentlichen und privaten Schlüssel auf die folgende Weise:

1. Zuerst wählt der Benutzer zwei große Primzahlen $p \neq q$ mit vorgeschriebener Bitlänge L. (Typische Werte für L liegen in der Praxis zwischen 384 und 1024 Bit.)
2. Er bildet das Produkt $n = p \cdot q$.
3. Als nächstes berechnet er $\varphi(n) = \varphi(p) \cdot \varphi(q) = (p-1) \cdot (q-1)$ und wählt eine beliebige Zahl e mit den Eigenschaften
$$1 < e < \varphi(n) \quad \text{und} \quad (e, \varphi(n)) = 1.$$
(In der Praxis muß hier aufgepaßt werden, siehe Menezes et al.[MvOV97] und Salomaa[Sal97, Ch.4.2].)
4. Er berechnet die eindeutig bestimmte Lösung d $(1 < d < \varphi(n))$ der linearen Kongruenz
$$e \cdot d \equiv 1 \pmod{\varphi(n)}.$$
5. Abschließend vernichtet der Benutzer die beiden Zahlen p und q und veröffentlicht das Paar (e, n). Die Zahl d wird geheimgehalten.

1.10 Definition (Öffentlicher und privater Schlüssel)
Das Paar (e, n) heißt der öffentliche Schlüssel des Benutzers. Die Zahl d heißt der geheime Schlüssel des Benutzers.

1.11 Beispiel Es seien $p = 47$ und $q = 59$, also $n = 2773$ und $\varphi(n) = 46 \cdot 58 = 2668$. Wir wählen $e = 17$ und berechnen d,

$$17d \equiv 1 \pmod{2668}.$$

Dazu lösen wir die entsprechende diophantische Gleichung $17d + 2668y = 1$ mit dem Euklidischem Algorithmus,

$$2668 = 17 \cdot 156 + 16$$
$$17 = 16 + 1$$

Also gilt $1 = 17 \cdot 157 - 2668$ und damit $d = 157$. Der Benutzer veröffentlicht den öffentlichen Schlüssel $(17, 2773)$ und hält den geheimen Schlüssel 157 geheim.

1.12 Bemerkung (Verschlüsselung)
Sei der Klartext m eine ganze Zahl mit $0 \leq m < n$. Dann lautet das zugehörige Geheimtextelement c

$$c \equiv m^e \pmod{n}.$$

1.13 Bemerkung (Entschlüsselung)
Sei c das gegebene Geheimtextelement, also eine ganze Zahl mit $0 \leq c < n$. Dann lautet das zugehörige Klartextelement

$$m \equiv c^d \pmod{n}.$$

1.14 Bemerkung
Warum gilt die Beziehung

$$c^d \equiv (m^e)^d \equiv m^{ed} \equiv m \pmod{n}?$$

1.15 Proposition
Seien p, q, n, e und d wie vorhin erklärt. Dann gilt für alle Klartextelemente $m \in \mathbb{Z}$:

$$c \equiv m^e \pmod{n} \;\Rightarrow\; c^d \equiv m \pmod{n}.$$

Beweis. Wir merken zunächst an, dass $ed \equiv 1 \pmod{\varphi(n)}$ ist. Es existiert also eine ganze Zahl k, für die $ed = 1 + k \cdot \varphi(n)$ ist. Weiters bemerken wir, dass gilt:

$$m^{p-1} \equiv \begin{cases} 1 \pmod{p}, & \text{falls } (m,p) = 1 \\ 0 \pmod{p}, & \text{falls } (m,p) > 1 \end{cases}$$

Erheben wir nun m^{p-1} zur $k \cdot (q-1)$-ten Potenz, so erhalten wir:

$$m^{k \cdot \varphi(n)} \equiv \begin{cases} 1 \pmod{p}, & \text{falls } (m,p) = 1 \\ 0 \pmod{p}, & \text{falls } (m,p) > 1 \end{cases}$$

Nun ist $(m,p) > 1$ genau dann, wenn $(m,p) = p$, da p prim ist. Es folgt in diesem Fall $m \equiv 0 \pmod{p}$. Auf analoge Weise läßt sich zeigen, dass die Äquivalenz $(m,q) > 1 \Leftrightarrow m \equiv 0 \pmod{q}$ gilt. Damit folgt

$$m^{ed} \equiv m^{1+k \cdot \varphi(n)} \equiv m \pmod{p}$$
$$m^{ed} \equiv m \pmod{q}.$$

Mit Hilfe der Rechnenregeln für Kongruenzen erhalten wir $m^{ed} \equiv m \pmod{p \cdot q}$, also $m^{ed} \equiv m \pmod{n}$. □

1.16 Korollar
Die Abbildung $m \mapsto m^{ed}$ ist bijektiv auf der Menge $\mathbb{Z}_n$, denn aus $m^{ed} \equiv a^{ed} \pmod{n}$ folgt $m \equiv a \pmod{n}$.

Damit wir bei der Entschlüsselung des Geheimtextes c genau die Nachricht m bekommen und nicht nur eine Zahl m' (mod n), muß $0 \leq m < n$ gelten.

Es sei n gegeben. Wir ermitteln die eindeutig bestimmte Zahl i mit $10^i < n < 10^{i+1}$. Der Klartext wird nun in eine Folge von dekadischen Ziffern verwandelt, wobei jedem Klartextelement die gleiche Anzahl von Ziffern zugeordnet wird (z.B. mit $a \mapsto 00$, $b \mapsto 01$, ..., $z \mapsto 25$). Die Folge wird anschließend in Blöcke der Länge i aufgeteilt, sodaß jeder Ziffernblock $b_0 b_1 \dots b_{i-1} \in \{0, 1, \dots, 9\}^i$ als dekadische Darstellung einer Zahl m mit $0 \leq m < 10^i$ angesehen wird,

$$m = b_0 \cdot 10^0 + b_1 \cdot 10^1 + \cdots + b_{i-1} \cdot 10^{i-1}.$$

Wegen $0 \leq m < 10^i < n$ liegt m im gewünschten Zahlenbereich $\{0, 1, \dots, n-1\}$. Beim RSA-Verfahren erhalten wir daher beim Entschlüsseln wieder die Zahl m.

1.17 Beispiel
In der Praxis wird RSA als eine Art Blockchiffre angewendet. Wir sehen uns das Konzept an einem Sandkastenbeispiel mit $p = 47$ und $q = 59$ an. Diese Werte für p und q kennen wir aus Beispiel 1.11, $n = 2773$, $e = 17$ und $d = 157$. Wir bestimmen die passende Blocklänge. Es gilt $10^3 < 2773 < 10^4$, also wäre die Blocklänge nach unserer Festlegung 3. Da aber gilt: je größer die Blocklänge, umso schneller ist die Verschlüsselung, überlegen wir uns, daß in diesem Fall auch die Blocklänge 4 funktioniert.

Dazu legen wir Die Zuordnung Text → Zahlen wie folgt fest:

$a \mapsto 00$	$c \mapsto 02$	$z \mapsto 25$
$b \mapsto 01$	usw.	$\_ \mapsto 26$

Die Zuordnung in Blöcken zu je vier Ziffern funktioniert folgendermaßen:

h	a	s	t	a		l	a		v	i	s	t	a
↓	↓	↓	↓	↓	↓	↓	↓	↓	...				
07	00	18	19	00	26	11	00	26					

Die größtmögliche Zahl, die hier auftreten kann, ist die Zahl 2626. Sie ist damit kleiner als der Modul $n = 2773$. Daher gilt stets $0 \leq m < n$, wir können daher mit der Blocklänge 4 arbeiten, ohne die Bijektivität zu gefährden. Die blockweise Verschlüsselung liefert abschließend die Folge der Geheimtextblöcke:

0700	1819	0026	1100	
↓	↓	↓	↓	...
0761	0818	1445	0778	

1.18 Bemerkung (Sicherheit von RSA)
Die Sicherheit von RSA beruht darauf, dass die Berechnung der Zahl $\varphi(n)$ schwierig ist. Man benötigt nach dem derzeitigen Stand des Wissens die Faktorisierung der Zahl n.

Public Key-Verfahren sind ein großes Geschäft. Die Firma RSA hatte jahrelang Preise ausgesetzt für diejenigen, die große Zahlen faktorisieren können (siehe `http://www.rsalabs.com`).

1.3 Große Primzahlen

In Zusammenhang mit den meisten asymmetrischen Systemen benötigt man große Primzahlen. Wir diskutieren in diesem Kapitel, wie viele solche Zahlen es gibt.

1.19 Frage
Wie groß ist die Wahrscheinlichkeit, daß eine ungerade Zahl prim ist, wenn ihre Binärdarstellung die Länge L hat?

Wir wiederholen dazu einige Eigenschaften der Binärentwicklung.

1.20 Bemerkung (Binärentwicklung)
Jede nichtnegative ganze Zahl g kann *eindeutig* in der Form

$$g \;=\; g_0 \cdot 2^0 + g_1 \cdot 2^1 + g_2 \cdot 2^2 + \ldots \;=\; \sum_{j=0}^{\infty} g_j \cdot 2^j \tag{1.3}$$

mit Ziffern $g_j \in \{0,1\}$, $j = 0, 1, \ldots$ dargestellt werden. Wir schreiben dann

$$g \;=\; (g_0, g_1, g_2, \ldots) \tag{1.4}$$

Die Darstellung (1.3) ist natürlich *endlich*. Ab einer Stelle sind alle Ziffern g_j gleich Null.

Eine nichtnegative ganze Zahl g besitzt L Bits in der Binärdarstellung genau dann, wenn die Binärdarstellung von g die Form

$$g \;=\; (g_0, g_1, \ldots, g_{L-2}, 1, \dot{0})$$

hat, wobei $g_0, \ldots g_{L-2}$ beliebig sind. Hier bezeichnet $\dot{0}$ die periodische Folge $(0, 0, \ldots)$.

1.21 Bemerkung (Übereinkunft)
Für $g = (g_0, g_1, \ldots, g_{L-2}, 1, \dot{0})$ schreiben wir ab nun $g = (g_0, g_1, \ldots, g_{L-2}, 1)$. Wir lassen also die überflüssigen Nullen weg.

1.22 Korollar
Es gibt genau 2^{L-2} ungerade Zahlen g mit L Bits. Dies ist leicht einzusehen: die Binärdarstellung jeder Zahl g mit L Bits hat die Gestalt

$$g = g_0 + 2 \cdot (g_1 + g_2 \cdot 2 + \ldots + g_{L-2} \cdot 2^{L-3} + 2^{L-2}).$$

(Beachten Sie, dass 2 herausgehoben wurde!)

Da g ungerade ist, muß $g_0 = 1$ gelten. Die Binärdarstellung von g hat also die Form

$$g \;=\; (1, g_1, \ldots, g_{L-2}, 1).$$

In dieser Darstellung sind die $L-2$ Bits $g_1, \ldots, g_{L-2}$ frei wählbar. Dies ergibt 2^{L-2} Möglichkeiten für g.

1.23 Lemma
Es gilt

$$g \;=\; (g_0, g_1, \ldots, g_{L-1}) \;\Leftrightarrow\; 0 \le g < 2^L.$$

Beweis. Dies ist leicht einzusehen: die Zuordnung

$$(g_0, \ldots, g_{L-1}) \;\mapsto\; \sum_{j=0}^{L-1} g_j \cdot 2^j$$

ist eine injektive Abbildung von $\{0,1\}^L$ in die Menge $\{0,1,\ldots,2^{L-1}-1\} = \{g \in \mathbb{Z} : 0 \leq g < 2^L\}$. Da es sich um endliche Mengen handelt, ist diese Abbildung sogar bijektiv. Somit entspricht jeder Zahl g, $0 \leq g < 2^L$, umkehrbar eindeutig ein binärer Vektor $(g_0, g_1, \ldots, g_{L-1})$. □

1.24 Korollar
Eine nichtnegative ganze Zahl g hat L Bits genau dann, wenn $2^{L-1} \leq g < 2^L$.

Dies ist einfach einzusehen. Es gilt

$$g \;=\; (g_0, g_1, \ldots, g_{L-2}, 1) \;\Rightarrow\; 2^{L-1} \leq g \leq 2^L - 1 < 2^L.$$

Sei umgekehrt $2^{L-1} \leq g < 2^L$. Dann folgt $0 \leq g - 2^{L-1} < 2^L - 2^{L-1} = 2^{L-1}$. Mit Hilfe von Lemma 1.23 schließen wir daher

$$g - 2^{L-1} \;=\; (g_0, g_1, \ldots, g_{L-2}).$$

Daraus folgt offensichtlich

$$g \;=\; (g_0, g_1, \ldots, g_{L-2}, 1).$$

1.25 Korollar
Es gilt für alle $L \geq 3$:

$$\begin{aligned} p \text{ prim, mit } L \text{ Bits } &\Leftrightarrow 2^{L-1} \leq p < 2^L \\ &\Leftrightarrow 2^{L-1} < p \leq 2^L. \end{aligned}$$

Dies ist trivial, da p wegen $L \geq 3$ eine ungerade Primzahl ist. Daher ist der Fall $p = 2^k$ unmöglich.

1.26 Definition (Die Funktion $\pi(x)$)
Für eine relle Zahl x sei $\pi(x)$ die Anzahl der Primzahlen kleiner oder gleich x.

1.27 Korollar
Für die Anzahl der Primzahlen mit $L \geq 3$ Bits gilt folgende Beziehung:

$$\text{Anzahl der Primzahlen mit } L \text{ Bits} = \pi(2^L) - \pi(2^{L-1}).$$

Die Begründung ist einfach. Jede Primzahl p mit L Bits erfüllt ja die Ungleichung $2^{L-1} < p \leq 2^L$, siehe Korollar 1.25. Im Intervall $]2^{L-1}, 2^L]$ liegen $\pi(2^L) - \pi(2^{L-1})$ Primzahlen.

1.28 Korollar
Die Wahrscheinlichkeit p_L, dass eine zufällig gewählte ungerade Zahl mit L Bits in ihrer Binärdarstellung prim ist, ist gleich der Zahl

$$\frac{\pi(2^L) - \pi(2^{L-1})}{2^{L-2}}.$$

Dies ist klar: im Zähler dieses Bruches steht die Anzahl der Primzahlen mit L Bits, im Nenner die Anzahl der ungeraden Zahlen mit L Bits.

Es stellt sich nun die Frage nach der Größe der Zahl $\pi(2^L) - \pi(2^{L-1})$. Es ist derzeit praktisch unmöglich, diese Zahl exakt zu bestimmen, aber wir können sie sehr gut abschätzen.

1.29 Bemerkung (Verteilung der Primzahlen)
Die Funktion $\pi(x)$ ist seit Jahrhunderten Objekt intensiver zahlentheoretischer Forschung. Es gilt der berühmte *Primzahlsatz*, der von Hadamard und de la Vallée-Poussin im Jahr 1896 bewiesen wurde:

$$\lim_{x\to\infty} \frac{\pi(x)}{x/\ln x} = 1.$$

Auf Tschebyscheff geht die Ungleichung

$$0.92 \cdot \frac{x}{\ln x} < \pi(x) < 1.11 \cdot \frac{x}{\ln x}$$

zurück, die für alle hinreichend großen x erfüllt ist. Tschebyscheff bewies diese Art von Ungleichung bei seinen Versuchen, einen Beweis für den Primzahlsatz zu finden.

Wir werdem im Folgenden eine Ungleichung von Rosser und Schoenfeld [RS62] verwenden, die für alle $x > 67$ gilt:

$$\frac{x}{\ln x - \frac{1}{2}} < \pi(x) < \frac{x}{\ln x - \frac{3}{2}}.$$

1.30 Lemma
Es gilt

$$2^{L-1} \cdot \operatorname{ld} \mathrm{e} \cdot \frac{L - 2 - \frac{5}{2}\operatorname{ld}\mathrm{e}}{\left(L - \frac{1}{2}\operatorname{ld}\mathrm{e}\right)\left(L - 1 - \frac{3}{2}\operatorname{ld}\mathrm{e}\right)} <$$
$$< \pi(2^L) - \pi(2^{L-1}) < 2^{L-1} \cdot \operatorname{ld} \mathrm{e} \cdot \frac{L - 2 + \frac{1}{2}\operatorname{ld}\mathrm{e}}{\left(L - \frac{3}{2}\operatorname{ld}\mathrm{e}\right)\left(L - 1 - \frac{1}{2}\operatorname{ld}\mathrm{e}\right)}.$$

Hier bezeichnet ld den Logarithmus zur Basis zwei (*logarithmus dualis*).

Beweis. Aufgrund des Resultates von Rosser und Schönfeld gelten die untere Abschätzung

$$\frac{2^L}{\ln 2^L - \frac{1}{2}} - \frac{2^{L-1}}{\ln 2^{L-1} - \frac{3}{2}} < \pi(2^L) - \pi(2^{L-1})$$

und die obere Abschätzung

$$\pi(2^L) - \pi(2^{L-1}) < \frac{2^L}{\ln 2^L - \frac{3}{2}} - \frac{2^{L-1}}{\ln 2^{L-1} - \frac{1}{2}}.$$

Mittels Umrechnung des natürlichen Logarithmus auf den Logarithmus zur Basis zwei und der Beziehung $\ln x = \operatorname{ld} x / \operatorname{ld}\mathrm{e}$, $x \in \mathbb{R}$, $x > 0$, folgt

$$2^{L-1} \cdot \operatorname{ld} \mathrm{e} \cdot \left(\frac{2}{L - \frac{1}{2}\operatorname{ld}\mathrm{e}} - \frac{1}{L - 1 - \frac{3}{2}\operatorname{ld}\mathrm{e}}\right) <$$
$$< \pi(2^L) - \pi(2^{L-1}) < 2^{L-1} \cdot \operatorname{ld} \mathrm{e} \cdot \left(\frac{2}{L - \frac{3}{2}\operatorname{ld}\mathrm{e}} - \frac{1}{L - 1 - \frac{1}{2}\operatorname{ld}\mathrm{e}}\right).$$

Als Endergebnis erhalten wir die behaupteten Abschätzungen der Anzahl der Primzahlen mit L Bits nach oben und unten. □

1.31 Korollar
Sei p_L die Wahrscheinlichkeit, daß eine ungerade Zahl mit L Bits in ihrer Binärdarstellung prim ist. Mit Hilfe der Abschätzung von Lemma 1.30 erhalten wir die Abschätzung

$$2 \cdot \ln 2 \cdot \frac{L - 2 - \frac{5}{2}\ln 2}{\left(L - \frac{1}{2}\ln 2\right)\left(L - 1 - \frac{3}{2}\ln 2\right)} < \\ < p_L < 2 \cdot \ln 2 \cdot \frac{L - 2 + \frac{1}{2}\ln 2}{\left(L - \frac{3}{2}\ln 2\right)\left(L - 1 - \frac{1}{2}\ln 2\right)}$$

Dieses Resultat hat eine praktische Bedeutung. Für gegebenes L (beispielsweise $L = 512$) kann man damit abschätzen, wie lange man bei der Schlüsselerzeugung von RSA und ähnlichen Verfahren warten muß, bis die benötigten Primzahlen gefunden sind.

In Tabelle 1.1 geben wir für häufig benutzte Bitlängen untere und obere Schranken für die entsprechenden Wahrscheinlichkeiten an.

Bitlänge L	untere Schranke	obere Schranke
256	0.011194	0.011387
384	0.007480	0.007565
512	0.005616	0.005664
640	0.004496	0.004527
768	0.003749	0.003770
1024	0.002813	0.002825
2048	0.001408	0.001411
4096	0.000704	0.000705

Tabelle 1.1: Schranken für p_L

1.4 Der Diffie-Hellman Schlüsselaustausch

Dieses Verfahren von Diffie und Hellman dient zum Austausch eines geheimen Schlüssels über unsichere Leitungen. Um es erklären zu können, benötigen wir den Begriff des *diskreten Logarithmus*.

1.32 Bemerkung (Zyklische Gruppe)
Sei $(G, \cdot)$ eine Gruppe und e das neutrale Element. Für ein Gruppenelement a und für $n \in \mathbb{N}$ definieren wir

$$a^n := \underbrace{a \cdot a \cdot \dots \cdot a}_{n\text{-mal}} \qquad a^0 := e \qquad a^{-n} := \underbrace{a^{-1} \cdot a^{-1} \cdot \dots \cdot a^{-1}}_{n\text{-mal}},$$

wobei a^{-1} das inverse Element zu a bezeichnet. Die Menge $\langle a \rangle = \{a^k : k \in \mathbb{Z}\}$ bildet dann eine Untergruppe von $(G, \cdot)$. Sie heißt die von a erzeugte zyklische Untergruppe von $(G, \cdot)$.

Eine Gruppe $(G, \cdot)$ heißt zyklisch, wenn ein (erzeugendes) Element $a \in G$ existiert mit $\langle a \rangle = G$.

1.33 Beispiel
Sei $(G, \cdot) = (\mathbb{Z}_7^*, \cdot)$. Wir erinnern an die Definition der Gruppe der primen Restklassen modulo m, $\mathbb{Z}_7^* = \{\overline{a} : (a, 7) = 1\} = \{\overline{1}, \overline{2}, \overline{3}, \overline{4}, \overline{5}, \overline{6}\}$. Es gilt zum Beispiel

$$\begin{aligned}
\langle \overline{1} \rangle &= \{\overline{1}\} \\
\langle \overline{2} \rangle &= \{\overline{2}^0 = \overline{1}, \overline{2}^1 = \overline{2}, \overline{2}^2 = \overline{4}, \overline{2}^3 = \overline{8} = \overline{1}\} = \{\overline{1}, \overline{2}, \overline{4}\} \\
\langle \overline{3} \rangle &= \{\overline{1}, \overline{3}, \overline{2}, \overline{6}, \overline{4}, \overline{5}, \} = \mathbb{Z}_7^*
\end{aligned}$$

Somit ist $\mathbb{Z}_7^*$ zyklisch und $\overline{3}$ ist ein erzeugendes Element von $\mathbb{Z}_7^*$.

1.34 Bemerkung
Allgemein läßt sich über $\mathbb{Z}_p^*$ mit p prim sagen:

1. $\mathbb{Z}_p^*$ ist zyklisch.
2. Es gibt genau $\varphi(p-1)$ erzeugende Elemente in $\mathbb{Z}_p^*$.

1.35 Definition (Primitivwurzeln)
Sei m eine natürliche Zahl größer gleich 2. Unter einer Primitivwurzel modulo m versteht man eine ganze Zahl g mit den Eigenschaften

1. $(g, m) = 1$, d.h. die Zahlen g und m sind relativ prim.
2. Die Restklasse $\overline{g}$ ist ein erzeugendes Element von $\mathbb{Z}_m^*$.

1.36 Beispiel
Die ganze Zahl 3 ist eine Primitivwurzel modulo 7, ebenso wie -4 und 31 Primitivwurzeln modulo 7 sind, da $\overline{-4} = \overline{31} = \overline{3}$.

Für jede ganze Zahl A mit $(A, 7) = 1$ existiert eine eindeutig bestimmte Zahl a mit $0 \leq a \leq 5$, sodaß

$$A = 3^a \pmod 7$$

Da A und 7 teilerfremd sind, ist $\overline{A} \in \mathbb{Z}_7^*$. Wie wir zuvor gesehen haben, ist $\overline{3}$ ein erzeugendes Element von $\mathbb{Z}_7^*$. Folglich existiert ein a mit $0 \leq a \leq 5$, für das $\overline{3}^a = \overline{A}$ gilt. Die Zahl a ist sozusagen der „Logarithmus von A zur Basis 3 modulo 7“.

1.37 Definition (Diskreter Logarithmus)
Sei p prim, sei g eine Primitivwurzel modulo p und sei A eine ganze Zahl mit $(A, p) = 1$. Unter dem diskreten Logarithmus von A zur Basis g verstehen wir die eindeutig bestimmte Zahl $a \in \{0, 1, \ldots, p-2\}$ mit der Eigenschaft

$$A \equiv g^a \pmod p .$$

Bezeichnung: $a = \log_g(A) \bmod p$

1.38 Bemerkung
Wegen der Bedingung $(A, p) = 1$ liegt $\overline{A}$ in der primen Restklassengruppe $\mathbb{Z}_p^*$. Da p prim ist, gibt es eine Primitivwurzel g modulo p. Weiters existiert wegen

$$\langle \overline{g} \rangle = \{\overline{g}^0, \overline{g}^1, \overline{g}^2, \ldots, \overline{g}^{p-2}\} = \mathbb{Z}_p^*$$

ein eindeutig bestimmter Exponent a mit $\overline{A} = \overline{g}^a$. Dies ist gleichwertig zur Aussage $A \equiv g^a \pmod{p}$.

1.39 Beispiel
Tabelle 1.2 gibt die Werte des diskreten Logarithmus zur Basis 3 modulo 7 an.

A	1	2	3	4	5	6
$\log_3(A) \bmod 7$	0	2	1	4	5	3

Tabelle 1.2: Werte von $\log_3 \bmod 7$

1.40 Bemerkung (Schlüsselaustausch nach Diffie-Hellman)
Der Schlüsselaustausch nach Diffie und Hellman sieht wie folgt aus:

1. Alice und Bob vereinbaren öffentlich eine große Primzahl p und eine Primitivwurzel g modulo p.
2. Alice erzeugt einen zufälligen Exponenten $a \in \{2, 3, \ldots, p-2\}$, berechnet die Zahl $A \equiv g^a \pmod{p}$ und schickt diese an Bob.
3. Bob erzeugt einen zufälligen Exponenten $b \in \{2, 3, \ldots, p-2\}$, berechnet die Zahl $B \equiv g^b \pmod{p}$ und schickt sie an Alice.
4. Alice bildet die Zahl $B^a \equiv g^{ab} \pmod{p}$.
5. Bob bildet die Zahl $A^b \equiv g^{ab} \pmod{p}$.
6. Die Zahl $k \equiv B^a \equiv A^b \equiv g^{ab} \pmod{p}$ ist der gemeinsame geheime Schlüssel für Bob und Alice.

1.41 Bemerkung (Sicherheit)
Eve erhält durch Mitlauschen im unsicheren Kanal (z.B. Internet) die Werte p, g, A und B. Daraus kann sie die für die Entschlüsselung benötigten Exponenten a und b jedoch nicht ohne weiteres berechnen.

Die Berechnung diskreter Logarithmen gilt als ein extrem aufwendiges Problem. Bisher sind keine effizienten Algorithmen gefunden worden, trotz intensiver Forschung.

Es gibt neben den besprochenen noch weitere asymmetrische Verfahren, worunter viele auf diskreten Logarithmen beruhen. Beispiele sind das El Gamal-Verfahren und der Digital Signature Algorithm (DSA).

Literaturverzeichnis

[DH76] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Trans. Inform. Theory*, IT-22:644–654, 1976.

[MvOV97] A. J. Menezes, P. C. van Oorschot, and S.A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Boca Raton, 1997.

[RS62] J. B. Rosser and L. Schönfeld. Approximate formulas for some functions of prime numbers. *Illinois J. Math.*, **6**:64–94, 1962.

[Sal97] A. Salomaa. *Public Key Cryptography*. Springer-Verlag, New York, 1997.

[Sch96] B. Schneier. *Applied Cryptography*. Wiley, New York, second edition, 1996.

[Sti06] D. R. Stinson. *Cryptography*. Chapman and Hall/CRC Press, Boca Raton, 3rd edition, 2006.

[TW06] W. Trappe and L. Washington. *Introduction to Cryptography with Coding Theory*. Pearson Prentice Hall, 2nd edition, 2006.