Humboldt Universität zu Berlin — *Institut für Informatik*
Lehrstuhl für Algorithmen und Komplexität II

Einwegfunktionen
Variationen und Beispiele

Von

Lukas Dölle

doelle@informatik.hu-berlin.de

Seminar „Perlen der Theoretischen Informatik“

Leitung: Prof. Dr. Johannes Köbler, Olaf Beyersdorff

Berlin, Dezember 2002

Inhaltsverzeichnis

1 Einführung

1.1 Public Key Kryptosysteme

Krytpologie ist die Wissenschaft der Geheimhaltung von Informationen durch Transformation von Daten. Unter dem Oberbegriff Kryptologie fasst man heute die beiden Teilgebiete *Kryptographie* und *Kryptoanalyse* zusammen. Im Rahmen der Kryptographie werden Kryptosysteme entwickelt, deren kryptographische Stärke mit Hilfe der Methoden der Kryptoanalysis zum Brechen von Kryptosystemen beurteilt werden kann.
Die Kryptosysteme unterteilt man in symmetrische und asymmetrische Kryptosysteme. Bei den symmetrischen Kryptosystemen verwendet man den gleichen Schlüssel k zum Verschlüsseln der Nachricht und zum Entschlüsseln des Schlüsseltextes. Dieser Schlüssel muss geheim gehalten werden, da jeder Unautorisierte mit Hilfe des Schlüssels k den Schlüsseltext decodieren kann.
Bei den asymmetrischen Kryptosytemen verwendet man zwei Schlüssel, einen privaten (streng geheimen) und einen öffentlichen Schlüssel. Deshalb werden sie auch *Public Key Kryptosysteme* (PKK) genannt. Wie der Name schon sagt, benutzen PKKs nicht geheime, sondern öffentliche Schlüssel zum Verschlüsseln der Texte. Hierbei werden Funktionen benutzt, welche leicht zu berechnen sind, aber ohne zusätzliches Wissen nicht invertierbar sind. Diese so genannten *Einwegfunktionen* können dann öffentlich bekannt gegeben werden, ohne die Geheimhaltung der Nachricht zu gefährden. Da die Funktionen schwierig zu invertieren sind, ist das Entschlüsseln nicht einfach möglich. Es gilt also:

- Jeder kann mit dem öffentlichen Schlüssel Meldungen codieren.
- Nur wer den geheimen Schlüssel kennt, kann die Meldung decodieren.

Da für diese Verfahren zwei verschiedene Schlüssel zum Ver- bzw. Entschlüsseln verwendet werden, werden sie asymmetrische Verfahren genannt.

Definition (Public Key Kryptosystem):
Für ein Public Key Kryptosystem müssen die folgenden Bedingungen erfüllt sein:

- Es gibt genügend viele Paare (E, D) von Verschlüsselungs- und Entschlüsselungsfunktionen (bzw. von öffentlichen und geheimen Schlüsseln (e, d)).
- Für jede Meldung m gilt $D(E(m)) = m$.
- E ist eine Einwegfunktion.
- E und D sind leicht zu berechnen, wenn man den Schlüssel e, bzw. d kennt.

1.2 Einwegfunktionen

Definition (Einwegfunktion):
Eine Funktion f: $X \rightarrow Y$ heißt Einwegfunktion (*one-way function*) wenn gilt:

- Es gibt ein effizientes Verfahren zur Berechnung von $y = f(x)$ für jedes $x \in X$.
- Es gibt kein effizientes Verfahren zur Berechnung von $x = f^{-1}(y)$ für jedes $y \in Y$ (bis auf vernachlässigbar viele).

Ein $y = f(x)$ ist demnach für jedes $x \in X$ „leicht“ zu berechnen, während es für alle $y \in Y$ (bis auf vernachlässigbar viele) „schwer“ ist, ein x zu bestimmen, so dass $x = f^{-1}(y)$ ist. Ein Rechner braucht beispielsweise nur einige Sekunden, um den Funktionswert $f(x)$ für einen Wert x zu berechnen, für die Umkehrung braucht er jedoch möglicherweise Monate oder sogar Jahre.
Auf der Grundlage dieser Definition, lassen sich verschiedene Kandidaten für Einwegfunktionen bestimmen. Da es nicht bewiesen ist, ob Einwegfunktionen überhaupt existieren, ist es

auch nicht bekannt, ob diese Funktionen tatsächlich Einwegfunktionen sind. Es sind somit nur Kandidaten für Einwegfunktionen.

Beispiel 1:

$f_{Telefon}$: Name → Nummer aus dem Telefonbuch

Ein einfaches Modell für eine Einwegfunktion ist ein Telefonbuch, mit welchem sehr schnell zu jedem Namen mit Adresse die Telefonnummer gefunden werden kann. Der Aufwand der Suche in dieser geordneten Liste ist O(log n). Hingegen ist es sehr aufwendig, mit Hilfe eines Telefonbuchs zu einer Telefonnummer den zugehörigen Namen zu finden. Es muss dafür das ganze Buch durchsucht werden (ungeordnete Liste: O(n)).

Beispiel 2:

f_{Coca_Cola}: Zutaten → Coca Cola

Aus der Analyse des Getränks „Coca Cola“ kann man nur sehr grob auf die Zutaten schließen.

Beispiel 3:

$f_{Stradivari}$: Konstruktionsanweisung → Stradivari (Violine)

Man hat es seit einigen hundert Jahren nicht geschafft, eine Violine von der Qualität einer Stradivari nachzubauen – trotz vielfacher Analysen.

Eine exaktere Definition von Einwegfunktionen ist folgende:

Definition ((starke) Einwegfunktion):

Eine Funktion f: $\{0, 1\}^* \rightarrow \{0, 1\}^*$ heißt (starke) Einwegfunktion, wenn die folgenden zwei Bedingungen erfüllt sind:

1. Leicht zu berechnen: Es existiert ein (deterministischer) Polynomialzeit-Algorithmus A, so dass bei Eingabe x der Algorithmus A $f(x)$ ausgibt (d. h. $A(x) = f(x)$).
2. Schwer zu invertieren: Für jeden probabilistischen Algorithmus A', jedes positive Polynom p und alle hinreichend große n gilt:

$$P\left(A'(f(U_n),1^n) \in f^{-1}(f(U_n))\right) < \frac{1}{p(n)}$$

wobei U_n eine Zufallsvariable ist, die gleichmäßig über $\{0, 1\}^n$ verteilt ist.

Diese Einwegfunktionen sind stark einweg, da jeder effiziente Invertierungsalgorithmus vernachlässigbaren Erfolg im Invertieren hat. Eine schwächere Definition fordert nur, dass jeder effiziente Invertierungsalgorithmus mit einiger erkennbarer Wahrscheinlichkeit fehlschlägt.

Definition (schwache Einwegfunktion):

Eine Funktion f: $\{0, 1\}^* \rightarrow \{0, 1\}^*$ heißt schwache Einwegfunktion, wenn die folgenden zwei Bedingungen erfüllt sind:

1. Leicht zu berechnen: Es existiert ein (deterministischer) Polynomialzeit-Algorithmus A, so dass bei Eingabe x der Algorithmus A $f(x)$ ausgibt (d. h. $A(x) = f(x)$).
2. Etwas schwer zu invertieren: Es existiert ein Polynom p, so dass für jeden probabilistischen Algorithmus A', und alle hinreichend große n gilt:

$$P\left(A'(f(U_n),1^n) \notin f^{-1}(f(U_n))\right) > \frac{1}{p(n)}$$

wobei U_n eine Zufallsvariable ist, die gleichmäßig über $\{0, 1\}^n$ verteilt ist.

Hier wird also nur für ein Polynom p gefordert, dass jeder probabilistische Algorithmus mit mehr als $\frac{1}{p(n)}$ Wahrscheinlichkeit fehlschlägt, die Funktion f bei $f(U_n)$ zu invertieren.

Satz:

Es existiert eine polynomiell berechenbare Funktion, die eine (starke) Einwegfunktion ist, genau dann, wenn Einwegfunktionen existieren.

2 Variationen von Einwegfunktionen

In diesem Kapitel wird zuerst eine alternative Formulierung für Einwegfunktionen präsentiert. Sie ist eher geeignet, um viele natürliche Kandidaten für Einwegfunktionen zu beschreiben. Dann wird die Formulierung benutzt, um zusätzliche Eigenschaften von Einwegfunktionen zu präsentieren. Besonders interessant sind Einwegfalltürfunktionen und Klauenfreie Funktionspaare (claw-free function pairs).

2.1 Familien von Einwegfunktionen

Die Definitionen von Einwegfunktionen, die bis jetzt benutzt wurden, ist für eine abstrakte Betrachtung geeignet. Um viele natürliche Kandidaten von Einwegfunktionen zu beschreiben, ist die folgende Formulierung (obwohl sie eher unhandlich ist) brauchbarer.
Anstatt Einwegfunktionen als Funktionen zu sehen, die auf einem unendlichen Bereich (d. h. {0, 1}*) operieren, betrachtet man unendliche Familien von Funktionen, die jeder auf einem endlichen Bereich operieren. Die Formulierung von einer Familie von Funktionen ist auch für die Präsentation von Einwegfalltürfunktionen und Klauenfreie Funktionen (claw-free functions) nützlich.

Definition (Familie von Funktionen):

Eine Familie von Funktionen besteht aus einer unendlichen Menge von Indizes, benannt mit $\bar{I}$, und einer entsprechenden Menge von endlichen Funktionen, benannt mit $\{f_i\}_{i\in\bar{I}}$. Für jedes $i \in \bar{I}$ ist der Definitionsbereich der Funktion f_i, benannt mit D_i, eine endliche Menge.

Eine notwendige Bedingung, um eine Familie von Funktionen zu benutzen, ist die Existenz von einer effizienten Funktion (Evaluationsalgorithmus) *F*, die bei Eingabe $i \in \bar{I}$ und $x \in D_i$ als Ausgabe $f_i(x)$ zurückgibt. Jedoch reicht das noch nicht aus. Man muss auch in der Lage sein, zufällig einen Index auszusuchen, der eine Funktion bezeichnet, und zufällig ein Element aus dem Definitionsbereich (wenn der Index gegeben ist) auszusuchen. Dazu braucht man einen Algorithmus *I*, der bei Eingabe einer ganzen Zahl *n* (in unärer Schreibweise) zufällig einen Index ausgibt, der eine Funktion und seinen dazugehörigen Definitionsbereich beschreibt. Ein Algorithmus *D* gibt bei Eingabe des Index *i* zufällig ein Element aus D_i aus. Die Einwegeigenschaft wird dadurch erfasst, dass verlangt wird, dass jeder effiziente Algorithmus, der einen Index einer Funktion und ein Element deren Definitionsbereichs bekommt, die Funktion nicht umkehren kann, außer mit geringfügiger vernachlässigbarer Wahrscheinlichkeit.
All das wird durch folgende Definition erfasst.

Definition (Familie von Einwegfunktionen):

Eine Familie von Funktionen $\{f_i : D_i \to \{0,1\}^*\}_{i\in\bar{I}}$ wird Familie starker Einwegfunktionen genannt, wenn drei probabilistische Polynomialzeit-Algorithmen *I*, *D* und *F* existieren, so dass die folgenden zwei Bedingungen erfüllt sind:

1. Leicht zu berechnen: Die Ausgangsverteilung des Algorithmus *I* bei Eingabe 1^n ist eine zufällige Variable, die einem Wert in der Menge $\bar{I} \cap \{0,1\}^n$ zugewiesen wird. Die Ausgangsverteilung des Algorithmus *D* bei Eingabe $i \in \bar{I}$ ist eine zufällige

Variable, die einem Wert in D_i zugewiesen wird. Bei Eingabe $i \in \bar{I}$ und $x \in D_i$ gibt Algorithmus F immer $f_i(x)$ zurück.

2. Schwer umzukehren: Für jeden Polynomialzeit-Algorithmus A', jedes positive Polynom p und hinreichend große n gilt:

$$P\left(A'(I_n, f_{I_n}(X_n)) \in f_{I_n}^{-1}(f_{I_n}(X_n))\right) < \frac{1}{p(n)}$$

wobei I_n eine zufällige Variable ist, die die Ausgabeverteilung von Algorithmus I bei Eingabe 1^n beschreibt und X_n eine zufällige Variable ist, die die Ausgabe von Algorithmus D bei Eingabe I_n beschreibt.

Die Ausgabe von Algorithmus I bei Eingabe 1^n muss nicht notwendigerweise gleichmäßig über $\bar{I} \cap \{0,1\}^n$ verteilt sein. Es kann sich sogar $I(1^n)$ auf einen einzelnen Index konzentrieren. Ebenso muss die Ausgabe von Algorithmus D, bei Eingabe i nicht notwendigerweise gleichmäßig über D_i verteilt sein. Jedoch impliziert die Schwer-umzukehren-Bedingung, dass sich die $D(i)$ nicht hauptsächlich auf polynomiell (in $|i|$) viele Indizes konzentrieren können. Die Familie ist schwer umzukehren, was die Verteilung anbetrifft, die durch die Algorithmen I und D verursacht wird (zusätzlich zur Abbildung der Funktion selber).
Mann kann Familie von Einwegfunktionen mit dem entsprechenden Tripel von Algorithmen beschreiben. Also kann man sagen, dass ein Tripel von probabilistischen Polynomialzeit-Algorithmen (I, D, F) eine Familie von Einwegfunktionen bildet, wenn eine Familie von Funktionen existiert, für welche die Algorithmen die vorherigen zwei Bedingungen erfüllen. Jede Familie von Einwegfunktionen kann als eine Einwegfunktion dargestellt werden und umgekehrt, aber jede Definition hat ihre eigenen Vorteile. Im folgenden wird die Formulierung von Familie von Einwegfunktionen benutzt, um beliebte Kandidaten für Einwegfunktionen zu präsentieren

Lockerungen:
Um eine weniger unhandliche Präsentation von natürlichen Kandidaten für Familien von Einwegfunktionen zuzulassen, kann man die Definition wie folgt lockern. Erstens kann man dem Algorithmus I erlauben bei Eingabe 1^n Indizes der Länge $p(n)$ auszugeben anstatt von n, wobei p ein beliebiges Polynom ist. Zweitens erlaubt man allen Algorithmen mit geringfügiger Wahrscheinlichkeit zu scheitern. Am wichtigsten erlaubt man dem Index-Algorithmus I einen Index auszugeben, der nicht in $\bar{I}$ ist, solange die Wahrscheinlichkeit dass $I(1^n) \notin \bar{I} \cap \{0,1\}^{p(n)}$ ist, geringfügig in n ist.

Zusätzliche Eigenschaften: Effizient erkennbare Indizes und Definitionsbereiche
Hier sind zwei zusätzliche (nützliche) Eigenschaften erwähnt, die einige Kandidaten für Familien von Einwegfunktionen besitzen. Erstens die Existenz einer effizient erkennbare Indexmenge und zweitens die Existenz einer effizient erkennbare Familie von Definitionsbereichen, d. h. die Existenz eines effizienten Algorithmus der bei Eingabe $i \in \bar{I}$ und x bestimmen kann, ob $x \in D_i$ oder $x \notin D_i$.

2.2 Einwegfunktionen mit Falltür

Echte Einwegfunktionen können im allgemeinen nicht unmittelbar zum Chiffrieren (mit anschließendem Dechiffrieren) von Nachrichten verwendet werden, wohl aber zum

- Identifizieren bzw. Autorisieren (z. B.: Ablage der UNIX-Passwörter)

- Schlüsselaustausch bzw. -vereinbarung (Beispiel: DIFFIE-HELLMAN-Schlüsselvereinbarung)

Für Chiffrierzwecke werden Einwegfunktionen mit einer „Falltür" benötigt, die ein effizientes Dechiffrieren ermöglichen. Das heißt, mit einer geheim zu haltenden Zusatzinformation ist es dem legalen Empfänger der Nachricht möglich, diese zu entschlüsseln, während es ohne diese Falltürinformation schwer ist, die Nachricht zu dechiffrieren.

Definition (Einwegfalltürfunktion):

Eine Funktion f: $X \to Y$ heißt Einwegfalltürfunktion (*trapdoor one-way function*), wenn gilt:

- $y = f(x)$ ist „leicht" mit einem effizienten Algorithmus E zu berechnen und
- $x = f^{-1}(y)$ ist „leicht" mit einem effizienten Algorithmus D zu berechnen.

Die Bestimmung von D aus E ist jedoch ohne eine geheim zu haltende Zusatzinformation (Falltür) schwer.

Auch hier kann aus der Definition der Familie von Funktionen eine Definition für eine Familie von Einwegfalltürfunktionen gewonnen werden. Das sind eine Familie von Einwegfunktionen $\{f_i\}$, mit der Zusatzinformation, dass f_i effizient invertiert werden kann, wenn eine Hilfseingabe für den Index i gegeben ist. Diese Falltür für dem Index i (notiert mit $t(i)$) kann nicht effizient aus i errechnet werden, jedoch kann man entsprechende Paare $(i, t(i))$ effizient generieren

Definition (Familie von Einwegfalltürfunktionen):

Sei I: $1^* \to \{0, 1\}^* \times \{0, 1\}^*$ ein probabilistischer Algorithmus und sei $I_1(1^n)$ das erste Element des Ausgabepaares von $I(1^n)$.

Ein Tripel von Algorithmen (I, D, F) wird Familie von starken (bzw. schwachen) Einwegfalltürfunktionen genannt, wenn die folgenden zwei Bedingungen erfüllt sind:

1. Die Algorithmen induzieren ein Familie von Einwegfunktionen: Das Tripel (I_1, D, F) stellt eine Familie von starken (bzw. schwachen) Einwegfunktionen dar.
2. Mit einer Falltür leicht invertierbar: Es existiert ein (deterministischer) Popynomialzeit-Algorithmus, notiert mit F^{-1}, so dass für jedes (i, t) im Bereich von I und für jedes $x \in D_i$ gilt: $F^{-1}(t, f_i(x)) = x$.

Eine nützliche Lockerung dieser Bedingungen ist es zu verlangen, dass diese nur mit einer überwältigenden hohen Wahrscheinlichkeit erfüllt sein müssen. Es darf nämlich der Indexgenerierende Algorithmus I mit vernachlässigbarer Wahrscheinlichkeit Paare (i, t) ausgeben, für die f_i keine Permutation darstellt oder $F^{-1}(t, f_i(x)) = x$ nicht für alle $x \in D_i$ erfüllt ist. Auf der anderen Seite wird typischerweise gefordert, dass der Definitionsbereichsalgorithmus (das heißt der Algorithmus D) eine fast gleichmäßige Verteilung auf dem entsprechenden Bereich produziert.

2.3 Klauenfreie Funktionen

Definition (Familie von Klauenfreie-Funktionen (*claw-free functions*)):

Eine Familie von Klauenfreie-Funktionen besteht aus einer unendlichen Menge von Indizes $\bar{I}$, zwei endlichen Mengen D_i^0 und D_i^1 für jedes $i \in \bar{I}$ und zwei Funktionen f_i^0 und f_i^1, die über D_i^0 bzw. D_i^1 definiert sind. So eine Familie wird klauenfrei (claw-free) genannt, wenn drei probabilistische Polynomialzeit-Algorithmen I, D und F existieren, so dass die folgenden zwei Bedingungen erfüllt sind:

1. Leicht zu berechnen: Die Zufallsvariable $I(1^n)$ ist Werten in der Menge $\bar{I} \cap \{0,1\}^n$ zugewiesen. Für jedes $i \in \bar{I}$ und $\sigma \in \{0,1\}$ ist die Zufallsvariable $D(\sigma, i)$ über D_i^σ verteilt und $F(\sigma, i, x) = f_i^\sigma(x)$ für jedes $i \in D_i^\sigma$.
2. Identische Verteilung: Für jedes i aus der Indexmenge $\bar{I}$ sind die Zufallsvariablen $f_i^0(D(0,i))$ und $f_i^1(D(1,i))$ identisch verteilt.
3. Schwer Klauen zu bilden: Ein Paar (x, y) mit $f_i^0(x) = f_i^1(y)$ wird Klaue für Index i genannt. Sei C_i die Menge aller Klauen für Index i. Dann wird gefordert, dass für jeden probabilistischen Polynomialzeit-Algorithmus A', jedes positive Polynom p und alle hinreichend große n gilt:

$$P\left(A'(I_n) \in C_{I_n}\right) < \frac{1}{p(n)}$$

wobei I_n eine Zufallsvariable ist, die die Ausgabeverteilung des Algorithmus I bei Eingabe 1^n beschreibt.

3 Kandidaten von Einwegfunktionen

In diesem Kapitel werden einige beliebte Kandidaten von Einwegfunktionen präsentiert. Diese bauen zum Teil auf der Elementaren Zahlentheorie auf. Wichtige Definitionen und Sätze der Elementaren Zahlentheorie sind im Kapitel 4 dargestellt.

3.1 Faktorisierungsproblem

Man betrachte folgende Funktion *f*:

$$f(x, y) = x \cdot y$$

Beispiel:

Sei $x = 43$ und $y = 79$. Wie man leicht sieht, sind 43 und 79 Primzahlen. $f(x, y) = x \cdot y$ ist relativ effizient zu berechnen.

$$f(x, y) = f(43, 79) = 43 \cdot 79 = 3397$$

Ist allerdings $f(x, y) = 3953$ gegeben, so ist es relativ schwer, eine Primzahlzerlegung dafür zu finden, das heißt $f^{-1}(x, y) = f^{-1}(3953)$ zu bestimmen. Als Lösung ergibt sich:

$$x = 59, y = 67$$

Im allgemein kann man zwei sehr große Primzahlen *p*, *q* (länger als 200 Dezimalstellen) relativ effizient multiplizieren.

$$n = p \cdot q$$

Die Zerlegung des Ergebnisses *n* in seine Primfaktoren ist ohne Kenntnis der Faktoren jedoch sehr zeit- und rechenaufwendig (*Faktorisierungsproblem*). Die heute schnellsten bekannten Algorithmen zerlegen eine zusammengesetzte ganze Zahl $n = p \cdot q$ (p, q prim) in einem Zeitraum proportional zu

$$L(n) = e^{\sqrt{\ln n \ln(\ln n)}}$$

Ein genauerer Wert ist $e^{(1{,}923+o(1))\sqrt[3]{\ln n(\ln \ln n)^2}}$. Solange kein besserer Algorithmus gefunden wird, heißt dies, dass Werte der Größenordnung 100 bis 200 Dezimalstellen zur Zeit sicher sind. Einschätzungen der aktuellen Computertechnik besagen, dass eine Zahl mit 100 Dezimalstellen bei vertretbaren Kosten in etwa zwei Wochen zerlegt werden könnte. Mit einer teuren Konfiguration (z. B. im Bereich von 10 Millionen US-Dollar) könnte eine Zahl mit 150 Dezimalstellen in etwa einem Jahr zerlegt werden. Eine 200-stellige Zahl dürfte noch für eine sehr lange Zeit unzerlegbar bleiben, falls es zu keinem mathematischen Durchbruch kommt. Zum Beispiel würde es etwa 1000 Jahre dauern, um eine 200-stellige Zahl mit den bestehenden Algorithmen in Primfaktoren zu zerlegen; dies gilt selbst dann, wenn 10^{12} Operationen pro Sekunde ausgeführt werden könnten, was jenseits der Leistung der heutigen Technik liegt und in der Entwicklung Milliarden von Dollar kosten würde. Dass es aber nicht doch schon morgen zu einem mathematischen Durchbruch kommt, kann man allerdings nie ausschließen.

3.2 Die RSA-Funktion

Das RSA-Verfahren ist nicht nur das älteste Public Key Kryptosystem, es ist heute wohl auch das am meisten verbreitete. So bildet RSA die Grundlage für SSL (Secure Socket Layer), welche vor allem für das WWW gebraucht werden, für SET (Secure Electronic Transactions), welche im Zusammenhang mit elektronischem Geld wichtig sind, für S/Mime, also sichere Email und vieles mehr (z. B. S-HTTP, SSH, ...).
Auf der Grundlage des Satzes von EULER-FERMAT haben die Kryptologen RON RIVEST, ADI SHAMIR und LEONARD ADLEMAN 1977 ein Verschlüsselungsverfahren für geheime Nachrichten entwickelt, dass nach erstem Buchstaben ihrer Nachnamen RSA-Verschlüsselungsverfahren genannt wird. 1983 meldete sie in den USA das Patent (US Patent, 4.405.829, 20.9.83 PKP) darauf an, welches 2000 auslief und aufgrund der Gesetzeslage in den USA nicht mehr verlängert werden konnte. Somit steht RSA allen zur Verfügung.
Der RSA-Algorithmus basiert auf dem mathematischen Problem der Faktorisierung großer Primzahlprodukte. Konkret heißt dies, dass man das Ergebnis, welches man durch die Multiplikation zweier großer Primzahlen (mit groß meint man Größenordnungen von mehreren hundert Stellen) erhält, sehr schwer wieder in die Ausgangsprimzahlen zerlegen kann. Grob gesagt kann man dies nur durch Ausprobieren aller möglicher Primzahlkombinationen erreichen, was bei diesen Größenordnungen so lange dauern würde, dass man diesen Angriff als praktisch unmöglich einstufen und das Verfahren als praktisch sicher ansehen kann. Die Sicherheit von RSA hängt damit ganz wesentlich an der Schwierigkeit der Faktorisierung von einer natürlichen Zahl n ab. Eine Äquivalenz der Probleme „Brechen von RSA“ und „Faktorisierung von n“ ist aber nicht bewiesen.

3.2.1 Der RSA-Algorithmus

Der Empfänger B einer Nachricht wählt zunächst zwei sehr große Primzahlen p und q aus (ungefähr gleich viele Stellen), bildet

$$n = p \cdot q$$

(n wird Modul genannt). Nun sucht er eine zu

$$\varphi(n) = (p - 1)(q - 1)$$

teilerfremde Zahl e mit $1 < e < (p - 1)(q - 1)$ (also ist $\mathrm{ggT}(\varphi(n), e) = 1$). Die Zahlen n und e gibt B öffentlich bekannt (öffentliche Schlüssel (n, e)), weil sie zur Verschlüsselung benötigt werden.
Will Absender A eine geheime Nachricht an den Empfänger B übermitteln, dann wird zunächst der Text der Nachricht in eine Ziffernfolge, bestehend aus gleichlangen Blöcken x (jeweils weniger als ca. 100 Stellen) mit $x \leq n - 1$, umgewandelt. Dann berechnet A den Rest m von x^e bei Division durch n:

$$m \equiv x^e \bmod n.$$

Der Absender A sendet die Zahl m an B, und zwar für jeden der aus dem Originaltext entstandenen Ziffernblöcke x. Der Empfänger kann die Nachricht m dechiffrieren, wenn er eine Lösung der linearen Kongruenz

$$d \cdot e \equiv 1 \bmod \varphi(n)$$

kennt (privater Schlüssel d), wobei gilt: $1 < d < (p - 1)(q - 1)$. d heißt dann die Inverse von e modulo $\varphi(n)$ und existiert, da e teilerfremd zu $\varphi(n)$ ist. Die Zahl x ist der Rest von m^d bei Division durch n:

$$x \equiv m^d \bmod n.$$

Falls erforderlich wandelt B nun noch die Ziffernfolge in Text um.

Verschlüsselungsoperation:

$$E\colon \{0, 1, ..., n-1\} \to \{0, 1, ..., n-1\}\ E(x) := x^e \text{ modulo } n = m$$

Entschlüsselungsoperation:

$$D\colon \{0, 1, ..., n-1\} \to \{0, 1, ..., n-1\}\ D(m) := m^d \text{ modulo } n = x$$

3.2.2 RSA als Einwegfunktion

Als nächstes soll gezeigt werden, dass die RSA-Funktion die erste Bedingung der Definition der Familie von Einwegfunktionen (also leicht zu berechnen ist) erfüllt. Dazu sind die drei Algorithmen gesucht: I_{RSA}, D_{RSA}, F_{RSA}.
Bei Eingabe 1^n wählt der Algorithmus I_{RSA} zwei Primzahlen p und q mit $2^{n-1} \le p < q < 2^n$ und eine natürliche Zahl e, die teilerfremd zu $(p-1)(q-1)$ ist, aus (e wird gleichmäßig unter den zulässigen Möglichkeiten ausgewählt). I_{RSA} gibt als Ausgabe (n, e) aus, wobei $n = p \cdot q$.
Für eine effiziente Implementation des Algorithmus I_{RSA} wird ein probabilistischer Polynomialzeit-Algorithmus benötigt, um gleichmäßig verteilte Primzahlen zu generieren. Solche Algorithmen gibt es.
Algorithmus D_{RSA} wählt bei Eingabe (n, e) ein Element aus der Menge $D_{n,e} = \{0, 1, ..., n-1\}$ aus und gibt dieses x aus.
Die Ausgabe von Algorithmus F_{RSA} bei Eingabe $((n, e), x)$ ist:

$$\text{RSA}_{n,e}(x) \stackrel{\text{def}}{=} x^e \bmod n$$

I_{RSA}: Eingabe: 1^n
Ausgabe: (n, e) mit $n = p \cdot q$; p, q prim; $2^{n-1} \le p < q < 2^n$; $\text{ggT}(e, \varphi(n)) = 1$

D_{RSA}: Eingabe: (n, e)
Ausgabe: x mit $x \in D_{n,e} = \{0, 1, ..., n-1\}$

F_{RSA}: Eingabe: $((n, e), x)$
Ausgabe: m mit $m \equiv x^e \bmod n$

Es wird weitgehend davon ausgegangen, dass RSA schwer zu invertieren ist.

Um RSA als Einwegfunktion mit Falltür zu definieren, muss der Algorithmus I_{RSA} modifiziert werden, so dass er neben (n, e) auch noch die Falltür (n, d) ausgibt, wobei gilt:

$$d \cdot e \equiv 1 \bmod \varphi(n)$$

Der Invertierungsalgorithmus F_{RSA}^{-1} ist identisch zum Algorithmus F_{RSA} (d. h.

$$F_{\text{RSA}}^{-1}((n,d),m) = m^d \bmod n\,.$$

Hier muss nur gezeigt werden, dass gilt:

$$F_{\text{RSA}}^{-1}((n,d), F_{\text{RSA}}((n,e),x)) = x^{ed} \bmod n \equiv x$$

bzw.

$$m^d \equiv x^{ed} \equiv x \bmod n$$

3.2.3 Beweis der Korrektheit

Es soll die Korrektheit des RSA-Algorithmus gezeigt werden, d. h. es ist zu zeigen:

$$m^d \equiv x \bmod n$$

wobei gilt:

$$n = p \cdot q$$
$$\varphi(n) = (p-1)(q-1)$$
$$d \cdot e \equiv 1 \bmod \varphi(n)$$
$$m \equiv x^e \bmod n$$

Beweis:

Als erstes folgt aus der Gleichung $d \cdot e \equiv 1 \bmod \varphi(n)$, dass ein $k \in \mathbb{Z}$ existiert mit:

$$d \cdot e \equiv 1 \bmod \varphi(n)$$
$$d \cdot e - 1 = k \cdot \varphi(n)$$
$$d \cdot e = k \cdot \varphi(n) + 1$$

Nun folgt:

$$m \equiv x^e \bmod n$$
$$m^d \equiv (x^e)^d \bmod n$$
$$\equiv x^{ed} \bmod n$$
$$\equiv x^{k\varphi(n)+1} \bmod n$$

Es gilt also $m^d \equiv x^{k\varphi(n)+1} \bmod n$. Das heißt, es ist nur noch zu zeigen:

$$x^{k\varphi(n)+1} \equiv x \bmod n$$

Hierzu müssen vier Fälle unterschieden werden.

1. Fall: ggT(x, n) = 1

Es gilt:

$$x^{k\varphi(n)+1} \equiv x \cdot (x^{\varphi(n)})^k \bmod n \quad | \text{ da ggT}(x,\ n) \ = \ 1 \text{ gilt der Satz von Euler - Fermat :}$$
$$\equiv x \cdot 1 \bmod n \quad | \ x^{\varphi(n)} \equiv 1 \bmod n$$
$$\equiv x \bmod n$$

2. Fall: ggT(x, n) = $n = p \cdot q$

Also gilt:

$$n = p \cdot q \mid x$$

(Da $0 \le x \le n - 1 = p \cdot q - 1$ folgt damit sogar $x = 0$.)

$$x \equiv 0 \bmod n$$
$$x^{k\varphi(n)+1} \equiv 0^{k\varphi(n)+1} \bmod n$$
$$\equiv 0 \bmod n$$
$$\equiv x \bmod n$$

3. Fall: $\text{ggT}(x, n) = p \wedge \text{ggT}(x, q) = 1$
Mit $\text{ggT}(x, q) = 1$ gilt dann nach dem Satz von EULER-FERMAT (bzw. des Kleinen FERMAT):

$$\begin{aligned} x^{\varphi(q)} &\equiv 1 \bmod q && \mid \varphi(q) = q - 1 \\ x^{q-1} &\equiv 1 \bmod q && \mid {}^{(p-1)k} \\ x^{(q-1)(p-1)k} &\equiv 1^{(p-1)k} \bmod q && \mid (q-1)(p-1) = \varphi(n) \\ x^{k\varphi(n)} &\equiv 1 \bmod q && \mid \cdot x \\ x^{k\varphi(n)+1} &\equiv x \bmod q && (1) \end{aligned}$$

Aus $\text{ggT}(x, n) = p$ folgt:

$$\begin{aligned} x &\equiv 0 \bmod p && \mid {}^{k\varphi(n)+1} \\ x^{k\varphi(n)+1} &\equiv 0 \bmod p && \mid x \equiv 0 \bmod p \\ &\equiv x \bmod p && (2) \end{aligned}$$

Aus (1) und (2) folgt:

$$\begin{aligned} x^{k\varphi(n)+1} &\equiv x \bmod p \cdot q \\ &\equiv x \bmod n \end{aligned}$$

4. Fall: $\text{ggT}(x, n) = q \wedge \text{ggT}(x, p) = 1$
Analog zu Fall 3.

Damit ist die Korrektheit bewiesen:

$$m^d \equiv x^{k\varphi(n)+1} \bmod n \equiv x \bmod n \qquad \square$$

Bemerkung:
Nebenbei wurde ein Spezialfall des kleinen FERMATschen Satzes bewiesen:

Kleiner FERMATscher Satz, Spezialfall:
Sei $n = p \cdot q$, ein Produkt von zwei verschiedenen Primzahlen. Dann gilt für alle $a < n \in \mathbb{Z}$ und alle $k \in \mathbb{Z}$:

$$a^{k\varphi(n)+1} \equiv a \bmod n,$$

wobei φ die EULERsche Funktion ist $(\varphi(n) = (p-1)(q-1))$.

3.2.4 Ein Beispiel

LUCIUS ANNAEUS SENECA (4 v. Chr. – 65 n. Chr.):
Longum iter est per praecepta, breve et efficax per exempla.
Lang ist der Weg durch Lehren, kurz und wirksam durch Beispiele.

Man betrachte folgendes Beispiel:
Seien Anton und Berta zwei Personen, die untereinander Nachrichten austauschen wollen. (Trotz der Tatsache, dass 75 % der Kommunikation zwischen Frauen stattfindet, sei aus Gründen der Gleichberechtigung hier ein gemischter Dialog gewählt.) Empfänger Berta erwartet vom Absender Anton eine geheime Nachricht. Sie wählt die Primzahlen $p = 29$ und $q = 37$ (für praktische Nutzung zu klein) und berechnet:

$$n = p \cdot q = 29 \cdot 37 = 1073,$$
$$\varphi(n) = (p-1)(q-1) \rightarrow \varphi(1073) = 28 \cdot 36 = 1008.$$

Nun wählt Berta ein e, das teilerfremd zu $\varphi(n)$ ist. Sie wählt

$$e = 5,$$

denn es gilt:

$$\text{ggT}(5, 1008) = 1.$$

Berta übermittelt an Anton nur

$$n = 1073 \text{ und } e = 5.$$

Anton will Berta die geheime Nachricht

$$x = 8$$

zukommen lassen. Er verschlüsselt sie durch

$$m \equiv x^e \bmod n$$
$$x^e = 8^5 \equiv 578 \bmod 1073 \text{ zu}$$
$$m = 578$$

Anton sendet an Berta nur die Nachricht

$$m = 578.$$

Um die Nachricht zu entschlüsseln muss Berta die Kongruenz:

$$d \cdot e \equiv 1 \bmod \varphi(n)$$
$$d \cdot 5 \equiv 1 \bmod 1008$$

lösen. Es existiert also ein $k \in \mathbb{Z}$ mit:

$$d \cdot 5 \equiv 1 \bmod 1008$$
$$5 \cdot d - 1 = k \cdot 1008$$
$$5 \cdot d - k \cdot 1008 = 1$$

Die letzte Gleichung ist eine lineare DIOPHANtische Gleichung mit 2 Unbekannten. Sie ist in $\mathbb{Z}$ lösbar, wenn ggT(5, 1008) | 1:

$$\text{ggT}(5, 1008) \overset{?}{|} 1$$
$$1 \,|\, 1$$

Die DIOPHANtische Gleichung ist also lösbar. Zur Berechnung des ggT(5, 1008) wird der EUKLIDischer Algorithmus verwendet. Um eine Lösung der DIOPHANtischen Gleichung zu erhalten, verwendet man den erweiterten EUKLIDischen Algorithmus:

$$1008 = 201 \cdot 5 + 3$$
$$5 = 1 \cdot 3 + 2$$
$$3 = 1 \cdot 2 + 1$$
$$2 = 2 \cdot 1 + 0$$

Aus der letzten Gleichung folgt: ggT(5, 1008) = 1. Nun wird ab der vorletzten Gleichung rückwärts gerechnet (erweiterten EUKLIDischer Algorithmus):

$$\begin{aligned} 1 &= 3 - 1 \cdot 2 \\ &= 3 - 1 \cdot (5 - 1 \cdot 3) = -1 \cdot 5 + 2 \cdot 3 \\ &= -1 \cdot 5 + 2 \cdot (1008 - 201 \cdot 5) = (-403) \cdot 5 - (-2) \cdot 1008 \end{aligned}$$

Somit erhält man als Lösung:

$$\Rightarrow \begin{aligned} d_0 &= -403 \\ k_0 &= -2 \end{aligned}$$

Die allgemeine Form der Lösung bestimmt man zu:

$$\overset{\text{allgemeine Lösung:}}{\Rightarrow} \begin{aligned} d_t &= -403 + 1008 \cdot t \\ k_t &= -2 + 5 \cdot t \end{aligned}$$

Da $1 < d < (p-1)(q-1)$ gelten soll, wählt man $t = 1$:

$$\overset{\text{mit } t=1 \text{ (damit } d>1\text{):}}{\Rightarrow} \begin{aligned} d_1 &= -403 + 1008 \cdot 1 = 605 \\ k_1 &= -2 + 5 \cdot 1 = 3 \end{aligned}$$

Berta erhält $d = 605$ und kann somit die Nachricht $m = 578$ entschlüsseln:

$$\begin{aligned} m^d = 578^{605} &\equiv 8 \bmod 1073 \\ \Rightarrow x &= 8 \end{aligned}$$

Man kann auch solch hohe Potenzen modulo n relativ effizient mittels *Wiederholtes Quadrieren und Multiplizieren* lösen, wobei sich jeder Faktor durch Quadrieren (evtl. mehrmals) des vorherigen Faktors modulo n berechnet, beispielsweise:

$$\begin{aligned} x &\equiv m^d \bmod n \\ &\equiv 18^{23} \bmod 55 \\ &\equiv 18^{(1+2+4+16)} \bmod 55 \\ &\equiv 18^1 \cdot 18^2 \cdot 18^4 \cdot 18^{16} \bmod 55 \\ &\equiv 18 \cdot 49 \cdot 36 \cdot 26 \bmod 55 \\ &\equiv 2 \bmod 55 \end{aligned}$$

3.2.5 Abschließende Bemerkungen

Die RSA-Verschlüsselungsfunktion ist ein Kandidat für Einwegfunktion mit Falltür. Die Falltürinformation (Zusatzinformation) ist (d, p, q), also die Kenntnis der Primfaktorzerlegung von n. Ohne diese ist es praktisch unmöglich, die Kongruenz

$$e \cdot d \equiv 1 \bmod (p-1)(q-1)$$

zu lösen. Für Anwender ist es dagegen ein rechentechnisch vergleichsweise geringer Aufwand, eine zu $\varphi(pq) = (p-1)(q-1)$ teilerfremde Zahl e zu finden.
Wenn es jemand schafft, n in p und q zu faktorisieren, so kann er natürlich d berechnen. Deshalb ist die Sicherheit von RSA mindestens an die Annahme geknüpft, Faktorisierung sei schwierig. Ein leichter Faktorisierungsalgorithmus würde RSA „brechen“. Also ist es klar, dass wenn das Faktorisierungsproblem „gelöst“ ist, dass dann das RSA-Verfahren nicht mehr sicher ist. Allerdings ist bis heute nicht bewiesen, dass das Problem, RSA zu brechen, äquivalent zum Faktorisierungsproblem ist.

Ein anderer Weg, RSA zu brechen, besteht darin, einen Algorithmus zur Berechnung der e-ten modularen Wurzel zu finden. Da $m = x^e \bmod n$ ist, ist die e-te modulare Wurzel aus m bezüglich n gleich der Nachricht x. Dieser Angriff gestatt es dem Angreifer, Nachrichten zu entschlüsseln und Unterschriften zu fälschen, ohne den privaten Schlüssel kennen zu müssen. Es ist bisher nicht bekannt, ob dieser Ansatz äquivalent zur Faktorisierung ist oder nicht. Es sind bis heute aber keine generellen Algorithmen bekannt, die RSA in dieser Weise gefährlich werden könnten.
Bei heutiger (1999) erreichten Schnelligkeit von Computern benötigt der Anwender des RSA-Codes zwei mindestens 100-stellige Primzahlen p und q ($n = pq > 10^{200}$), um für Unberechtigte einen Entschlüsselungsaufwand von etwa 74 Jahren zu verursachen.
Es wird als sicher angesehen, wenn genügend lange Schlüssel benutzt werden. Heute (1998) bedeutet dies, dass eine Schlüssellänge von 512 Bit als unsicher, 1024 Bit als hinreichend sicher für die meisten Anwendungen und 2048 Bit als sehr sicher angesehen wird

Weiterhin ist zu erwähnen, dass die Differenz zwischen p und q nicht zu klein sein sollte, da sonst n durch das Verfahren der Differenz der Quadrate faktorisiert werden kann. Der ggT($p - 1$, $q - 1$) sollte klein sein und beide sollten mindestens einen großen Primteiler haben.
Nachteile des RSA-Verfahrens sind eine relativ große Schüssellänge und die Geschwindigkeit. Es ist im Vergleich zu anderen Verfahren etwas langsam. Als Verbesserung könnte (besser: sollte) anstelle der EULERschen Funktion die CARMICHAELsche Funktion verwendet werden. Für diesen Fall also $\lambda(n) = \text{kgV}(p - 1, q - 1)$.

3.3 RABIN-Funktion

Die RABIN-Funktion wurde 1979 von MICHAEL O. RABIN entwickelt. Wie das RSA-Verfahren beruht das RABIN-System darauf, dass es zwar effizient Algorithmen für das Testen der Primzahleigenschaften gibt, effiziente Faktorisierungsalgorithmen aber nicht bekannt sind. Im Gegensatz zum RSA-Verfahren, von dem nicht bekannt ist, dass es mindestens so schwer zu brechen ist wie das Faktorisierungsproblem, ist genau dies beim RABIN-System der Fall. Die Sicherheit des RABIN-Systems ist also äquivalent zur Schwierigkeit des Faktorisierungsproblems.

3.3.1 Der RABIN-Algorithmus

Es seien p, q zwei verschiedene Primzahlen und $n = p \cdot q$. Dann gilt für alle $x \in \{0, ..., n - 1\}$

$$\text{Rabin}_n(x) \stackrel{\text{def}}{=} x^2 \bmod n$$

Verschlüsselungsoperation:

$$E\colon \{0, 1, ..., n - 1\} \to \{0, 1, ..., n - 1\}\ E(x) := x^2 \text{ modulo } n = a$$

Die Quadratwurzeln modulo n können mithilfe der Primzahlzerlegung von n und des Chinesischen Restsatzes berechnet werden. Das heißt, der öffentliche Schlüssel ist n, der geheime Schlüssel (Falltürinformation) ist (p, q).
Allerdings kann die Gleichung $a \equiv x^2 \bmod n$ vier Lösungen x_i (für $1 \le i \le 4$) haben, was ein Entschlüsseln etwas schwierig macht.

Definition (Quadratischer (Nicht-)Rest):

Sei n eine natürliche Zahl. Ein Element $a \in \mathbb{Z}_n^*$ (also ggT(a, n) = 1) heißt quadratischer Rest modulo n (kurz: $a \in \mathsf{QR}_n$), falls ein $x \in \mathbb{Z}_n^*$ existiert mit

$$a \equiv x^2 \bmod n .$$

Andernfalls heißt a quadratischer Nichtrest (kurz: $a \in \mathsf{QNR}_n$).

Satz:

Sei $n = p \cdot q$ für Primzahlen p, q mit $p \equiv q \equiv 3 \bmod 4$. Dann besitzt die quadratische Kongruenz $a \equiv x^2 \bmod n$ für jedes $a \in \mathsf{QR}_n$ genau vier Lösungen, wovon nur eine ein quadratischer Rest ist.

Als weitere zahlentheoretische Funktion mit für die Kryptographie wichtigen Eigenschaften erhalten wir also die Quadratfunktion x^2: $\mathsf{QR}_n \to \mathsf{QR}_n$ die nach vorigem Satz bijektiv ist (falls $n = p \cdot q$ für Primzahlen p, q mit $p \equiv q \equiv 3 \bmod 4$). Ihre Umkehrfunktion $x \to \sqrt{x}$ heißt *diskrete Wurzelfunktion*, und kann (nur) bei Kenntnis der Primfaktoren p und q von n effizient berechnet werden. Bis heute ist kein effizientes Verfahren bekannt, mit dem man ohne Kenntnis der Faktorisierung von n für ein $a \in \mathbb{Z}_n^*$ entscheiden kann, ob $a \in \mathsf{QR}_n$ ist oder nicht.

Man kann zeigen, dass für Zahlen $n = p \cdot q$ für Primzahlen p, q mit $p \equiv q \equiv 3 \bmod 4$ das Faktorisierungsproblem und das Problem, eine Lösung der quadratischen Kongruenz $a \equiv x^2 \bmod n$ für ein gegebenes $a \in \mathsf{QR}_n$ zu finden, äquivalent sind.

3.3.2 RABIN-Funktion als Einwegfunktion

Es soll gezeigt werden, dass die Rabin-Funktion ein Kandidat für Familie von Einwegfunktionen. Dazu sind die drei Algorithmen anzugeben, die die Definition erfüllen

$$\mathrm{SQR} \overset{\mathrm{def}}{=} (I_{\mathrm{BI}}, D_{\mathrm{QR}}, F_{\mathrm{SQR}}) .$$

Bei Eingabe 1^n wählt Algorithmus I_{BI} zwei Primzahlen p und q mit $2^{n-1} \le p < q < 2^n$ und $p \equiv 3 \bmod 4 \wedge q \equiv 3 \bmod 4$ aus und gibt $n = p \cdot q$ aus. Bei Eingabe n wählt Algorithmus D_{QR} ein Element aus QR_n aus. Die Ausgabe von Algorithmus F_{SQR} ist

$$\mathrm{Rabin}_n(x) \overset{\mathrm{def}}{=} x^2 \bmod n .$$

I_{BI}: Eingabe: 1^n
Ausgabe: n mit $n = p \cdot q$; p, q prim; $p \equiv 3 \bmod 4 \wedge q \equiv 3 \bmod 4$; $2^{n-1} \le p < q < 2^n$

D_{QR}: Eingabe: n
Ausgabe: x mit $x \in \mathsf{QR}_n$

F_{SQR}: Eingabe: (n, x)
Ausgabe: a mit $a \equiv x^2 \bmod n$

3.3.3 Ein Beispiel

Es sei:

$$p = 3$$
$$q = 7$$
$$n = p \cdot q = 3 \cdot 7 = 21.$$

Dann folgt:

$$\mathbb{Z}_n = \mathbb{Z}_{21} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20\}$$
$$\mathbb{Z}_n^* = \mathbb{Z}_{21}^* = \{1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20\}$$

Die RABIN-Funktion ist:

$$\text{RABIN}_n(x) = \text{RABIN}_{21}(x) = x^2 \bmod 21$$

x	x^2	$x^2 \bmod n$	x	x^2	$x^2 \bmod n$
0	0	0			
1	1	1	11	121	16
2	4	4	12	144	18
3	9	9	13	169	1
4	16	16	14	196	7
5	25	4	15	225	15
6	36	15	16	256	4
7	49	7	17	289	16
8	64	1	18	324	9
9	81	18	19	361	4
10	100	16	20	400	1

Da gilt:

$$p \equiv 3 \bmod 4$$
$$q \equiv 3 \bmod 4$$

lässt sich also eine bijektive Abbildung: $\mathsf{QR}_{21} \rightarrow \mathsf{QR}_{21}$ konstruieren:

$$a = x^2 \bmod 21$$

a	Anzahl	x
0	1	0
1	4	1, 8, 13, 20
4	4	2, 5, 16, 19
7	2	7, 14
9	2	3, 18
15	2	6, 15
16	4	4, 10, 11, 17
18	2	9, 12

Somit sind die quadratischen Reste: 1, 4, 16 und die Abbildung $a \rightarrow a^2 \bmod n$:

a	a^2	$a^2 \bmod n$
1	1	1
4	16	16
16	256	4

3.4 Diskreter Logarithmus

3.4.1 Der DLP-Algorithmus

Ein ähnlich schwieriges Problem, wie das Faktorisieren einer Zahl, ist Problem des diskreten Logarithmus DLP (*discrete-logarithm problem*). Um eine diskrete Einwegfunktion zu schaffen, wählt man eine Primzahl p und einen Erzeuger g der Gruppe $\mathbb{Z}_p^* = \{1, ..., p-1\}$.

Definition (Erzeuger/Primitivwurzel)

Eine Zahl g heißt Erzeuger von $\mathbb{Z}_n^*$ (oder Primitivwurzel modulo n), falls

$$\mathbb{Z}_n^* = \{g^0, g^1, g^2, \cdots, g^{\varphi(n)-1}\}$$

ist.

So ein Erzeuger kann für $\mathbb{Z}_p^*$ mit p prim recht bequem berechnet werden, wenn man die Primzahlzerlegung von $p-1$ kennt.

Satz (GAUß):

Ist p prim, so gibt es genau $\varphi(p) - 1$ Erzeuger von $\mathbb{Z}_p^*$.

Satz:

Sei p prim. Dann ist $g \in \mathbb{Z}_p^*$ genau dann ein Erzeuger, wenn für jeden Primteiler q von $p-1$ gilt:

$$g^{\frac{p-1}{q}} \not\equiv 1 \bmod p.$$

Folgender probabilistische Algorithmus berechnet einen Erzeuger $g \in \mathbb{Z}_p^*$, falls alle Primteiler q von $p-1$ bekannt sind.

Algorithmus COMPUTEGENERATOR($p, q_1, ..., g_k$):

1 **Eingabe:** Primzahl p, Primteiler $q_1, ..., g_k$ von $p-1$
2 **repeat**
3 **rate zufällig** $g \in \{2, \ldots, p-1\}$
4 **until** $g^{\frac{p-1}{q_i}} \not\equiv 1 \bmod p$ für $i = 1, \ldots, k$
5 **Ausgabe:** g

Die erwartete Anzahl der Schleifendurchläufe ist $O(\ln \ln p)$.

Somit sieht die diskrete Exponentialfunktion f zur Basis g so aus:

$$f: \{1, ..., p-1\} \to \{1, ..., p-1\} \text{ mit } x \to g^x \bmod p$$

bzw.

$$\mathrm{DLP}_{p,g}(x) \stackrel{\mathrm{def}}{=} g^x \bmod p\,.$$

Um die Funktion $f(x) = g^x \bmod p$ zu berechnen gibt es effiziente Algorithmen. Die Umkehrfunktion nennt man diskreten Logarithmus $\log_{p,g}(f(x))$. Zur Berechnung des diskreten Logarithmus sind keine effizienten Verfahren bekannt.

3.4.2 DLP-Funktion als Einwegfunktion

Die DLP Familie von Funktionen ist durch das Tripel von Algorithmen definiert: (I_{DLP}, D_{DLP}, F_{DLP}). Der Algorithmus I_{DLP} wählt bei Eingabe 1^n eine Primzahl p aus mit $2^{n-1} \le p < 2^n$, einen Erzeuger g von $\mathbb{Z}_p^* = \{1, ..., p-1\}$ und gibt (p, g) aus. Algorithmus D_{DLP} wählt bei Eingabe (p, g) ein Element x aus der Menge $\{1, ..., p-1\}$ aus und Algorithmus F_{DLP} bekommt als Eingabe $((p, g), x)$ und gibt folgendes aus:

$$\mathrm{DLP}_{p,g}(x) \stackrel{\text{def}}{=} g^x \bmod p .$$

I_{DLP}: Eingabe: 1^n
Ausgabe: (p, g) mit p prim; $2^{n-1} \le p < 2^n$; g ist Erzeuger von $\mathbb{Z}_p^* = \{1, ..., p-1\}$

D_{DLP}: Eingabe: (p, g)
Ausgabe: x mit $x \in \{1, ..., p-1\}$

F_{DLP}: Eingabe: $((p, g), x)$
Ausgabe: $f(x) \equiv g^x \bmod p$.

Also muss man den diskreten Logarithmus (Basis g) modulo p berechnen, um DLP zu invertieren.

3.4.3 Ein Beispiel

Beispielsweise sieht der Graph der diskreten Exponentialfunktion $f(x) \equiv 3^x \bmod 101$ so aus:

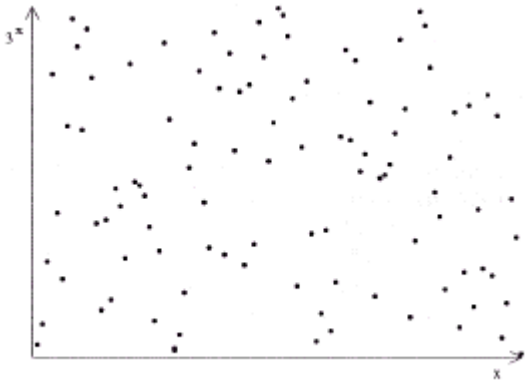

Bild 8.1 Graph der diskreten Exponentialfunktion modulo p = 101 zur Basis 3

Sei nun

$$p = 11.$$

Gewählt. Dann ist

$$p - 1 = 10$$

mit den Primteilern

$$q_1 = 2$$
$$q_2 = 5.$$

Dann ist

$$\mathbb{Z}_p = \mathbb{Z}_{11} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$
$$\mathbb{Z}_p^* = \mathbb{Z}_{11}^* = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$

Gesucht ist also ein Erzeuger g von $\mathbb{Z}_{11}^* = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$. Für g muss mit jedem Primteiler q von $p - 1$ gelten:

$$g^{\frac{p-1}{q}} \not\equiv 1 \bmod p.$$

	$q_1 = 2$		$q_2 = 5$	
x	$x^{\frac{10}{2}}$	$x^{\frac{10}{2}} \bmod 11$	$x^{\frac{10}{5}}$	$x^{\frac{10}{5}} \bmod 11$
1	1	1	1	1
2	32	10	4	4
3	243	1	9	9
4	1024	1	16	5
5	3125	1	25	3
6	7776	10	36	3
7	16807	10	49	5
8	32768	10	64	9
9	59049	1	81	4
10	100000	10	100	1

Die Menge aller Erzeuger von $\mathbb{Z}_{11}^*$ ist folglich

$$G = \{2, 6, 7, 8\}$$

Man wähle $g = 7$.

x	7^x	$7^x \bmod p$
0	1	1
1	7	7
2	49	5
3	343	2
4	2401	3
5	16807	10
6	117649	4
7	823543	6
8	5764801	9
9	40353607	8
10	282475249	1

Für $x \in \{1, ..., p - 1\} = \{1, ..., 10\}$ gibt diese Tabelle die Funktion

$$\mathrm{DLP}_{11,7}(x) = 7^x \bmod 11$$

an.

3.5 Das SUMMEN-Problem

Das Summen-Problem ist das folgende: Gegeben seinen natürliche Zahlen a_1, ..., a_n und s. Das Problem SUMME besteht darin zu entscheiden, ob es eine Menge $I \subseteq \{1, ..., n\}$ gibt, so dass

$$\sum_{i \in I} a_i = s .$$

Daraus lässt sich ein Kandidat für eine Einwegfunktion herleiten. Es sei

$$f_{\text{SUMME}}(a_1, \cdots, a_n, I) = \left(a_1, \cdots, a_n, \sum_{i \in I} a_i \right)$$

mit $|a_1| = ... = |a_n| = n$ und $I \subseteq \{1, ..., n\}$. Offensichtlich ist f_{SUMME} in Polynomialzeit berechenbar. Das SUMMEN-Problem ist NP-vollständig. Allerdings reicht diese Tatsache nicht aus, um die Einweg-Eigenschaft von f_{SUMME} zu beweisen. Für manche Fälle ist das SUMMEN-Problem sogar einfach zu lösen. Die Vermutung, dass f_{SUMME} eine Einwegfunktion ist, basiert darauf, dass bekannte Algorithmen fehlschlagen, wenn sie zufällige Instanzen mit „hoher Schreibdichte“ (“high-density”) handhaben müssen, d. h. Instanzen, in denen die Länge der Elemente ungefähr ihrer Anzahl gleicht, wie in der Definition von f_{SUMME}.

4 Elementare Zahlentheorie

CARL FRIEDRICH GAUß (1777 – 1855):

Die Mathematik ist die Königin der Wissenschaften, die Zahlentheorie aber ist die Königin der Mathematik.

Die Elementare Zahlentheorie befasst sich mit den Teilbarkeitseigenschaften der ganzen Zahlen. Der 1. Hauptsatz der Zahlentheorie (zugleich Hauptsatz der elementaren Zahlentheorie) wurde das erste Mal präzise von CARL FRIEDRICH GAUß in seinen Disquisitiones Arithmeticae (1801) formuliert.

1. Hauptsatz der Zahlentheorie (CARL FRIEDRICH GAUß 1801):

Jede natürliche Zahl a größer als 1 lässt sich als Produkt von Primzahlen schreiben. Sind zwei solche Zerlegungen $a = p_1 \cdot p_2 \cdot ... \cdot p_n = q_1 \cdot q_2 \cdot \ldots \cdot q_m$ gegeben, dann gilt nach eventuellem Umsortieren $n = m$ und für alle i: $p_i = q_i$.

4.1 Kongruenzen

Definition:

Sei n eine natürliche Zahl mit $n > 1$. Lassen zwei Zahlen a und b bei Division durch n den gleichen Rest, so nennt man a und b kongruent modulo n und schreibt dafür

$$a \equiv b \bmod n \text{ oder } a \equiv b(n)$$

Satz:

$$a \equiv b \bmod n \quad \Leftrightarrow \quad \exists k \in \mathbb{Z} : a = b + k \cdot n \quad \Leftrightarrow \quad n \mid (a - b)$$

Satz:

Es gelte $a \equiv b \bmod n \wedge c \equiv d \bmod n$, dann folgt:

$$a \equiv b \bmod n \quad \Leftrightarrow \quad b \equiv a \bmod n$$
$$(a + c) \equiv (b + d) \bmod n$$
$$(a - c) \equiv (b - d) \bmod n$$
$$a \cdot c \equiv b \cdot d \bmod n.$$

Insbesondere folgt mit $c = a$ und $d = b$:

$$a^2 \equiv b^2 \bmod n \text{ oder allgemein:}$$
$$a^d \equiv b^d \bmod n$$

Satz:

Es gelte $ac \equiv bd \bmod n \wedge c \equiv d \bmod n$ und ggT$(c, n) = 1$, dann folgt:

$$a \equiv b \bmod n.$$

Beweis:

Es gilt:

$$ac \equiv bd \bmod n$$
$$ac - bd \equiv 0 \bmod n$$
$$(a - b)c + b(c - d) \equiv 0 \bmod n$$

Daraus folgt:

$$n \mid ((a-b)c + b(c-d)).$$

Aus $c \equiv d \bmod n$ folgt:

$$c - d \equiv 0 \bmod n\,.$$

Das heißt:

$$n \mid (c-d) \text{ und insbesondere}$$
$$n \mid b(c-d)$$

Also muss auch gelten:

$$n \mid (a-b)c$$

Da aber ggT(c, n) = 1, also $n \nmid c$, folgt:

$$n \mid (a-b)$$

und somit gilt:

$$a - b \equiv 0 \bmod n$$
$$a \equiv b \bmod n$$

Damit ist der Satz beweisen. □

Satz:

Es gelte $a \equiv b \bmod p \wedge a \equiv b \bmod q$ und p, q seinen verschiedene Primzahlen, dann folgt:

$$a \equiv b \bmod (p \cdot q)$$

Beweis:

Vor.: $a \equiv b \bmod p \wedge a \equiv b \bmod q$ und p, q seinen Primzahlen
Daraus folgt:

$$\exists k_1, k_2 \in \mathbb{Z} : (a-b) = k_1 p \wedge (a-b) = k_2 q$$
$$\Leftrightarrow k_1 p = k_2 q$$

Da p, q verschiedene Primzahlen folgt:

$$p \nmid q \Rightarrow p \mid k_2$$
$$\Leftrightarrow \exists k_3 \in \mathbb{Z} : p \cdot k_3 = k_2$$

Daraus folgt:

$$k_1 p = k_2 q$$
$$= k_3 \cdot p \cdot q$$

Und mit $(a-b) = k_1 p$ folgt:

$$k_3 \cdot p \cdot q = k_1 p$$
$$= (a-b)$$
$$\Leftrightarrow a - b \equiv 0 \bmod (p \cdot q)$$
$$a \equiv b \bmod (p \cdot q)$$

Damit ist der Satz beweisen. □

Definition (EULERsche φ-Funktion):

Die EULERsche φ-Funktion ordnet jeder natürlichen Zahl $n > 0$ die Anzahl $\varphi(n)$ der zu n teilerfremden Zahlen x zu mit $1 \le x \le n$.

(Anders ausgedrückt: $\varphi(n)$ ist die Anzahl der Zahlen x mit $1 \le x \le n$ und ggT(x, n) = 1.)

Satz:

Ist n eine beliebige natürliche Zahl, dann kann man $\varphi(n)$ wie folgt ausrechnen:

$$\varphi(n) = n\prod_{p|n}\left(1-\frac{1}{p}\right)$$

wobei das Produkt über alle Primteiler p von n zu erstrecken ist.

Bemerkung:

Für Primzahlen p gilt: $\varphi(p) = p - 1$.

Für Primzahlenpotenzen p^α gilt: $\varphi(p^\alpha) = p^\alpha - p^{\alpha-1}$.

Satz:

Seien m und n natürliche Zahlen mit ggT(m, n) = 1, dann gilt:

$$\varphi(mn) = \varphi(m) \cdot \varphi(n)$$

Satz von EULER-FERMAT:

Sind a und n teilerfremde natürliche Zahlen (also ggT(a, n) = 1), dann gilt:

$$a^{\varphi(n)} \equiv 1 \bmod n\,.$$

Kleiner FERMATscher Satz:

Sei p eine Primzahl, die $a \in$ ù nicht teilt, also pða bzw. ggT(a, p) = 1, dann gilt:

$$a^{p-1} \equiv 1 \bmod p.$$

Beweis:

Man betrachte die Vielfachen von a, also a, $2a$, $3a$, ..., $(p-1)a$. Keine zwei dieser Zahlen können kongruent modulo p sein, denn sonst wäre ihre Differenz kongruent zu 0 modulo p, also $(i-j)a \equiv 0 \bmod p$ für $1 \le i \le j \le (p-1)$ und das ist nicht möglich, da $(i-j) \le (p-1)$ und deswegen pð$(i-j)$ und pða per Voraussetzung.

Also sind die Zahlen a, $2a$, $3a$, ...,$(p-1)a$ kongruent zu den Zahlen 1, 2, ..., $(p-1)$ in irgendeiner Reihenfolge, da die modulo-Relation diese Äquivalenzklassen bildet.

Also gilt:

$$a \cdot 2a \cdot 3a \cdot \ldots \cdot (p-1)a = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot (p-1)a^{p-1} \equiv 1 \cdot 2 \cdot 3 \cdot \ldots \cdot (p-1) \bmod p$$

nach den Sätzen über Kongruenzen. Es gilt auch für die Differenz:

$$1 \cdot 2 \cdot 3 \cdot \ldots \cdot (p-1)(a^{p-1} - 1) \equiv 0 \bmod p,$$

also

$$k(a^{p-1} - 1) \equiv 0 \bmod p,$$

da aber k nicht durch p teilbar ist, gilt dies für $a^{p-1} - 1$ und somit $a^{p-1} - 1 \equiv 0 \bmod p$ und somit $a^{p-1} \equiv 1 \bmod p$. □

Kleiner FERMATscher Satz, Spezialfall:

Sei $n = p \cdot q$, ein Produkt von zwei verschiedenen Primzahlen. Dann gilt für alle $a < n \in \mathbb{Z}$ und alle $k \in \mathbb{Z}$:

$$a^{k\varphi(n)+1} \equiv a \bmod n$$

wobei φ die EULERsche Funktion ist $(\varphi(n) = (p-1)(q-1))$.

Beweis:

Siehe Beweis zur Korrektheit des RSA-Verfahren.

Die Umkehrung des Kleinen FERMAT ist jedoch falsch. Gegenbeispiele sind die so genannten CARMICHAEL-Zahlen. Die kleinste ist 561 = 3 · 11 · 17, eine andere 32150317751 = 151 · 751 · 28351

Definition (CARMICHAEL-Zahl):

Eine zerlegbare Zahl n heißt CARMICHAEL-Zahl, wenn in der Primfaktorzerlegung von $n = p_1 \cdot p_2 \cdot \cdots \cdot p_k$ jeder Primfaktor nur einmal vorkommt und für alle Primfaktoren p von n die Zahl $n - 1$ durch $p - 1$ teilbar ist.

Definition (CARMICHAEL-Funktion: $\lambda(n)$):

Ist die Primfaktorzerlegung einer natürlichen Zahl n gegeben durch $n = p_1^{\alpha_1} \cdot p_2^{\alpha_2} \cdot \cdots \cdot p_k^{\alpha_k}$, so gilt:

$$\lambda(n) = \mathrm{kgV}\left(\lambda(p_1^{\alpha_1}), \cdots, \lambda(p_k^{\alpha_k})\right)$$

wobei

$$\lambda(p_i^{\alpha_i}) = \begin{cases} 2^{\alpha_i - 2} & \text{, wenn } p_i = 2 \text{ und } \alpha_i > 2 \\ p_i^{\alpha_i - 1}(p_i - 1) & \text{, sonst} \end{cases}$$

$\lambda(n)$ heißt CARMICHAEL-Funktion.

Satz von CARMICHAEL:

Für zwei natürliche teilerfremde Zahlen a und n gilt:

$$a^{\lambda(n)} \equiv 1 \bmod n$$

Für ein festes a und variables m ist $\lambda(n)$ der kleinste Exponent mit dieser Eigenschaft.

Definition ($\mathbb{Z}_n$):

$\mathbb{Z}_n$ umfasst alle ganzen Zahlen von 0 bis $n - 1$: $\mathbb{Z}_n = \{0, 1, 2, ..., n-2, n-1\}$.
$\mathbb{Z}_n$ wird auch Menge der Restklassen modulo n genannt.

$\mathbb{Z}_n$ ist eine häufig verwendete endliche Gruppe aus den natürlichen Zahlen. Sie wird manchmal auch als Restemenge R modulo n bezeichnet.

Definition ($\mathbb{Z}_n^*$):

$\mathbb{Z}_n^* := \{a \in \mathbb{Z}_n \mid \mathrm{ggT}(a, n) = 1\}$

$\mathbb{Z}_n^*$ wird manchmal auch als reduzierte Restemenge *R'* modulo *n* bezeichnet.

Beispiel:

Für $n = 10 = 2 \cdot 5$ gilt:
vollständige Restmenge $R = \mathbb{Z}_n = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.
reduzierte Restemenge $R' = \mathbb{Z}_n^* = \{1, 3, 7, 9\} \rightarrow \varphi(n) = 4$ (siehe EULERfunktion).

Bemerkung:

R' bzw. $\mathbb{Z}_n^*$ ist immer eine echte Teilmenge von *R* bzw. $\mathbb{Z}_n$, da 0 immer Element von $\mathbb{Z}_n$, aber nie Element von $\mathbb{Z}_n^*$ ist.

Da 1 (per Definition) und $n - 1$ immer teilerfremd zu *n* sind, sind sie stets Elemente beider Mengen. Wählt man irgendein Element aus $\mathbb{Z}_n^*$ und multipliziert es mit jedem anderen Element von $\mathbb{Z}_n^*$, so sind die Produkte alle wieder in $\mathbb{Z}_n^*$ und außerdem sind die Ergebnisse eine eindeutige Permutation der Elemente von $\mathbb{Z}_n^*$. Da die 1 immer Element von $\mathbb{Z}_n^*$ ist, gibt es in dieser Menge eindeutig einen „Partner", so dass das Produkt 1 ergibt.

4.2 Grundlagen für das RABINverfahren

Definition (Quadratischer (Nicht-)Rest):

Sei *n* eine natürliche Zahl. Ein Element $a \in \mathbb{Z}_n^*$ (also ggT(a, n) = 1) heißt quadratischer Rest modulo *n* (kurz: $a \in \mathsf{QR}_n$), falls ein $x \in \mathbb{Z}_n^*$ existiert mit

$$a \equiv x^2 \bmod n .$$

Andernfalls heißt *a* quadratischer Nichtrest (kurz: $a \in \mathsf{QNR}_n$).

Satz:

Sei *n* eine natürliche Zahl, $a, x \in \mathbb{Z}_n$, und es gelte $a \equiv x^2 \bmod n$.
Dann gilt:

$$a \in \mathbb{Z}_n^* \quad \Leftrightarrow \quad x \in \mathbb{Z}_n^* .$$

Beweis:

Aus $a \equiv x^2 \bmod n$ folgt:

$$\exists k \in \mathbb{Z} : a - x^2 = k \cdot n$$

$\Rightarrow$: Sei $a \in \mathbb{Z}_n^*$, also ggT(a, n) = 1.

Annahme: $x \notin \mathbb{Z}_n^*$, also ggT(x, n) = $k_g > 1$.

Das heißt:

$$x = k_g \cdot x_0 \quad \wedge \quad n = k_g \cdot n_0 \quad \text{mit } x_0, n_0, k_g \in \mathbb{Z} .$$

Also gilt:

$$\begin{aligned} a - x^2 &= k \cdot n \\ a &= x^2 + k \cdot n \\ &= (k_g \cdot x_0)^2 + k \cdot k_g \cdot n_0 \\ &= k_g (k_g \cdot x_0^2 + k \cdot n_0) \end{aligned}$$

Daraus folgt:

$$k_g \mid a$$

Da aber

$$k_g \mid n \quad \wedge \quad k_g \mid a \quad \wedge \quad k_g > 1$$

folgt:

$$\text{ggT}(a,n) \geq k_g > 1,$$

was ein Widerspruch zur Voraussetzung ist. Folglich ist die Annahme falsch und es gilt:

$$x \in \mathbb{Z}_n^*.$$

$\Leftarrow$: Sei $x \in \mathbb{Z}_n^*$, also ggT(x, n) = 1.

Annahme: $a \notin \mathbb{Z}_n^*$, also ggT(a, n) = $k_g > 1$.

Sei p prim mit

$$\begin{aligned} & p \mid k_g \\ \Leftrightarrow\ & k_g = p \cdot k_{g'} \end{aligned}$$

also ist p ein Primfaktor von k_g (dieser existiert nach dem 1. Hauptsatz der Zahlentheorie).

Das heißt:

$$a = p \cdot a_0 \quad \wedge \quad n = p \cdot n_0 \quad \text{mit } a_0, n_0 \in \mathbb{Z}.$$

Also gilt:

$$\begin{aligned} a - x^2 &= k \cdot n \\ x^2 &= a - k \cdot n \\ &= p \cdot a_0 - k \cdot p \cdot n_0 \\ &= p(a_0 - k \cdot n_0) \end{aligned}$$

Daraus folgt:

$$\begin{aligned} & p \mid x^2 \\ \Rightarrow\ & p \mid x \quad \text{(da } p \text{ prim)} \end{aligned}$$

Da aber

$$p \mid n \quad \wedge \quad p \mid x \quad \wedge \quad p > 1$$

folgt:

$$\text{ggT}(x,n) \geq p > 1,$$

was ein Widerspruch zur Voraussetzung ist. Folglich ist die Annahme falsch und es gilt:

$$a \in \mathbb{Z}_n^*.$$

Damit ist der Satz bewiesen. □

Satz:

Sei $a \in \mathbb{Z}_p^*$, $p > 2$ prim.

Dann gilt:

$$a^{\frac{p-1}{2}} \equiv 1 \bmod p \quad \Leftrightarrow \quad a \in \mathsf{QR}_p.$$

Satz:

Sei $p > 2$ prim, dann besitzt die quadratische Kongruenzgleichung $a \equiv x^2 \bmod p$ für jedes $a \in \mathsf{QN}_p$ genau zwei Lösungen. Im Fall $p \equiv q \equiv 3 \bmod 4$ sind dies $\pm a^k$ (für $k = \frac{p+1}{4}$) von denen genau eine ein quadratischer Rest ist.

4.3 Grundlagen für das Problem des Diskreten Logarithmus

Nehmen wir ein beliebiges Element a aus $\mathbb{Z}_n^*$ und betrachten die Folge $a^0 = 1$, $a^1 = a$, a^2, a^3, ... , so wissen wir nach dem Satz von EULER-FERMAT, dass spätestens für $e = \varphi(n)$ wieder $a^e = 1$ gilt.

Definition (Ordnung):

Es sei $n \geq 1$ und ggT$(a, n) = 1$. Die Ordnung von a modulo n ist

$$\mathrm{ord}_n(a) = \min\{e > 1 \mid a^e \equiv 1 \bmod n\}.$$

Für das folgende besonders interessant sind Elemente a aus $\mathbb{Z}_n^*$, die ganz $\mathbb{Z}_n^*$ aufspannen.

Definition (Erzeuger/Primitivwurzel)

Eine Zahl g heißt Erzeuger von $\mathbb{Z}_n^*$ (oder Primitivwurzel modulo n), falls

$$\mathbb{Z}_n^* = \{g^0, g^1, g^2, \cdots, g^{\varphi(n)-1}\}$$

ist.

Ein Element $a \in \mathbb{Z}_n^*$ ist also genau dann ein Erzeuger, wenn $\mathrm{ord}_n(a) = \varphi(n)$ ist. Falls $\mathbb{Z}_n^*$ einen Erzeuger besitzt, wird $\mathbb{Z}_n^*$ auch zyklisch genannt. Da $\{a^0, a^1, a^2, \cdots, a^{ord_n(a)-1}\}$ eine Untergruppe von $\mathbb{Z}_n^*$ ist, ist $\mathrm{ord}_n(a)$ für alle $a \in \mathbb{Z}_n^*$ ein Teiler von $\varphi(n)$, d. h. $\varphi(n) \equiv 0 \bmod \mathrm{ord}_n(a)$. Allgemeiner gilt

$$a^i \equiv a^j \bmod n \quad \Leftrightarrow \quad i \equiv j \bmod \mathrm{ord}_n(a)$$

Definition (Index/Diskreter Logarithmus):

Sei g ein Erzeuger von $\mathbb{Z}_n^*$ und $a \in \mathbb{Z}_n^*$.

Dann heißt der eindeutig bestimmte Exponent $e \in \{0, 1, ..., \varphi(n) - 1\}$ mit

$$g^e \equiv a \bmod n$$

Index oder diskreter Logarithmus modulo n von a zur Basis g (kurz: $e \in \log_{n,g}(a)$).

Die Exponentialfunktion $e \to g^e \bmod n$ ist eine bijektive Abbildung von der Menge $\{0, 1, ..., \varphi(n) - 1\}$ auf $\mathbb{Z}_n^*$.
Während die diskrete Exponentialfunktion $e \to g^e \bmod n$ durch wiederholtes Quadrieren und Multiplizieren effizient berechnet werden kann, sind bis heute keine effizienten Verfahren zur Berechnung des diskreten Logarithmus bekannt.

Es soll nun betrachtet werden, unter welchen Umständen ein Erzeuger von $\mathbb{Z}_n^*$ existiert und wie er berechnet werden kann.

Satz (Gauß):
Genau für $n \in \{1, 2, 4, p^k, 2p^k \mid 2 < p \text{ prim}\}$ ist $\mathbb{Z}_n^*$ zyklisch.

Satz (GAUß):
Ist p prim, so gibt es genau $\varphi(p) - 1$ Erzeuger von $\mathbb{Z}_p^*$.

Satz:
Sei p prim. Dann ist $g \in \mathbb{Z}_p^*$ genau dann ein Erzeuger, wenn für jeden Primteiler q von $p - 1$ gilt:

$$g^{\frac{p-1}{q}} \dagger 1 \bmod p.$$

Folgender probabilistische Algorithmus berechnet einen Erzeuger $g \in \mathbb{Z}_p^*$, falls alle Primteiler q von $p - 1$ bekannt sind.

Algorithmus COMPUTEGENERATOR($p, q_1, ..., g_k$):
1 **Eingabe:** Primzahl p, Primteiler $q_1, ..., g_k$ von $p - 1$
2 **repeat**
3 **rate zufällig** $g \in \{2, \ldots, p - 1\}$
4 **until** $g^{\frac{p-1}{q_i}} \dagger 1 \bmod p$ für $i = 1, \ldots, k$
5 **Ausgabe:** g

Die erwartete Anzahl der Schleifendurchläufe ist $O(\ln \ln p)$.

5 Personenverzeichnis

- EUKLID VON ALEXANDRIA (ca. 325 v. Chr. – ca. 265 v. Chr.), griechischer Mathematiker
- LUCIUS ANNAEUS SENECA (4 v. Chr. – 65 n. Chr.), philosophischer Schriftsteller und Dichter
- DIOPHANTUS VON ALEXANDRIA (ca. 200 – ca. 284), griechischer Mathematiker
- PIERRE DE FERMAT (1601 – 1665), französischer Mathematiker
- ANTONIO STRADIVARI (1644 – 1737), italienischer Geigenbauer
- LEONARD EULER (1707 – 1783), schweizer Mathematiker
- CARL FRIEDRICH GAUß (1777 – 1855), deutscher Mathematiker
- ROBERT D. CARMICHAEL (1879 – 1967)

6 Literaturverzeichnis

- Oded Goldreich: "Foundations of Cryptography", Cambridge University Press, 2001
- I. N. Bronstein, K. A. Semendjajew, G. Musiol, H. Mühlig: „Taschenbuch der Mathematik", Verlag Harri Deutsch, Frankfurt am Main, 1999
- Aus dem Internet die Skripte, Mitschriften und Publikationen:
  - Ulli Wölfel: „Kryptographische Hashfunktionen und Digitale Signaturen"
  - Alexander Schick: „Einführung in die Kryptologie für Anfänger und Fortgeschrittene"
  - Andreas Lochbihler: „Die EULERsche φ-Funktion"
  - Sven Tantau: „Kryptographie: digitale Signatur"
  - Matthias Seeger: „Probabilischtische Verschlüsselung und beweisbar sichere Public-Key-Kryptosysteme"
  - Kai Fleischer: „Asymmetrische Verschlüsselungsverfahren"
  - Sabine Blechinger: „Kryptographische Verfahren"
  - Wolfgang Beinhauer: „Kryptographie"
  - Thomas Hungenberg: „Asymmetrische Verschlüsselungsverfahren"
  - Anton Gerold: http://home.in.tum.de/~gerold/aktvorl20012002/gliederung.html
  - Ernst Günter Gießmann, Dirk Verworner u. a.: „Kryptologie", http://www.informatik.hu-berlin.de/Institut/dokumente/Scripte/source/Kryptologie.ps.gz
  - Prof. Dr. Johannes Köbler: „Kryptologie 1", http://www.informatik.hu-berlin.de/Institut/struktur/algorithmenII/Lehre/SS2002/Kryptologie/krypt.pdf
  - http://www.cryptool.de; CrypTool Skript Mathematik und Kryptographie
  - http://www.uni-mainz.de/~pommeren/Kryptologie/Asymmetrisch
  - http://fma2.math.uni-magdeburg.de/~bessen/krypto
  - http://www.math-it.de
  - http://www-groups.dcs.st-and.ac.uk/~history/index.html
  - http://www.iti.fh-flensburg.de/lang/algorithmen/code/krypto/index.htm
  - http://www.weltchronik.de
  - http://www.zahlentheorie.de