Algorithmische Zahlentheorie

Seminararbeit im Masterstudiengang Angewandte Informatik
WS 2015/16 Hochschule Hannover [*]

Marcel Reichenbach[†]
Hochschule Hannover
Fakultät IV - Wirtschaft und Informatik
30459 Hannover
Marcel.Reichenbach@stud.HS-Hannover.de

Marius Rohde[‡]
Hochschule Hannover
Fakultät IV - Wirtschaft und Informatik
30459 Hannover
Marius.Rohde@stud.HS-Hannover.de

ZUSAMMENFASSUNG

Diese Arbeit beschäftigt sich mit der theoretischen Grundlage für die asymmetrische Kryptographie. Es werden zunächst die benötigten Grundlagen der Zahlentheorie gegeben, um anschließend einen genaueren Blick auf die Algorithmen zur Primzahlerkennung, des Diffie-Hellmann-Austausches und der Anwendung elliptischer Kurven zu geben. Neben der Ausarbeitung verwendeter Algorithmen wird auch die Laufzeit dieser untersucht. Zufallszahlengeneratoren werden nicht betrachtet.

1. EINLEITUNG

Die algorithmische Zahlentheorie bildet die Grundlage der heutigen asymmetrischen Kryptographie und somit auch für einen Großteil des sicheren Datenverkehrs in Netzwerken. Ob Onlinebanking, digitale Signaturen oder virtuelle private Netzwerke, überall finden asymmetrische Verschlüsselungsalgorithmen Anwendung. Obwohl bereits Euklid ca. 300 v. Chr. zahlentheoretische Algorithmen entwickelt hat, konnte erst mit der asymmetrischen Verschlüsselung eine praktische Anwendung der Zahlentheorie gefunden werden. Zu den bedeutendsten Mathematikern, die sich mit der Zahlentheorie beschäftigten, gehören Euklid, Eratosthenes von Kyrene, Sun-Tse, Pierre de Fermat, Leonhard Euler, Carl Friedrich Gauß und David Hilbert. Trotz der langen Historie sind einige Fragen der Zahlentheorie wie z.B. die Unendlichkeit der Primzahlzwillinge oder die Goldbachsche Vermutung seit Jahrhunderten ungelöst. Erst 2002 konnten die drei indischen Wissenschaftler Manindra Agrawal, Neeraj Kayal und Nitin Saxena einen Beweis für die Existenz eines deterministischen Primzahltests liefern.[1]

Im folgenden werden die Grundlagen der Zahlentheorie eingeführt, um anschließend ausgewählte Algorithmen, die in der asymmetrischen Kryptographie eingesetzt werden, betrachten zu können. Zum Nachschlagen weiterer Informationen über Persönlichkeiten der Zahlentheorie mit historischer Einordnung bietet das Buch [18] von Jochen Ziegenbalg einen guten Überblick.

2. GRUNDLAGEN

In diesem Kapitel werden die Grundlagen zur Analyse von Primzahlen und dem diskreten Logarithmus gegeben. Dazu werden im allg. nur die Definitionen und Sätze geliefert. Für Beweise der hier angeführten Sätze sei auf die Bücher [2], [4], [10] und [6] verwiesen. Es wird vorausgesetzt, dass die Rechenvorschriften des Modulo bekannt sind.

Für diese Ausarbeitung wurden diverse Quellen unterstützend verwendet. Da es Notationsunterschiede der einzelnen Quellen gibt, führt dies dazu, dass sich nicht an die Notation aller genutzten Quellen gehalten werden kann. Speziell in [4] wird eine multiplikative Gruppe aller Einheiten in G mit G^X bezeichnet. Von dieser Notation wird abgesehen und die aus den anderen Quellen weiter verbreitete Notation G^* verwendet.

2.1 Algebraische Strukturen

In diesem Kapitel werden die algebraischen Strukturen: Halbgruppen, Gruppen, Ringe und Körper vorgestellt. Diese werden für ein späteres Kapitel benötigt. Die algebraischen Strukturen beschreiben ein abstraktes Rechnen mit Zahlen. Dies ermöglicht gezielter, nur die Rechenregeln an sich zu untersuchen, unabhängig von der Rechengröße und der jeweiligen Operation. Ein Anwendungsbereich ist u. a. in der Kryptographie zu finden. [11]

2.1.1 Halbgruppen

Eine Halbgruppe ist eine Menge M mit einer assoziativen Operation ∘, geschrieben mit (M, ∘) oder einfach nur M. Zur Erinnerung, das Assoziativgesetz besagt: (a ∘ b) ∘ c = a ∘ (b ∘ c) für alle a, b, c ∈ M. Dies gilt für sämtliche Elemente einer Halbgruppe. Das Zeichen ∘ ist Platzhalter für eine beliebige Operation. Der Wertebereich von ∘ ist eine Teilmenge von M, sodass a ∘ b ∈ M für alle a, b ∈ M gilt. Für das Zeichen ∘ werden auch die folgenden Operationszeichen verwendet: ∗, ·, +. Auch muss die Menge nicht zwangsläufig M sein. [4]

[*]Im Kurs Algorithmen und Komplexität
[†]B.Sc.
[‡]B.Sc.

Durch das Assoziativgesetz können also Klammern weggelassen werden. Zum besseren Verständnis folgen einige konkrete Beispiele von Halbgruppen [4]:

- $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$ sind Halbgruppen mit der Addition als Operation, ebenso wie mit der Multiplikation.
- Wenn a $\circ$ b = |b - a| für alle a, b $\in \mathbb{Z}$, dann ist ($\mathbb{Z}$, $\circ$) keine Halbgruppe, da in diesem Fall (1 $\circ$ 2) $\circ$ 3 = 1 $\circ$ 3 = 2 ist, aber 1 $\circ$ (2 $\circ$ 3) = 1 $\circ$ 1 = 0 ist. Ein Verändern der Klammerung ergibt unterschiedliche Ergebnisse. Somit ist das Assoziativgesetz nicht mehr gewährleistet, wodurch $\mathbb{Z}$ in diesem Fall keine Halbgruppe mehr sein kann.

Sobald es ein neutrales Element in einer Halbgruppe gibt, heißt dieses **Monoid**. Ein neutrales Element ist immer dann gegeben, wenn es ein e $\in$ M gibt, sodass für alle a $\in$ M gilt: a $\circ$ e = e $\circ$ a = a. Dieses weitere Axiom muss von jeder Halbgruppe erfüllt werden, um ein Monoid zu sein. Als Zeichen für ein Monoid wird neben e oft auch 1 verwendet, bei den Operationszeichen $\circ$, $\cdot$, $*$. Wird das + als Operationszeichen verwendet, ist oft 0 das neutrale Element. [4]

Wenn ein Monoid auch das folgende Axiom erfüllt, ist es eine **Gruppe**. Existiert für alle a $\in$ G ein b $\in$ G, sodass a $\circ$ b = b $\circ$ a = e gilt, so heißt b invers zu a. [4]

2.1.2 Ringe

In einem Ring als algebraische Struktur sind mehr als nur eine Operation vorhanden. Eine Menge R mit den zwei Operationen + und $\cdot$ auf R ist genau dann ein Ring, wenn die folgenden drei Bedingungen gelten [4]:

- (R, +) ist eine abelsche Gruppe. (Eine Operation $\circ$ auf einer Menge M heiß kommutativ oder abelsch, wenn a $\circ$ b = b $\circ$ a für alle a, b $\in$ M gilt.)
- (R, $\cdot$) ist ein Monoid
- Für alle a, b, c $\in$ R gilt: a(b + c) = ab + ac, (a + b)c = ac + bc (Distributivgesetze)

Zusätzlich heißt ein Ring kommutativ, wenn die Operation $\cdot$ kommutativ ist. Ein Ring besitzt also immer eine kommutative Addition und eine nicht notwendigerweise kommutative Multiplikation. Die beiden Distributivgesetze verbinden diese beiden Operationen miteinander. Ein Ring heißt nullteilerfrei wenn a $\cdot$ b = 0 ist und dadurch impliziert wird, dass a = 0 oder b = 0 sein muss, für alle a, b $\in$ R. Wenn es für ein a $\in$ R ein b gibt, so dass ab = ba = 1 gilt, dann ist a invertierbar oder eine Einheit. Alle Elemente im Monoid (R, $\cdot$) wo dies zutrifft, sind in einer multiplikativen Gruppe zusammengefasst, und werden als R^* bezeichnet. [4]

Es gibt spezielle Ringe, die sogenannten Körper. Ist eine Menge (K, +, $\cdot$) ein Ring und ist ($K/\{0\}$, $\cdot$) eine kommutative Gruppe, ist K ein Körper. Alle von Null verschiedenen Elemente sind Einheiten und es gilt: $K^* = K/\{0\}$. [4]

2.1.3 Integritätsbereiche

Wie in [5] angegeben ist ein Integritätsbereich ein kommutativer, nullteilerfreier Ring R mit Einselement. Aus dieser Definition ergeben sich die Integritätsbereiche der ganzen Zahlen $\mathbb{Z}$, der ganzen Gauß'schen Zahlen $\mathbb{Z}$[i] und des Polynomrings in einem unbestimmten X über einem Körper K, der K[X] bezeichnet wird.

$$\mathbb{Z}[i] = \{n + im \in \mathbb{C} : n, m \in \mathbb{Z}\}$$

$$K[X] = \{p(X) = a_0 + a_1X + a_2X^2 + ... + a_nX^n \mid a_i \in K, n \in \mathbb{N}\}$$

Des weiteren wird ein Integritätsbereich mit einer Funktion $\beta : R \longrightarrow \mathbb{N}$, für die gilt:

$$\mathrm{x} = \mathrm{qy} + \mathrm{r}, \quad \mathrm{q}, \mathrm{r} \in \mathrm{R}, \text{mit } \mathrm{r} = 0 \text{ oder } \beta(\mathrm{r}) < \beta(\mathrm{y})$$

als euklidischer Ring bezeichnet und lässt die Division mit Rest zu. Wie in [5] gezeigt, ist jeder der Ringe Z, Z[i] und K[X], für einen beliebigen Körper K, euklidisch. Die Erkenntnis, dass es weitere Integritätsbereiche als $\mathbb{Z}$ gibt, wird in den effizienten Primzahltests wieder aufgegriffen.

Die Teilbarkeit einer Zahl a $\in$ R durch b $\in$ R ohne Rest wird b | a geschrieben. Kann a durch b nicht ohne Rest geteilt werden, wird b $\nmid$ a geschrieben.

2.1.4 Restklassenringe in $\mathbb{Z}$

Restklassenringe $\mathbb{Z}/\mathrm{m}\mathbb{Z}$ oder auch $\mathbb{Z}_\mathrm{m}$ ergeben sich aus der Definition einer Addition und Multiplikation auf Restklassen. Eine Restklasse bezeichnet zwei Zahlen $\in \mathbb{Z}$ die bei der Division durch m $\in \mathbb{N}$ den gleichen Rest haben. Diese zwei Zahlen sind also in der gleichen Restklasse. Die Anzahl der Restklassen in einem Restklassenring ist gleich m. Die Äquivalenzrelation der beiden Zahlen bezüglich der Restklasse kann wie folgt definiert werden: Zwei Zahlen x, y $\in \mathbb{Z}$ heißen kongruent modulo m $\in \mathbb{N}$ wenn m | $x - y$. In Zeichen:

$$x \equiv y \ mod \ m$$

Die zugehörigen Beweise und Rechenvorschriften für Restklassenringe finden sich in [5].

2.2 Euklidischer Algorithmus

Der euklidische Algorithmus wird zur Berechnung des größten gemeinsamen Teilers zweier Zahlen benötigt. Der Algorithmus findet in fast allen weiteren Betrachtungen Anwendung und wird deshalb explizit angegeben. Wie in [12] bewiesen, kann der Algorithmus wie folgt definiert werden:

Sind m, n $\in \mathbb{N}$ zwei natürliche Zahlen mit m $\leq$ n und m $\nmid$ n, so gilt:

$$ggT(m, n) = ggT(n \ mod \ m, m).$$

2.3 Euler'sche φ-Funktion

Die Eulersche phi-Funktion φ(m) berechnet die Anzahl teilerfremder Zahlen für eine gegebene Zahl m > 1. Also gilt nach [5]:

$$\varphi(m) = Card((\mathbb{Z}/m\mathbb{Z})^*).$$

Mit Card(M) ist die Kardinalität der Menge M gemeint. Die Bezeichnung $(\mathbb{Z}/\mathrm{m}\mathbb{Z})^*$ steht für die Menge der Zahlen aus

$\mathbb{Z}/m\mathbb{Z}$, die ein multiplikatives Inverses besitzen. Geschrieben sieht die Menge wie folgt aus:

$$(\mathbb{Z}/m\mathbb{Z})^* = \mathbb{Z}_m^* = \{a \in \mathbb{Z}_m \mid ggT(a,m) = 1\}.$$

Die Menge $\mathbb{Z}_m^*$ kann auch Einheitengruppe genannt werden. Ein Beispiel für den Restklassenring modulo 10 ($\mathbb{Z}_{10}^*$) sind die Elemente 1, 3, 7 und 9. In [13] kann gut nachvollzogen werden, warum ein multiplikatives Inverses von a existiert, wenn der ggT(a,m) = 1 ist.

2.4 Elliptische Kurven Grundlagen

In diesem Kapitel sollen nur die Grundlagen von elliptischen Kurven näher gebracht werden, um so die **E**lliptic **C**urve **C**ryptography, kurz **ECC**, verstehen zu können. Der Vorteil beim ECC-Verfahren im Vergleich zum RSA-Verfahren liegt darin, dass die Schlüssellänge deutlich kürzer ausfallen kann, ohne dabei an Sicherheit zu verlieren. Ein RSA-Schlüssel mit 1024 Bit ist etwa so sicher wie ein Schlüssel aus einer elliptischen Kurve mit gerade mal ca. 160 Bit. Dazu kommt, dass der Rechenaufwand und Speicherbedarf beim ECC-Verfahren wesentlich geringer ist, als beim RSA-Verfahren. So kann ECC in Smartcards und Mobiltelefonen genutzt werden.[6]

Um die Funktionsweise der elliptischen Kurven in ihrer vollen Breite und Tiefe zu verstehen, ist eine sehr komplexe Mathematik notwendig. Innerhalb dieser Seminararbeit kann dieses Thema nicht breiter und tiefer durchleuchtet werden. Es wird nur ein Teilbereich der elliptischen Kurven betrachtet, der aus kryptografischer Sicht sehr interessant ist. Für weitergehende Informationen sei auf die folgende Literatur verwiesen: [6] und [10].

Eine elliptische Kurve ist eine ebene Kurve wie in Abbildung 1 gezeigt. Sie wird durch eine Gleichung der Form: $y^2 = x^3 + ax + b$ beschrieben. Damit ist eine Menge Z aller Punkte P(x, y), die auf der elliptischen Kurve liegen, definiert. Wichtig dabei ist, dass die Kurvenparameter a und b so gewählt sind, dass die partiellen Ableitungen nach x und nach y auf keinem Punkt der Kurve gleichzeitig null sind. Dazu später mehr.

Das Addieren von zwei Punkten, die auf der elliptischen Kurve liegen, ergibt wieder einen Punkt, welcher ebenfalls auf der Kurve liegt [6]. Mit Addition ist das Verknüpfen von zwei Punkten gemeint. Man könnte es auch als Multiplikation bezeichnen. In beiden Fällen hat es nichts mit den bekannten Operationen auf Zahlen zu tun. Das Addieren von zwei Punkten ist vielmehr geometrisch definiert, siehe dazu auch Abbildung 1:

> Definition 1. *Durch die gegebenen Punkte P und Q wird eine Gerade gelegt, welche die Kurve in einem dritten Punkt R schneidet. Dieser wird anschließend an der x-Achse gespiegelt. Als Ergebnis erhält man den Punkt S, welcher als Addition von P und Q bezeichnet wird.[6]*

Die so definierte Addition ist kommutativ. Zur Erinnerung: P + Q = Q + P. Nicht für alle elliptischen Kurven kann eine Addition von Punkten durchgeführt werden. Wie oben

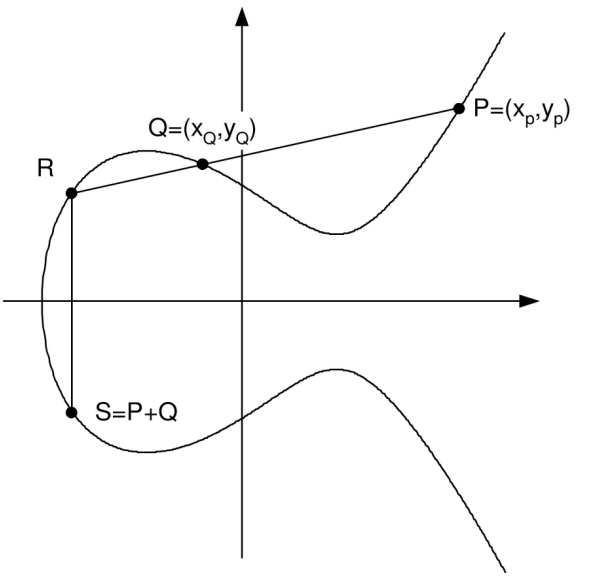

Abbildung 1: Addition von zwei Punkten auf einer elliptischen Kurve [6]

bereits erwähnt, dürfen die partiellen Ableitungen nach x und nach y auf keinem Punkt der Kurve gleichzeitig null sein. Anders ausgedrückt: Die Kurve darf sich nicht selbst schneiden, ansonsten kann die Additionsoperation nicht für beliebige Punkte durchgeführt werden. Zusätzlich muss beachtet werden, dass bei einer Addition von zwei Punkten die nachfolgenden Spezialfälle auftreten können[6]:

- Wenn für die beiden zu addierenden Punkte Q = P gilt, wird die Tangente an der Kurve im Punkt P verwendet. Dabei entsteht der Schnittpunkt mit der Kurve in R und durch Spiegelung resultiert daraus S = P + P = 2P.
- Sollten die X-Koordinaten beider zu addierender Punkte gleich sein, sodass ($Q_X = P_X$) gilt, entsteht eine vertikale Gerade und die Kurve wird kein weiteres Mal geschnitten. Für diesen Fall wird die elliptische Kurve um einen weiteren Punkt ∞, welcher im Unendlichen liegt, ergänzt. Die Addition von Punk P mit ∞ ist so definiert, dass man wiederum P als Ergebnis erhält (P + ∞ = P). Somit ist ∞ das neutrale Element der Addition. Es gilt also: P + Q = ∞ wenn die x-Koordinaten von P und Q gleich sind. Daraus folgt, dass Q das inverse Element von P ist und es gilt: Q = -P.

Das Addieren eines Punktes P mit einem Skalar k $\in$ {1, 2, 3 ...} wird als wiederholte Addition definiert:

$$kP = P1 + P2 + ... + Pk$$

2.4.1 Asymmetrische Verschlüsselung mit Elliptischen Kurven

Um Elliptische Kurven für Asymmetrische Verschlüsselung einsetzen zu können, muss in einem endlichen Körper gerechnet werden, um Rundungsfehler zu vermeiden. Bei der Addition und Multiplikation in endlichen Körpern sind diese so definiert, dass das Ergebnis immer wieder ein Element des endlichen Körpers ist. Aufgrund dessen muss eine weitere Operation durchgeführt werden: *mod* $|Z|$. Dies stellt sicher, dass der resultierende Rest in jedem Fall wieder ein Element aus Z ist. Für die Addition besitzt jedes Element ein inverses Element -a, damit gilt für die Subtraktion: $b - a = b + (-a)$. Bei der Multiplikation ist das inverse Element a^{-1}, damit gilt für die Division: $b/a = b \cdot a^{-1}$. Für ein konkretes Beispiel sei an dieser Stelle auf S. 254 - 257 in [6] verwiesen.

Um elliptische Kurven für kryptologische Anwendungsfälle zu nutzen, muss die Ordnung eines Punktes der elliptischen Kurve berechnet werden.

DEFINITION 2. *Die Ordnung eines Punktes ist die Anzahl der Punkte, die durch fortwährender Addition dieses Punktes erzeugt werden.[6]*

Dazu folgendes Beispiel:

$$P + P = 2P \Rightarrow 2P + P = 3P \Rightarrow ... \Rightarrow xP + P = (x + 1)P$$

P ist dabei immer ein Punkt auf der elliptischen Kurve. Irgendwann ist $x_i P = \infty$, somit kein Punkt mehr auf der Kurve, und damit hat der Punkt P die Ordnung x_i, im weiteren Verlauf als n bezeichnet.

2.4.2 Schlüsselaustausch mit elliptischen Kurven

Zuerst muss der Körper und eine elliptische Kurve bestimmt werden. Dazu wählt man eine große Primzahl und die Kurvenparameter a und b. Weiter wird nun ein Erzeugerpunkt G vereinbart. Dabei soll die Ordnung des Punktes G möglichst groß und eine Primzahl sein. A wählt eine geheime ganze Zahl n_A, welche kleiner sein muss als n und berechnet daraus den öffentlichen Schlüssel $P_A = n_A \cdot G$. Teilnehmer B macht das Gleiche jeweils mit n_B und P_B. P_A und P_B können über eine unsichere Leitung ausgetauscht werden. Nun kann Teilnehmer A den Schlüssel $K = n_A \cdot P_B$ berechnen. B berechnet ebenfalls K mit n_B und P_A. So haben beide ein und das selbe geheime K berechnet.

Es folgt der Beweis, dass A und B wirklich das gleiche K berechnet haben müssen:

BEWEIS 1. *A berechnet $K = n_A \cdot P_B$. P_B wurde ursprünglich von Teilnehmer B berechnet mit $n_B \cdot G$. Dies kann in die Berechnung für K von A anstelle von P_B eingesetzt werden, sodass daraus folgt: $K = n_A \cdot P_B = n_A \cdot (n_B \cdot G)$. Das gleiche Prinzip angewendet für die Berechnung von K durch Teilnehmer B ergibt: $K = n_B \cdot P_A = n_B \cdot (n_A \cdot G)$. Unter Berücksichtigung des Assoziativgesetzes können diese beiden Gleichungen gleich gesetzt werden:*

$$n_A \cdot (n_B \cdot G) = n_B \cdot (n_A \cdot G)$$

Asymmetrische Verschlüsselungsverfahren basieren auf Einwegfunktionen, wobei es nicht allzu schwierig ist $k \cdot P$ zu berechnen. Allerdings ist das Berechnen von k aus $k \cdot P$ und P sehr aufwendig. Anzumerken ist, dass diese Aussage allerdings bis heute noch nicht bewiesen wurde.

3. PRIMZAHLEN

Natürliche Primzahlen werden definiert durch Zahlen > 1, die nur durch Eins oder sich selbst teilbar sind. Wie im Kapitel Primfaktorzerlegung gezeigt, können alle natürlichen Zahlen mit einer Multiplikation von Primzahlen erzeugt werden. Sie bilden sozusagen die Bausteine aller natürlichen Zahlen. Die Unberechenbarkeit, mit der sie auftreten, gibt Mathematikern schon seit Jahrtausenden Rätsel auf und ist ein Grundstein unserer heutigen Verschlüsselungsverfahren. In anderen Zahlensystemen als den natürlichen Zahlen ist die gewohnte Definition von Primzahlen nicht vollständig/korrekt. Sie sagt nur etwas über die Irreduzibelität eines Elements in einem Integritätsbereich aus. Da in den natürlichen Zahlen aber jedes irreduzibles Element auch prim ist, reicht diese Definition für $\mathbb{N}$ aus. Für alle Integritätsbereiche gilt für die Primheit folgende Definition nach [5]:

Ein Element $p \in R \setminus (R^* \cup \{0\})$ heißt prim oder Primelement, wenn für alle $a, b \in R \setminus \{0\}$ gilt:

$$p \mid ab \Longrightarrow p \mid a \text{ oder } p \mid b.$$

Mit Hilfe der Primzahlen kann der Restklassenring $\mathbb{Z}/m\mathbb{Z}$ spezialisiert werden. Ist m eine Primzahl p, so ist $\mathbb{Z}/p\mathbb{Z}$ ein Körper und wird auch $\mathbb{F}_p$ bzw. $\mathbb{GF}_p$ bezeichnet.

Neben den normalen Primzahlen gibt es weitere sogenannte Pseudoprimzahlen. Diese Primzahlen verhalten sich bezogen auf einen Algorithmus genauso wie echte Primzahlen, sie sind jedoch zusammengesetzt. Ein Beispiel für solche Zahlen sind Carmichaelzahlen, die in einem späteren Kapitel thematisiert sind.

Leider gibt es keine bekannten effizienten Verfahren, um Primzahlen zu generieren. Dennoch kann man Primzahlen recht einfach raten. Grundsätzlich ist es möglich, eine große Anzahl von Zahlen auszuschließen. Man denke an: nur ungerade Zahlen, die Teilungsgesetze und die Tatsache, dass alle Primzahlen > 3 in der Form $4k + 1$ oder $4k + 3$, $k \in \mathbb{N}$ vorliegen. Alle geratenen Zahlen müssen jedoch einem Primzahltest (vgl. Kapitel 3.5) zur Verifizierung unterzogen werden. In [16] ist beschrieben, wie ein Generierungsprozess erfolgen kann.

3.1 Primfaktorzerlegung

Jede natürliche Zahl größer als Eins, kann als Produkt von Primzahlen geschrieben werden. Dieser zentrale Satz wird als Fundamentalsatz der Zahlentheorie bezeichnet und kann auf jeden euklidischen Ring R angewendet werden. Eine Primfaktorzerlegung wird mit $x \in R$, $u \in R^*$, $e_1, e_2, ..., e_m \in \mathbb{N}$ wie folgt definiert:

$$x = u \cdot p_1^{e_1} \cdot p_2^{e_2} \cdot ... \cdot p_m^{e_m},$$

wobei $p_1 < p_2 < ... < p_m$ Primelemente sind.

Neben der Existenz einer solchen Primfaktorzerlegung ist auch die Eindeutigkeit, die durch das Vergleichszeichen "klei-

ner als“ implizit angegeben ist, von entscheidender Bedeutung. In [8] und [5] wird die Existenz und Eindeutigkeit bewiesen.

Die meisten kryptographischen Algorithmen bauen auf die ineffiziente Berechnung der Primfaktoren. Es ist zwar einfach, zwei Primzahlen miteinander zu multiplizieren, aber es ist schwer, aus der multiplizierten Zahl die beiden Primfaktoren zurückzugewinnen. Kennt man jedoch eine der beiden Primzahlen, so berechnet sich die zweite durch einfache Division. Zur Verschlüsselung ergibt sich damit die Idee einer sogenannten Falltür-Funktion, die die Primfaktoren mit Hilfe eines geheimen Hinweises schnell berechnen kann.

3.2 Satz von Euler und Kleiner Satz von Fermat

Die Vermutung vom kleinen Satz von Fermat wurde von Fermat im 17. Jahrhundert aufgestellt und von Euler bewiesen. Euler konnte zudem zeigen, dass der kleine Satz von Fermat nur ein Spezialfall für Primzahlen darstellt. Der Satz von Euler lautet:

Sei m $\geq 2 \in \mathbb{N}$. Dann gilt für jede zu m teilerfremde Zahl a $\in \mathbb{N}$

$$a^{\varphi(m)} \equiv 1 \; mod \; m.$$

Denn wie in [5] anhand der Gruppentheorie bewiesen, ist

$$a^{Ord(G)} = a^{|G|} = e,$$

wobei G eine endliche Gruppe ist, a $\in$ G und e das Einselement (neutrales Element einer multiplikativen Gruppe). Da das Einselement in $\mathbb{N}$ 1 ist, folgt: Kongruenz 1 modulo m.

Des weiteren ergibt sich mit

$$\varphi(p) = Card((\mathbb{Z}/p\mathbb{Z})^*) = |\mathbb{F}_p^*| = p - 1$$

sofort der kleine Satz von Fermat:

$$a^{p-1} \equiv 1 \; mod \; p$$

Mit dem Satz von Euler und Fermat kann nun ein erster primitiver Primzahltest definiert werden. Für ein beliebiges a $\in \mathbb{N}$ mit 1 < a < m gilt:

$$a^{m-1} \not\equiv 1 \; mod \; m \Longrightarrow nicht \; prim$$

Leider ist der fermatsche Primzahltest nur ein negativ Test, der eindeutig zeigen kann, dass eine Zahl nicht prim ist. Wenn der Test einer Zahl kongruent 1 modulo m ergibt, muss m also nicht prim sein. Eine Zahl, die für den Test prim ist, ist entweder eine echte Primzahl oder eine Pseudoprimzahl m zur Basis a.[18]

3.3 Carmichaelzahlen

Da Pseudoprimzahlen die Primalität immer zu einer Basis aufweisen, liegt der Versuch nahe, andere Basen zu suchen, für die der fermatsche Test nachweisen kann, dass die vermeintliche Primzahl gar keine ist. Dies ist der Grundbaustein für probabilistische Primzahltests. Dennoch gibt es Zahlen, für die der fermatsche Test bei allen teilerfremden Basen kongruent eins Äquivalenz feststellt. Diese starken fermatschen Pseudoprimzahlen heißen Carmichaelzahlen.

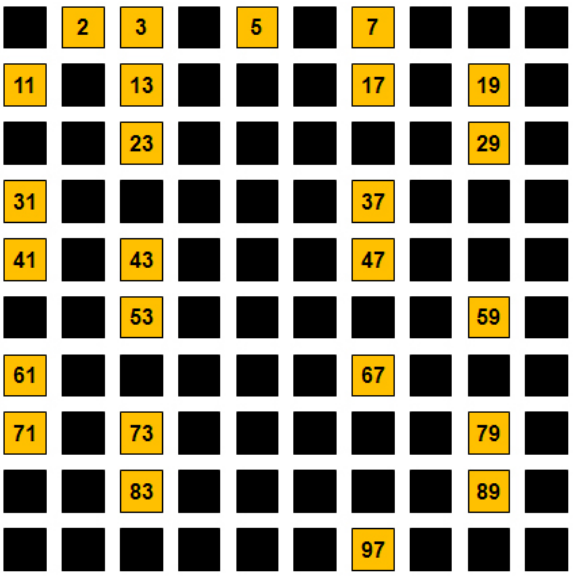

Abbildung 2: Primzahlen bis 100 Quelle: [7]

Carmichaelzahlen werden nach [16] so definiert:

Eine zusammengesetzte Zahl m $\in \mathbb{N}$, m ≥ 3, heißt Carmichaelzahl genau dann, wenn für alle Basen a mit ggT(m, a) = 1 gilt:

$$a^{m-1} \equiv 1 \; mod \; m$$

Die weiteren zwei folgenden Eigenschaften seien zudem genannt, da sie für den probabilistischen Primzahltest von Miller und Rabin von entscheidender Bedeutung sind.

Eine zusammengesetzte Zahl m $\geq 3 \in$ N ist eine Carmichaelzahl, wenn m keine mehrfachen gleichen Primfaktoren enthält und für jeden Primfaktor p | m auch

$$p - 1 \mid m - 1$$

gilt.

3.4 Sieb des Eratosthenes

Das Sieb des Eratosthenes ist eines der ältesten bekannten Verfahren, um Primzahlen zu erhalten. Es berechnet für ein gegebenes n $\in \mathbb{N}$ alle Primzahlen p $\in \mathbb{N}$, für die gilt 1 < p < n. Der Algorithmus iteriert dazu über alle nicht gestrichenen Zahlen von 1 bis $\sqrt{n}$. Als erstes wird die Zahl Zwei genommen, sie ist nicht gestrichen und somit eine Primzahl. Dann werden alle Vielfache von Zwei gestrichen (die geraden Zahlen). Anschließend sucht der Algorithmus die nächste nicht gestrichene Zahl. In diesem Fall ist es die Drei und es werden wieder alle Vielfache von drei gestrichen. Das Verfahren wiederholt sich, bis die Suche $\sqrt{n}$ erreicht hat. Alle nicht gestrichen Zahlen sind die Primzahlen bis n. Die Abbildung 2 illustriert die Primzahlen bis 100. Wie in [16] gezeigt, hat das Sieb des Eratosthenes eine subexponentielle Laufzeit, da die Laufzeit von der Länge der Eingabe abhängt. Aus diesem Grund eignet es sich nicht für einen praktischen Primzahltest.

3.5 Effiziente Primzahltests

Die in diesem Kapitel vorgestellten Algorithmen zum Erkennen von Primzahlen sind effiziente Algorithmen im Sinne der Komplexitätsklasse P und im Gegensatz zum fermatschen Test nicht anfällig für starke Pseudoprimzahlen. Dennoch unterscheiden sich die beiden vorgestellten Verfahren deutlich voneinander. Neben der Funktionsweise wird auch die Laufzeit gegenübergestellt.

3.5.1 Miller-Rabin-Test

Der Miller-Rabin-Test ist ein probabilistischer Primzahltest und gehört zu den Montecarlo- Algorithmen. Im Gegensatz zum fermatschen Primzahltest ist der Miller-Rabin-Test nicht so anfällig für Carmichaelzahlen, denn es können immer Basen gefunden werden, die die Carmichaelzahlen entlarven. Dieser Test wird am häufigsten in Primzahlgeneratoren eingesetzt und hat damit die größte praktische Relevanz. Nach der Vorstellung des Tests werden die Unterschiede zum fermatschen Test herausgearbeitet.

Der Test lautet wie folgt:

Wenn $m \in \mathbb{N} > 9$ eine ungerade Zahl und $m - 1 = 2^t n$ mit ungeradem n ist, dann gilt für jede teilerfremde Zahl $a \in \mathbb{Z}$:

$$a^n \equiv 1 \ mod\ m \ oder\ \exists s \in \{0, ..., t-1\} : a^{2^s n} \equiv -1 \ mod\ m \quad (1)$$

dann ist m prim oder nicht prim.

Aber wenn m nicht prim ist, so gilt außerdem für die Menge A aller zu m teilerfremden a, $0 < a < m$ die (1) erfüllen:

$$Card(A) \leq 1/4\varphi(m)$$

Durch diese zusätzliche Aussage kann eine Wahrscheinlichkeit angegeben werden, mit der die getestete Zahl prim ist. Wiederholt man den Test t mal, ergibt sich somit eine Wahrscheinlichkeit von $1/4^t$. Nach nur zehn Iterationen ist die Wahrscheinlichkeit für eine zusammengesetzte Zahl kleiner als 10^{-6}.

Um die Ausdrücke in (1) zu verstehen müssen folgende Vorüberlegungen gemacht werden. Jeder Körper F_p^* kann nach [5] in die zwei Teile (Untergruppen) der Quadrate und nicht Quadrate geteilt werden. Jede dieser Gruppen besteht aus $Card(F_p^*)/2 = (p-1)/2$ Elementen. Dies folgt aus der Tatsache, dass $x^2 \equiv$ a mod p in F_p^* nur die Lösungen +1 oder -1 hat. Ein Element ist ein Quadrat, wenn es gleich dem quadratischen Rest eines x^2 mod p ist.

Eine weitere Überlegung ergibt sich aus dem umgekehrten Satz des Eulerkriteriums, der wie folgt lautet: Sei $m \geq 3$ eine ungerade ganze Zahl, so dass (m - 1)/2 ungerade ist. Es gilt dann für jede zu m teilerfremde ganze Zahl a

$$a^{(m-1)/2} \equiv \ ^+_-1 \ mod\ m,$$

so dass m eine Primzahl ist.

Wenn (m - 1)/2 gerade ist, kann diese Aussage nicht getroffen werden. Für den Fall, dass m eine Carmichaelzahl ist, ist auch $m - 1 = 2^{s_i} u_i (p_i - 1)$, da alle Primfaktoren m-1 teilen, wobei u_i ungerade ist. Es gilt dann O.B.d.A mit $s_1 \leq s_2 \leq ... \leq s_r$ und $s = s_1$:

$$a^{(m-1)/2^s} \equiv 1 \ fuer\ alle\ a \in \mathbb{Z}_m^*$$

Anschließend müssen wie in [5] durchgeführt, mehrere Fallunterscheidungen für gerade und ungerade vielfache von $(a^{(m-1)/2^{s+1}})$ durchgeführt werden. Dabei können die Ergebnisse der Kongruenzen in Teilmengen eingeteilt werden. Die Kardinalität der Mengen gibt dabei die Wahrscheinlichkeit der Zusammengesetzten Zahlen vor.

Etwas abstrakter dargestellt zielt der Test darauf ab, im Gegensatz zu dem fermatschen Test, die Quadratwurzeln der Carmichaelzahlen in dem Primzahlkriterium zu berücksichtigen. Durch diese Erweiterung kommt es nur noch maximal bei einem Viertel der Basen zur Annahme, es handle sich um eine Primzahl.

Wie in [16] angegeben, benötigt der Test, da die Exponentiation effizient berechenbar ist, O(t * (log m)) arithmetische Operationen und $O(t * (\log m)^3)$ Bit Operationen bezogen auf die Eingabelänge von m. Damit liegt der Test in der polynomialen Laufzeitklasse.

Eine interessante Aussage die in [5] getroffen wird, ist dass der Miller-Rabin-Test unter Annahme der Riemannschen Vermutung zu einem deterministischen polynomialen Test wird.

3.5.2 AKS-Test

Der 2004 von M. Agrawal, N. Kayal und N. Saxena entwickelte AKS-Test ist ein deterministischer in polynomialer Zeit durchführbarer Primzahltest. Bei diesem Test stellt die Rechnung in einem Polynomring $(\mathbb{Z}/m)[X]$ den größten Unterschied zu den bisherigen Tests da. Besonders wertvoll ist hier die notwendige und hinreichende Bedingung für die Primmalität des folgenden Satzes.

Der Satz lautet in [5] wie folgt: Sei $m > 1$ eine natürliche Zahl und a eine zu m teilerfremde ganze Zahl. Genau dann ist m prim, wenn

$$(X + a)^m \equiv X^m + a \ mod\ m$$

Dieser Satz ist natürlich noch nicht schneller als die Probedivision bis $\sqrt{m}$, aber folgende Erweiterungen führen zu einem polynominelle Laufzeitverhalten. Die Idee hinter der Erweiterung ist, ein Polynom zu finden, das mit getestet wird und dadurch die Laufzeit reduziert. Ein solches Polynom ist $P(x) = x^r - 1$, r ist prim . Mit diesem Zusatz ist der Test wiederum nur ein notwendiges Kriterium für eine Primzahl. Dies kann verhindert werden, indem versucht wird, eine Menge Zahlen zu finden, in der mindestens ein Element für eine beliebige zusammengesetzte Zahl m das Kriterium nicht erfüllt. Diese Menge kann angegeben werden und lautet wie folgt:

$$A(r, m) = \{1, 2, ..., 2 * \lfloor log\ m * \sqrt{\varphi(r)} \rfloor\}$$

Anders ausgedrückt muss jedes Element in der Menge A(r,m) die weiter unten angeführte Bedingung (2) erfüllen, damit m prim sein kann.

Mit diesen Überlegungen kann im folgenden der Satz von Agrawal, Kayal und Saxena nach [5] aufgestellt werden.

Sei $m > 1$ eine ungerade Zahl. Weiter sei r eine Primzahl mit folgenden Eigenschaften:

1. m hat keinen Primteiler $\leq$ r.

2. $\text{ord}_{(\mathbb{Z}/\text{r})^*}(\text{m}) > (\log_2 \text{m})^2$.

Gilt dann für alle $\text{a} = 1, 2, ..., \text{A} := \lfloor \sqrt{\text{r}} \log_2 \text{m} \rfloor$

$$(X + a)^m \equiv X^m + a \ mod \ (m, X^r - 1), \tag{2}$$

so ist m eine Primzahlpotenz, d.h. $\text{m} = \text{p}^\text{k}$, p prim, $\text{k} \geq 1$.

Wie in [5] und [16] beschrieben, lässt sich der vollständige Test so ausdrücken:

Sei $\text{m} > 2^{24}$ eine ganze Zahl. Um zu entscheiden, ob m prim ist, gehe man wie folgt vor:

1. Im Intervall $(\log_2 \text{m})^2 < \text{r} < (\log_2 \text{m})^5$ suche man nach einer Primzahl r mit $\text{r} \nmid \text{m}$ und $\text{ord}_{(\mathbb{Z}/\text{r})^*}(\text{m}) > (\log_2 \text{m})^2$. Falls es kein solches r gibt, ist m nicht prim.

2. Man vergewissere sich, dass m keinen Primteiler $\text{p} \leq \text{r}$ besitzt.

3. Für alle ganzen Zahlen a mit $1 \leq \text{a} \leq \lfloor \sqrt{\text{r}} \log_2 \text{m} \rfloor$ überprüfe man die Kongruenz (2). Ist diese Kongruenz auch nur für ein einziges a nicht erfüllt, ist m nicht prim.

4. Hat m die Tests 1. bis 3. bestanden, ist m prim oder eine Primzahlpotenz. Man hat deshalb noch auszuschließen, dass sich m als Potenz $\text{m} = \text{n}^\text{k}$ einer ganzen Zahl n mit einem Exponenten $\text{k} \geq 2$ darstellen lässt.

Wie in [5] und [16] gut dargestellt, beträgt die Gesamtlaufzeit des Algorithmus unter bewiesenen Annahmen $\text{O}(\log^{7,5}\text{m})$. Schritt 3 spielt bei der praktischen Laufzeitbetrachtung die größte Rolle, da hier sehr große Polynome miteinander verrechnet werden müssen. Leider eignet sich der AKS-Test dadurch nicht für die Praxis. Für die theoretische Betrachtung war die Entdeckung des AKS-Test jedoch ein großer Erfolg, da nach sehr langer Zeit bewiesen werden konnte, dass die Primzahlerkennung in polynomieller Zeit durchführbar ist.

4. DISKRETER LOGARITHMUS

In diesem Kapitel soll der diskrete Logarithmus betrachtet werden. Es werden Anwendungsbeispiele aufgezeigt und genau erläutert, weswegen sich der diskrete Logarithmus besonders gut in der Kryptografie als Einwegfunktion eignet. Zu Beginn wird das grundsätzliche Problem beim diskreten Logarithmus an einem Beispiel praxisnahe erläutert, um im Anschluss auf algorithmische Lösungen einzugehen, wie der diskreten Logarithmus in $\mathbb{Z}_\text{p}^*$ und auf elliptische Kurven berechnet werden kann. Hierfür wird der Baby-Step-Giant-Step-Algorithmus vorgestellt. Für alle Berechnungen ist dabei entscheidend, dass p wirklich prim ist. Durch die zuvor vorgestellten Primzahltests ist dies jedoch zu einer beliebigen Wahrscheinlichkeit gewährleistet.

4.1 Das Problem des diskreten Logarithmus im Detail

Es existiert eine Primzahl p, ein erzeugendes Element g für $\mathbb{Z}_\text{p}^*$ sowie eine ganze Zahl x. Zu der diskreten Exponentialfunktion $g^x \ mod \ p$ gibt es die diskrete Logarithmusfunktion, die zu einem gegebenen y und g, x beschreibt. Somit ist x der diskrete Logarithmus von y zur Basis g, $(y = g^x \ mod \ p)$. Jede Zahl aus $\mathbb{Z}_\text{p}^*$ lässt sich als Potenz von g darstellen, wenn g ein erzeugendes Element von $\mathbb{Z}_\text{p}^*$ ist. Ist dies nicht der Fall, so muss es nicht zu jedem $y \in \mathbb{Z}_\text{p}^*$ einen diskreten Logarithmus geben. [2] Um das eigentliche Problem des diskreten Logarithmus zu verstehen ist es hilfreich, die Logarithmen in $\mathbb{R}$ mit Logarithmen in $\mathbb{Z}_\text{p}^*$ gegenüberzustellen.

$$\begin{aligned} y &= g^x \\ 1024 &= 2^x \\ x &= log_2 1024 \\ x &= 10 \end{aligned} \tag{3}$$

$$\begin{aligned} y &= g^x \ mod \ p \\ 10 &= 2^x \ mod \ 11 \\ x &= log_2 10 \\ x &= 3.32193... \ \lightning \end{aligned} \tag{4}$$

Die Gleichungen aus (3) zeigen, wie der Logarithmus von x zur Basis g mit der Logarithmusfunktion berechnet werden kann. Auf diese Weise sind Gleichungen für die positiven reellen Zahlen $\mathbb{R}^+$ immer eindeutig lösbar. Für die Gleichungen aus (4) wird ebenfalls mit der Logarithmusfunktion versucht, den Logarithmus von 10 zur Basis 2 zu erhalten. Als Ergebnis erhält man eine Zahl, die nicht in $\mathbb{Z}_\text{p}^*$ enthalten ist, was auch nicht verwundert, da die Logarithmusfunktion *mod* 11 gar nicht berücksichtigt. Genau hier ist das grundsätzliche Problem beim diskreten Logarithmus. Es gibt keine mathematische Rechenoperation, die es ermöglicht, den diskreten Logarithmus in einem endlichen Körper mit nur einem Rechenschritt zu berechnen. Um dennoch eine Lösung zu erhalten, scheint das Enumerationsverfahren das naheliegendste zu sein. Hierbei werden einfach alle Werte, die für x in Frage kommen, durchprobiert. So ist die Lösung der Gleichungen aus (4), $x = 5$. [14] Für sehr große Gruppen, wo x beispielsweise eine 160 Bit große Zahl ist, gibt es bis heute keine Algorithmen, die den diskreten Logarithmus effizient berechnen.[2] Es gibt allerdings eine ganze Reihe von Algorithmen, die in der Lage sind, den diskreten Logarithmus gezielter zu berechnen, als das naive Ausprobieren. Siehe hierfür Kapitel 4.2.

4.1.1 Diskreter Logarithmus auf elliptische Kurven

In Kapitel 2.4 wurde die Addition von Punkten und Skalaren auf elliptische Kurven beschrieben. In diesem Unterkapitel soll kurz beschrieben werden, was genau der Logarithmus auf einer elliptischen Kurve über $\mathbb{Z}_\text{p}^*$ ist. Für eine elliptische Kurve E über den Primkörper $\mathbb{Z}_\text{p}^*$ mit den Punkten $P, Q \in E(\mathbb{Z}_\text{p}^*)$ gibt es ein $k \in \mathbb{Z}_\text{p}^*$, sodass die folgende Gleichung erfüllt ist:

$$Q = kP$$

So wird die Zahl k als diskreter Logarithmus von Q zur Basis P bezeichnet. Den diskreten Logarithmus aus $E(\mathbb{Z}_\text{p}^*)$ zu bestimmen, ist noch einmal ungleich komplexer als aus $\mathbb{Z}_\text{p}^*$. Dieses Berechnungsproblem wird **E**lliptic **C**urve **D**iscrete **L**ogarithm **P**roblem, kurz **ECDLP**, genannt.

4.2 Baby-Step-Giant-Step-Algorithmus

In der Praxis ist es mit diesem Algorithmus nicht möglich, eine Verschlüsselung, die auf dem DLP aufbaut, zu brechen. Die Komplexität dieses Algorithmus liegt in $O(\sqrt{p-1})$ und ist somit schon deutlich besser, als eine naive vollständige Suche, deren Komplexität in $O(p-1)$ liegt, und dennoch zu stark von der Größe der zugrundeliegenden Gruppe abhängig ist. Anzumerken ist, dass dieser Algorithmus ein sogenannter generischer Algorithmus ist, womit dieser für jede

Gruppe funktioniert und nicht von einer speziellen Struktur der Gruppe abhängt. [2]

Zuerst wählt der Algorithmus eine Zahl t $\in \mathbb{N}$, sodass $t \geq \sqrt{p-1}$ ist. Mit einer zweiten Zahl r $(0 \leq r < t)$ lässt sich die Gleichung des diskreten Logarithmus schreiben als:

$$x = q \cdot t + r \tag{5}$$

und lässt sich wie folgt umformen:

$$y = g^x = g^{q \cdot t + r} \;\Leftrightarrow\; y \cdot g^{-r} = g^{q \cdot t} \tag{6}$$

Nun müssen die zwei Zahlen r und q gesucht werden, sodass die Gleichung $y \cdot g^{-r} = g^{q \cdot t}$ gilt. Dazu werden zwei Listen angelegt und im Nachhinein werden die Einträge miteinander verglichen.

Baby-Step Liste: $y \cdot g^{-r}$ für alle r mit $0 \leq r < t$

Giant-Step Liste: $g^{q \cdot t}$ für alle q mit $0 \leq q < t$

Sollte ein Eintrag vorhanden sein, der in beiden Listen vorkommt, liefert dieser den diskreten Logarithmus x. [2]

4.2.1 Baby-Step-Giant-Step Vorgehen

Zuerst errechnet man t mit $\sqrt{p-1}$. Nun werden alle Baby-Steps benötigt, diese werden mit Gleichung (7) ermittelt:

$$B = \{\ (x \cdot g^{-r}\ mod\ p,\ r)\ :\ 0 \leq r < t\ \} \tag{7}$$

Die Ergebnisse von (7) mit dem dazugehörigen r werden in einer Liste gespeichert. Mit Gleichung (8) werden nun die Giant-Steps berechnet.

$$G = \{\ g^{t \cdot q}\ mod\ p\ :\ q = 1, 2, 3, ...\ \} \tag{8}$$

Die so ermittelten Lösungen von (8) werden mit denen aus der Liste der Baby-Steps verglichen. Stimmen Baby-Step und Giant-Step überein, wurde eine Kollision gefunden und die Gleichung aus (6) ist erfüllt. Die so ermittelten Werte für q und r können nun in Gleichung (5) eingesetzt werden und man erhält so den diskreten Logarithmus von y.

4.2.2 Baby-Step-Giant-Step zum Lösen des ECDLP

Am grundsätzlichen Vorgehen beim Baby-Step-Giant-Step-Algorithmus auf elliptische Kurven ändert sich nicht viel, weswegen hier nur noch einmal die Unterschiede gezeigt werden. Die Baby-Steps werden nach Gleichung (9) berechnet:

$$B = \{\ (\ Q - rP,\ r\ )\ :\ 0 \leq r < t\ \} \tag{9}$$

Diese werden in einer Liste gespeichert, um sie in einem weiteren Schritt mit den Giant-Steps zu vergleichen:

$$G = \{\ (\ qmP,\ q\ )\ :\ 0 \leq q < t\ \} \tag{10}$$

Bei gefundener Kollision kann aus q und r, $k = qm + r$ errechnet werden und man erhält den diskreten Logarithmus von Q.

4.3 Index-Calculus-Algorithmus

Der Index-Calculus-Algorithmus kann das DLP „besser“ lösen als der Baby-Step-Giant-Step-Algorithmus. Es sei an dieser Stelle schon vorweggenommen, dass auch dieser vom effizienten Lösen noch weit entfernt ist. Ausgangsbasis ist die schon aus 4.1 bekannte Gleichung (11) zum diskreten Logarithmus aus $\mathbb{Z}_p^*$.

$$y = g^x\ mod\ p \tag{11}$$

Jedes $h \not\equiv 0\ (mod\ p)$ kann als $h \equiv g^k$ geschrieben werden, womit eindeutig $mod\ p-1$ bestimmt wird. Der diskrete Logarithmus von x wird als $L(x)$ geschrieben, sodass $x = L(x)$ gilt. Daraus folgt:

$$g^{L(h)} \equiv h \quad (mod\ p) \tag{12}$$

Weiter gelten für alle Werte $x_1, x_2 \in \mathbb{Z}_p^*$:

$$g^{L(h_1 h_2)} \equiv h_1 h_2 \equiv g^{L(h_1) + L(h_2)} \quad (mod\ p) \tag{13}$$

Der diskrete Logarithmus des Produktes aus zwei Zahlen ist die Summe der diskreten Logarithmen der einzelnen Zahlen:

$$L(h_1 h_2) \equiv L(h_1) + L(h_2) \quad (mod\ p) \tag{14}$$

Die Grundidee beim Index-Calculus-Algorithmus ist, aus der Kenntnis des diskreten Logarithmus für einige Zahlen, kann über die passenden Gleichungen der diskrete Logarithmus von beliebigen Zahlen berechnet werden.[14]

Zunächst bildet man eine Faktorbasis Menge B aus kleinen Primzahlen. Als nächstes müssen passende Gleichungen gefunden werden, bei denen die Primfaktorzerlegung von g^x nur aus Potenzen von Elementen aus B besteht. Ziel ist es nun, ein Gleichungssystem aus Potenzen von Elementen aus B zu lösen. Durch Umstellen der Gleichungen und Multiplizieren mit dem inversen Element, für Elemente aus B, kann der diskrete Logarithmus errechnet werden. Die so erlangten Ergebnisse können genutzt werden, um auch die restlichen diskreten Logarithmen von Elementen aus B, mit Hilfe von Addition und Subtraktion wie in Gleichung (14) gezeigt, zu errechnen. Siehe dazu das Beispiel aus [15] S. 144. Nun muss solange ein zufälliger Wert für j in Gleichung

$$g^j \cdot x = y\ mod\ p \tag{15}$$

eingesetzt werden, bis ein y, dass das Produkt aus Elementen von B ist, gefunden wurde. Da die diskreten Logarithmen der Elemente aus B bekannt sind, kann mit dessen Hilfe, wie in Gleichung (15) gezeigt, der diskrete Logarithmus von y berechnet werden.

Entscheidend für diesen Algorithmus ist, dass die Faktorbasis Menge B nicht zu klein gewählt wird, da man ansonsten nicht genügend Gleichungen für das Gleichungssystem findet. Die Menge darf aber auch nicht zu groß werden, denn sonst wird das Gleichungssystem auch zu groß und komplex. Entscheidend für die Laufzeit dieses Algorithmus ist das Finden von Gleichungen, bei denen die Primfaktorzerlegung aus Potenzen von Elementen aus B besteht und das zufällige Raten von j, bis das Produkt aus Elementen von B gefunden wurde.

5. FAZIT

Verschlüsselung von Daten ist aus der heutigen vernetzten Welt nicht mehr wegzudenken. Aus diesem Grund sind heutige Verschlüsselungstechniken, die auf Faktorisierung von

Algorithmus	Laufzeit	256 Bit	1024 Bit
Index Calculus	$L(\frac{1}{2};\sqrt{2})$	$6,4 \cdot 10^6$	$1,3 \cdot 10^{30}$
Zahlkörpersieb	$L(\frac{1}{3};3\frac{2}{3})$	2444	$2,8 \cdot 10^{16}$
Pollard Rho	$O(\sqrt{p})$	$5,3 \cdot 10^{26}$	$2 \cdot 10^{142}$

Abbildung 3: Laufzeiten von Algorithmen für DLP [14]

Algorithmus	Laufzeit	80 Bit	160 Bit
Pollard Rho parallel. (n=1000)	$O\left(\sqrt{\frac{\pi p}{2n}}\right)$	0,4	$4,4 \cdot 10^{11}$
Pohlig-Hellman	$O\left(\sum_{i=1}^{r}\left(e_i\left(log\ N+\sqrt{q_i}\right)\right)\right)$	0,01	10,3
Frey-Rück	$L(\frac{1}{2};\sqrt{2})$	0,004	14

Abbildung 4: Laufzeiten von Algorithmen für ECDLP [14]

Primzahlen oder dem diskreten Logarithmus-Problem beruhen, zum Standard geworden. Diese Ausarbeitung hat in Kapitel 2 Grundlagen-Wissen u.a. über algebraische Strukturen, den Euklidischen Algorithmus, sowie zur Euler'schen φ-Funktion vermittelt. Weiter wurde auch ein kleiner Teilbereich aus den elliptischen Kurven vorgestellt, um die nötigen Grundlagen für das ECC zu schaffen. In Kapitel 3 wurden die Primzahlen genauer analysiert. Dabei ist die Primfaktorzerlegung essenziell für die Zahlentheorie und ermöglichen erst kryptographische Algorithmen, die auf eine ineffiziente Berechnung der Primfaktoren aufbauen.

Das grundlegende Problem vom diskreten Logarithmus wurde in Kapitel 4 ausführlich erläutert. Des weiteren wurden Algorithmen vorgestellt, mit dessen Hilfe es möglich ist, den diskreten Logarithmus zu berechnen. Besonders der Index-Calculus-Algorithmus ist den allgemein verwendbaren Algorithmen deutlich überlegen, denn dieser hat eine Laufzeit von etwa $O(exp(\sqrt{2 \cdot ln\ p \cdot ln\ ln\ p}))$. Eine genaue Beschreibung dieses Algorithmus ist dem Buch [3] zu entnehmen. Dieser ist auch dafür mitverantwortlich, weswegen elliptische Kurven gegenüber den multiplikativen Gruppen der endlichem Körper für kryptographische Anwendungsfälle „sicherer" sind. Der Baby-Step-Giant-Step-Algorithmus kann sowohl für das DLP sowie für das ECDLP eingesetzt werden, besitzt allerdings nur eine Laufzeit von $O(\sqrt{p-1})$. Der Index-Calculus-Algorithmus hingegen lässt sich nicht auf elliptischen Kurven umschreiben. Hier bleiben nur die wesentlich langsameren allgemeinen Verfahren.[17]

Bezüglich der Sicherheit von heutigen kryptographischen Verfahren muss man sich aktuell keine Sorgen machen, zumindest wenn die Mindestanforderungen bezüglich der Größe des Körpers und der Schlüssellänge eingehalten werden. Bei zu kleinen Schlüssellängen ist auch bei der besten Verschlüsselung heutzutage keine Sicherheit gegeben. Dies zeigen auch noch einmal die Abbildungen 3 und 4, mit den entsprechenden hochgerechneten Laufzeiten in Jahren, für Algorithmen zum Lösen des DLP und des ECDLP.

Der Höhepunkt im Bereich der Verschlüsselung kann noch nicht erreicht sein. Noch schnellere und effizientere und somit „sicherere" Verschlüsselungstechniken werden benötigt, um auch zukünftigen Anforderungen gerecht zu werden. Denn laut Peter W. Shor existieren bereits die nötigen Algorithmen, um sämtliche heutigen Verschlüsselungen zu brechen. Diese stellt er in [9] vor. Diskrete Logarithmen können in polynomialer Zeit berechnet werden, es muss nur noch gelingen einen genügend großen Quantencomputer zu bauen.

6. REFERENCES

[1] M. Agrawal, N. Kayal, and N. Saxena. Primes is in p. Technical report, Kanpur-208016, INDIA, 2002.

[2] A. Beutelspacher, H. B. Neumann, and T. Schwarzpaul. *Kryptografie in Theorie und Praxis*. Vieweg + Teubner and GWV Fachverlage GmbH, Wiesbaden, 2010.

[3] J. Buchmann. *Einführung in die Kryptographie*. Springer, Wiesbaden, 2006.

[4] O. Deiser and C. Lasser. *Erste Hilfe in Linearer Algebra*. Springer Spektrum, Berlin Heidelberg, 2015.

[5] O. Forster. *Algorithmische Zahlentheorie*. Springer Fachmedien, Wiesbaden, 2015.

[6] M. Hufschmid. *Information und Kommunikation*. Teubner Verlag, Wiesbaden, 2006.

[7] Mathe-Lexikon.at. *Mathe-Lexikon: Sieb-Des-Eratosthenes*, http://www.mathe-lexikon.at/arithmetik/natuerliche-zahlen/teilbarkeit/primzahlen/sieb-des-eratosthenes.html, 22.12.2015.

[8] R. Schulze-Pillot. *Einführung in Algebra und Zahlentheorie*. Springer, Berlin Heidelberg, 2015.

[9] P. W. Shor. *Algorithms for Quantum Computation: Discrete Logarithms and Factoring*. AT and T Bell Labs, 600 Mountain Ave. Murray Hill, NJ 07974, USA, 1994.

[10] S. Spitz, M. Pramateftakis, and J. Swoboda. *Kryptographie und IT-Sicherheit*. Vieweg + Teubner Verlag and Springer Fachmedien, Wiesbaden, 2011.

[11] P. D. F. Sprengel. Kryptographie und algorithmen. Technical report, Hochschule Hannover, 2012.

[12] A. Steger. *Diskrete Strukturen Band 1*. Springer, Berlin Heidelberg, 2007.

[13] G. Teschl and S. Teschl. *Mathematik für Informatiker Band 1*. Springer, Berlin Heidelberg New York, 2007.

[14] D. Wallerstorfer. *DLP/ECDLP Probleme und Lösungen*. Fachhochschul für Computer- und Mediensicherheit in Hagenberg, 2006.

[15] L. C. Washington. *Elliptic Curves - Number Theory and Cryptography*. Taylor Francis Group, LLC, University of Maryland, 2008.

[16] K.-U. Witt. *Algebraische und zahlentheoretische Grundlagen für die Informatik*. Springer Fachmedien, Wiesbaden, 2014.

[17] M. Wohlgemuth. *Mathematisch für fortgeschrittene Anfänger*. Springer Spektrum, Wiesbaden, 2010.

[18] J. Ziegenbalg. *Elementare Zahlentheorie*. Springer Spektrum, Wiesbaden, 2015.