h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

Analyse eines neuen digitalen Signaturverfahrens basierend auf Twisted-Edwards-Kurven

Abschlussarbeit zur Erlangung des akademischen Grades
Master of Science (M.Sc.)

vorgelegt von

Johanna Jung

Referent:	Prof. Dr. Marian Margraf
Korreferent:	Prof. Dr. Michael Braun
Ausgabedatum:	1. Mai 2016
Abgabedatum:	31. Oktober 2016

Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht. Die Zeichnungen oder Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen. Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

Darmstadt, den 31. Oktober 2016

Johanna Jung

Danksagung

An dieser Stelle möchte ich mich bei all denjenigen bedanken, die mich während der Anfertigung dieser Arbeit unterstützt haben.
Zunächst danke ich Prof. Dr. Marian Margraf für die Betreuung meiner Masterarbeit, für die konstruktive Kritik, hilfreiche Diskussionen und seine Geduld.
Ein besonderer Dank gilt auch Christian Wieschebrink (BSI) für die zahlreichen Anmerkungen und Anregungen zu meiner Arbeit.
Auch Prof. Dr. Michael Braun danke ich für seine hilfreichen Ratschläge.
Außerdem bedanke ich mich bei meinem Freund Florian, meinen Eltern und meiner Schwester, für die emotionale Unterstützung während meines gesamten Studiums.

Zusammenfassung

Diese Arbeit beschäftigt sich mit der Sicherheit digitaler Signaturverfahren der Elliptische-Kurven-Kryptographie. Das erste betrachtete Signaturverfahren ist der bereits 1998 standardisierte Elliptic Curve Digital Signature Algorithm (ECDSA).
Nach der Entdeckung von Twisted-Edwards-Kurven konzipierten Bernstein et al. den Edwards-curve Digital Signature Algorithm (EdDSA), ein Signaturverfahren auf Grundlage dieser Kurven. Er wurde 2012 veröffentlicht. Neben der Effizienz bietet das Verfahren auch Vorteile bezüglich der Sicherheit.
Ein Vergleich dieser Signaturverfahren zeigte, dass der EdDSA niedrigere Anforderungen an die verwendete Hashfunktion stellt als der ECDSA. Zudem erfordert der Signaturalgorithmus des EdDSA keinen Zufallszahlengenerator. Außerdem ist die Addition auf Twisted-Edwards-Kurven resistenter gegenüber Seitenkanalangriffen als die Addition auf Weierstraß-Kurven, die beim ECDSA eingesetzt werden. Auch die Skalarmultiplikation des EdDSA enthält eine Reihe von Maßnahmen gegen Seitenkanalangriffe.
Des Weiteren konnte die Sicherheit des EdDSA unter der Annahme, dass das Diskreter-Logarithmus-Problem für elliptische Kurven (ECDLP) schwer ist, im Random-Oracle-Modell bewiesen werden.

Abstract

This thesis deals with the security of digital signature schemes in elliptic curve cryptography. The first signature scheme considered is the Elliptic Curve Digital Signature Algorithm (ECDSA) which was first standardized in 1998. It uses the arithmetic of Weierstraß curves.
After the discovery of twisted Edwards curves Bernstein et al. designed the Edwards-curve Digital Signature Algorithm (EdDSA), a signature scheme based on these curves. It was published in 2012. EdDSA not only offers advantages in terms of efficiency but also in terms of security.
A comparison of both signature schemes showed that EdDSA has lower requirements concering hash functions. Moreover, signing in EdDSA does not require random numbers. EdDSA benefits from the addition law defined over twisted Edwards curves as it is more resistent to side-channel attacks than the Weierstraß addition law. Furthermore, the scalar mulitiplication for EdDSA includes several countermeasures to prevent side-channel attacks.
In addition, this thesis provides a formal proof for the security of EdDSA. It shows that EdDSA is secure in the random oracle model assuming that the elliptic curve discrete logarithim is hard.

Inhaltsverzeichnis

1 Einleitung **17**

2 Mathematische Grundlagen **21**

2.1 Gruppen 21

2.2 Zyklische Gruppen 22

2.3 Untergruppen 23

2.4 Ringe 24

2.5 Körper 25

2.6 Diskreter-Logrithmus-Problem 28

3 Kryptographische Grundlagen **29**

3.1 Schutzziele 29

3.2 Kryptographische Verfahren 30

4 Digitale Signaturverfahren **33**

4.1 Definition 33

4.2 Sicherheitsbegriff für digitale Signaturen 35

4.3 DSA 36

4.4 Schnorr-Signaturverfahren 39

5 Elliptische Kurven **41**

5.1 Weierstraß-Kurven 41

5.1.1 Vereinfachte Weierstraßgleichung 43

5.1.2 Addition 45

5.1.3 Projektive Koordinaten 47

5.1.4 Skalarmultiplikation 48
5.2 Montgomery-Kurven 50
5.2.1 Addition 51
5.2.2 Montgomery-Leiter 52
5.2.3 Curve25519 53
5.3 Edwards-Kurven 54
5.4 Twisted-Edwards-Kurven 55
5.5 Transformationen zwischen elliptischen Kurven 57
5.5.1 Weierstraß-Kurven und Montgomery-Kurven 57
5.5.2 Montgomery-Kurven und Twisted-Edwards-Kurven . . 58
5.5.3 Edwards-Kurven und Twisted-Edwards-Kurven 60
5.6 Diskreter-Logarithmus-Problem für Elliptische Kurven 60
5.6.1 Bekannte Angriffe 61
5.6.2 Seitenkanalangriffe 62

6 ECDSA **67**
6.1 Das Signaturverfahren 67
6.1.1 Systemparameter 67
6.1.2 Schlüsselpaare 69
6.1.3 Signatur 70
6.1.4 Verifikation 70
6.2 Sicherheitsanalyse 72

7 EdDSA **75**
7.1 Das Signaturverfahren 75
7.1.1 Systemparameter 75
7.1.2 Schlüsselpaare 77
7.1.3 Signatur 77
7.1.4 Verifikation 78
7.2 Anmerkungen zum Design 79
7.3 Ed25519 82
7.4 Sicherheitsanalyse 85

8 Gegenüberstellung von ECDSA und EdDSA **93**

9 Fazit und Ausblick **97**

Literaturverzeichnis **99**

Abkürzungsverzeichnis

ANSI American National Standards Institute

BSI Bundesamt für Sicherheit in der Informationstechnik

DLP Diskreter-Logarithmus-Problem

DSA Digital Signature Algorithm

DSS Digital Signature Standard

ECC Elliptische-Kurven-Kryptographie

ECDLP Diskreter-Logarithmus-Problem für elliptische Kurven

ECDSA Elliptic Curve Digital Signature Algorithm

EdDSA Edwards-curve Digital Signature Algorithm

EUF-CMA *existential unforgeable under chosen message attack*

FIPS Federal Information Processing Standard

IEEE Institute of Electrical and Electronics Engineers

ISO International Standards Organization

NIST National Institute of Standards and Technology

Kapitel 1

Einleitung

Digitale Signaturen ermöglichen die Sicherung von Integrität, Nachrichtenauthentizität und Verbindlichkeit. Damit sind sie ein wesentlicher Bestandteil für sichere digitale Kommunikation.
Verfahren zur Erstellung digitaler Signaturen basieren auf privaten und öffentlichen Schlüsseln. Der Verfasser einer Nachricht kann diese mit seinem privaten Schlüssel signieren. Ein Empfänger dieser Nachricht kann mittels des öffentlichen Schlüssels des Verfassers den Autor und die Unverändertheit der Nachricht verifizieren.
Die Sicherheit der heutzutage eingesetzten digitalen Signaturverfahren, z.B. RSA und DSA, basiert auf dem Diskreter-Logarithmus-Problem (DLP) oder dem Faktorisierungsproblem. Bereits 2010 wurde ein RSA Modulus der Länge 768 Bit faktorisiert [KAF+10]. Adrian et al. lösten das DLP im letzten Jahr für Gruppen, deren Ordnung eine 512 Bit große Primzahl ist [AVV+15]. Sowohl für RSA als auch für DSA werden daher Schlüssellängen von 2000 Bit empfohlen [BSI16].

Seit 1998 spielen außerdem digitale Signaturverfahren auf Grundlage des Diskreter-Logarithmus-Problem für elliptische Kurven (ECDLP) eine signifikante Rolle [ISO98]. Sie ermöglichen wesentlich kürzere Schlüssellängen bei gleichem Sicherheitsniveau im Vergleich zu den oben genannten Verfahren [PP10]. Dadurch bieten sie zudem schnellere Laufzeiten als vergleichbare Verfahren. Sie

sind daher besonders für den Einsatz in Umgebungen mit wenig verfügbarem Speicherplatz oder geringer Bandbreite geeignet [HVM03].
Zu den digitalen Signaturverfahren auf Basis von elliptischen Kurven, empfohlen vom Bundesamt für Sicherheit in der Informationstechnik (BSI), gehören der ECDSA, EC-KCDSA und ECGDSA [BSI16]. Tabelle 1.1 stellt die erforderlichen Schlüssellängen von RSA, Digital Signature Algorithm (DSA) und ECDSA für ein Sicherheitsniveau von 100 Bit gegenüber.

Signaturverfahren	RSA	DSA	ECDSA
Schlüssellänge	1900	1900	200

Tabelle 1.1: Erforderliche Schlüssellängen für ein Sicherheitsniveau von mindestens 100 Bit [BSI16]

2012 wurde in [BDL+12] von Bernstein et al. der EdDSA vorgeschlagen. Dieses Signaturverfahren verwendet elliptische Kurven aus der Familie der Twisted-Edwards-Kurven. Es verspricht schnelle Laufzeiten und wenig Angriffsmöglichkeiten für Seitenkanalangriffe [BDL+12].
In dieser Arbeit werden wir die Sicherheit des ECDSA und des EdDSA analysieren und einander gegenüberstellen.

Aufbau der Arbeit

In Kapitel 2 werden zunächst einige mathematische Grundlagen der Public-Key-Kryptographie vorgestellt. Kryptographische Grundbegriffe und Verfahren werden in Kapitel 3 eingeführt. Kapitel 4 beschreibt die Bestandteile und den Sicherheitsbegriff digitaler Signaturverfahren. Zudem werden der DSA und das Schnorr-Signaturverfahren erläutert.
Kapitel 5 bietet eine Einführung in elliptische Kurven und deren Arithmetik. Die darin beschriebenen Weierstraß- und Twisted-Edwards-Kurven sind die Grundlage der in den folgenden Kapiteln behandelten Signaturverfahren.
Daraufhin wird in Kapitel 6 der ECDSA beschrieben. Außerdem wird die Sicherheit des Verfahrens untersucht.
Kapitel 7 beschäftigt sich mit dem EdDSA. Hierbei werden die Unterschiede zum Vorbild des EdDSA, dem Schnorr-Signaturverfahren, erläutert. Darauf

folgt eine Sicherheitsanalyse und ein Sicherheitsbeweis für den EdDSA.
In Kapitel 8 werden ECDSA und EdDSA im Hinblick auf die Sicherheit der Verfahren verglichen. Die Arbeit schließt mit einer Zusammenfassung der Erkenntnisse sowie einem Ausblick auf weitere Forschungsthemen, die den EdDSA betreffen, in Kapitel 9.

Kapitel 2

Mathematische Grundlagen

Dieser Abschnitt führt in die mathematischen Grundlagen der Public-Key-Kryptographie (siehe Kapitel 3.2) ein. Dabei werden die Eigenschaften mathematischer Gruppen, Ringe und Körper vorgestellt.
Die Ausführungen dieses Abschnitts basieren auf gängigen Lehrbüchern der Kryptographie ([Sti06], [MvOV96], [Buc16], [PP10]) und der Elliptische-Kurven-Kryptographie ([CF06], [HVM03], [Wer02]), sowie auf einer Technischen Richtlinie des BSI [BSI12]. Es wird weitgehend auf Beweise verzichtet. Diese sind den angegebenen Quellen zu entnehmen.

2.1 Gruppen

Die Menge G zusammen mit einer zweistelligen Verknüpfung $\circ : G \times G \to G$ bildet genau dann eine **Gruppe** $(G, \circ)$, wenn die folgenden Eigenschaften erfüllt sind:

Assoziativität Für alle $a, b, c \in G$ gilt $a \circ (b \circ c) = (a \circ b) \circ c$.

Neutrales Element Es existiert ein Element $e \in G$, sodass für alle $a \in G$ gilt $e \circ a = a \circ e = a$.

Inverses Element Für jedes Element $a \in G$ existiert ein Element $b \in G$, sodass gilt $a \circ b = b \circ a = e$.

Eine Gruppe G heißt **abelsche Gruppe** oder kommutative Gruppe, wenn außerdem für alle $a, b \in G$ gilt $a \circ b = b \circ a$.
In Abhängigkeit von der verwendeten Verknüpfung sind verschiedene Notationen für Gruppen gebräuchlich.
Im Falle einer **additiven Gruppe** $(G, +)$ heißt das neutrale Element 0. Für ein Element $a \in G$ wird das inverse Element mit $-a$ bezeichnet. Weiter ist $ka = \Sigma_1^k\, a, k \in \mathbb{N}$, die k-fache Addition des Elements a.
Für eine **multiplikative Gruppe** $(G, \cdot)$ ist das neutrale Element die 1. Das Inverse zu $a \in G$ wird bezeichnet mit a^{-1}. $a^k = \Pi_1^k\, a, k \in \mathbb{N}$, ist die k-fache Multiplikation von a.

Ein Beispiel für eine Gruppe ist die abelsche Gruppe $(\mathbb{Z}, +)$ mit der 0 als neutrales Element der Addition. $(\mathbb{Z}, \cdot)$ hingegen ist keine Gruppe, weil nicht jedes Element ein Inverses besitzt.

2.2 Zyklische Gruppen

Wenn G eine endliche Menge mit n Elementen ist und $(G, \circ)$ eine Gruppe ist, so heißt $(G, \circ)$ **endliche Gruppe**. Die Anzahl n der Elemente in G ist dann die **Gruppenordnung** oder Kardinalität der Gruppe und wird mit $|G|$ bezeichnet.

Betrachten wir die Menge $\mathbb{Z}_n = \{0, 1, ..., n-1\}$ mit $n \in \mathbb{N}$. Sie bildet zusammen mit der modularen Addition eine endliche Gruppe $(\mathbb{Z}_n, +)$ der Ordnung n. Die Addition in $\mathbb{Z}_n$ ist definiert als $a + b := a + b \mod n$ für $a, b \in \mathbb{Z}_n$.

In einer endlichen multiplikativen Gruppe G existiert für jedes Element $g \in G$ eine ganze Zahl e mit $1 \leq e \leq n$, sodass gilt $g^e = 1$. Die kleinste positive Zahl, die dies erfüllt, ist die **Ordnung** $ord(g)$ von g in G.

Theorem 1. *Sei $g \in G$ und $e \in \mathbb{Z}$. Dann gilt $g^e = 1$ genau dann, wenn e durch die Ordnung von g teilbar ist.*

Beweis. Sei $n = ord(g)$. Wenn e ein Vielfaches von n ist, folgt $g^e = g^{kn} =$

$(g^n)^k = 1^k = 1$ für ein $k \in \mathbb{Z}$.
Sei umgekehrt $g^e = 1$. Wir können e schreiben als $e = qn + r$ mit $q \in \mathbb{Z}$ und $0 \leq r \leq n-1$. Da $g^e = g^{qn+r} = g^{qn}g^r = 1$ und $g^{qn} = (g^n)^q = 1$, muss gelten $g^r = 1$. Falls $r > 0$ widerspricht dies der Minimalität von n. Also ist $r = 0$ und damit $e = qn$. □

Eine endliche Gruppe $(G, \cdot)$ der Ordnung n heißt **zyklisch**, wenn ein $g \in G$ existiert mit der Eigenschaft

$$G = \{g^1, g^2, g^3, ..., g^n\}.$$

In diesem Fall wird g als **Erzeuger** von G bezeichnet.

2.3 Untergruppen

Sei $(G, +)$ eine Gruppe. Eine nichtleere Untermenge $H \subseteq G$, heißt **Untergruppe**, wenn [CF06]

- $0 \in H$,
- für beliebige Elemente $a, b \in H$ gilt, dass auch $a + b \in H$,
- für jedes $a \in H$ auch $-a \in H$ ist.

Theorem 2 (Satz von Lagrange)**.** *Sei G eine endliche Gruppe und H eine Untergruppe von G. Dann teilt die Ordnung von H die Ordnung von G.*

Beweis. Siehe [Buc16, S. 49]. □

Theorem 3. *Sei G eine endliche Gruppe. Die Ordnung eines Gruppenelementes $g \in G$ teilt die Gruppenordnung $|G|$.*

Beweis. Jedes Element $g \in G$ erzeugt eine Untergruppe $\langle g \rangle = \{g^k \mid 1 \leq k \leq ord(g)\}$ von G. Die Ordnung von g ist die Ordnung der von g erzeugten Untergruppe [Buc16, S. 48]. Also folgt die Behauptung aus Theorem 2. □

Aus den Theoremen 3 und 1 folgt der Satz:

Theorem 4. *Sei G eine endliche Gruppe. Es gilt $g^{|G|} = 1$ für jedes $g \in G$.*

Sei $(G, \cdot)$ eine endliche zyklische Gruppe. Für jedes $g \in G$ bildet die Menge

$$\langle g \rangle = \{g^k \mid 1 \leq k \leq ord(g)\}$$

eine zyklische Untergruppe von G. Wenn n die Gruppenordnung von G ist, so enthält G für jeden Teiler d von n, genau eine zyklische Untergruppe der Ordnung d [CF06].

2.4 Ringe

Sei R eine Menge. Seien $+ : R \times R \to R$ und $\cdot : R \times R \to R$ zweistellige Verknüpfungen auf R. $(R, +, \cdot)$ heißt **Ring**, wenn

- $(R, +)$ eine abelsche Gruppe ist,
- die Verknüpfung $\cdot$ assoziativ ist und
- die Distributivgesetze $a \cdot (b + c) = a \cdot b + a \cdot c$ und $(a + b) \cdot c = a \cdot c + b \cdot c$ gelten für alle $a, b, c \in R$.

Gilt für die Verknüpfung $\cdot$ auch das Kommutativgesetz, so ist $(R, +, \cdot)$ ein **kommutativer Ring**. Existiert zudem ein neutrales Element bzgl. $\cdot$, so sagt man $(R, +, \cdot)$ ist ein **kommutativer Ring mit Einselement**.

Betrachten wir als Beispiel die Menge $\mathbb{Z}_n$ $(n \in \mathbb{N})$ mit den Verknüpfungen $+$ und $\cdot$. Wie bereits erwähnt ist $(\mathbb{Z}_n, +)$ eine abelsche Gruppe. Die Multiplikation in $\mathbb{Z}_n$ ist definiert als $a \cdot b := a \cdot b \mod n$ für alle $a, b \in \mathbb{Z}_n$. Die Verknüpfung $\cdot$ ist sowohl assoziativ, als auch kommutativ. Das neutrale Element ist die 1. Außerdem gelten die Distributivgesetze. Folglich ist $(\mathbb{Z}_n, +, \cdot)$ ein kommutativer Ring mit Einselement. Man bezeichnet $(\mathbb{Z}_n, +, \cdot)$ auch als **Restklassenring** modulo n [Buc16].

Ist R ein Ring, so wird mit R^* die Menge aller multiplikativ invertierbaren Elemente in R bezeichnet. Ein Element $a \in R$ heißt invertierbar, falls ein

Element $b \in R$ existiert, sodass gilt $a \cdot b = 1$. Für den Ring $\mathbb{Z}_n$ ist

$$\mathbb{Z}_n^* := \{a \in \mathbb{Z}_n \mid \exists b \in \mathbb{Z}_n : a \cdot b = 1\}$$

die Menge der multiplikativ invertierbaren Elemente. Jedes von 0 verschiedene Element $a \in \mathbb{Z}_n$ ist genau dann invertierbar, wenn gilt $\gcd(a, n) = 1$ (in diesem Fall heißen a und n teilerfremd). Dabei ist $\gcd(a, n)$ der größte gemeinsame Teiler von a und n. $\mathbb{Z}_n^*$ lässt sich daher auch schreiben als $\mathbb{Z}_n^* := \{a \in \mathbb{Z}_n \setminus \{0\} \mid \gcd(a, n) = 1\}$. Es lässt sich zeigen, dass $\mathbb{Z}_n^*$ eine multiplikative Gruppe ist. Man bezeichnet sie als **prime Restklassengruppe** modulo n.

Um zu einem Element $a \in \mathbb{Z}_n^*$ das multiplikative Inverse $a^{-1} \in \mathbb{Z}_n^*$ zu bestimmen, kann der **Erweiterte Euklidische Algorithmus** (siehe [HVM03, S. 39ff]) verwendet werden.

Die Ordnung der Gruppe $(\mathbb{Z}_n^*, \cdot)$ lässt sich mittels der **Eulerschen Φ-Funktion** bestimmen. Die Funktion $\Phi : \mathbb{N} \to \mathbb{N}$ ist definiert durch

$$\Phi(n) := |\{a \in \mathbb{Z}_n \setminus \{0\} \mid \gcd(a, n) = 1\}|.$$

Für den Fall, dass $n = p$ eine Primzahl ist, ist $\Phi(n) = \Phi(p) = p - 1$, da kein Element in $\mathbb{Z}_p \setminus \{0\}$ einen gemeinsamen Teiler größer 1 mit p hat.

Theorem 5 (Satz von Euler)**.** *Seien n und a zwei teilerfremde ganze Zahlen. Dann gilt*

$$a^{\Phi(n)} = 1 \mod n.$$

Beweis. Da $\Phi(n)$ die Ordnung der endlichen Gruppe $\mathbb{Z}_n^*$ bestimmt, folgt der Satz aus Theorem 4. □

2.5 Körper

Ein **Körper** $(\mathbb{F}, +, \cdot)$ ist eine Menge $\mathbb{F}$ zusammen mit zwei Verknüpfungen $+ : \mathbb{F} \times \mathbb{F} \to \mathbb{F}$ und $\cdot : \mathbb{F} \times \mathbb{F} \to \mathbb{F}$, für die gilt [BSI12]:

- $(\mathbb{F}, +)$ ist eine abelsche Gruppe,
- $(\mathbb{F} \setminus \{0\}, \cdot)$ ist eine abelsche Gruppe,
- das Distributivgesetz $(a + b) \cdot c = a \cdot c + b \cdot c$ gilt für alle $a, b, c \in \mathbb{F}$.

Ein **endlicher Körper** ist ein Körper, der nur endlich viele Elemente enthält. Die **Charakteristik** $char(\mathbb{F})$ eines Körpers $\mathbb{F}$ ist die kleinste positive Zahl m, sodass gilt $m1 = \underbrace{1 + 1 + ... + 1}_{m\text{-mal}} = 0$. Falls für alle natürlichen Zahlen m das Element $m1$ von Null verschieden ist, so hat $\mathbb{F}$ die Charakteristik 0. Dies ist beispielsweise für die Körper der rationalen Zahlen $\mathbb{Q}$, der reelen Zahlen $\mathbb{R}$ und der komplexen Zahlen $\mathbb{C}$ der Fall.
Man kann zeigen, dass die Charakteristik eines Körpers entweder 0 oder eine Primzahl p ist. Insbesondere gilt für jeden endlichen Körper, dass seine Charakteristik eine Primzahl ist [MvOV96, S. 95].

Restklassenringe $\mathbb{Z}_p$ modulo einer Primzahl p finden häufig Verwendung in der Kryptographie. Da jedes von Null verschiedene Element multiplikativ invertierbar ist, bildet $\mathbb{Z}_p$ einen Körper. An dieser Stelle ist anzumerken, dass die multiplikative Gruppe $(\mathbb{F} \setminus \{0\}, \cdot)$ eines endlichen Körpers $\mathbb{F}$ eine zyklische Gruppe ist [CF06, S. 32]. Da $\mathbb{Z}_p$ einen Körper bildet, ist also $(\mathbb{Z}_p^*, \cdot)$ eine zyklische Gruppe.

In der Elliptische-Kurven-Kryptographie (ECC) sind besonders zwei Arten von endlichen Körpern von Bedeutung: Primkörper und Erweiterungskörper (mit Charakteristik 2).
Ein **Primkörper** $\mathbb{F}_p$ ist ein endlicher Körper mit p Elementen, wobei p eine Primzahl ist. Die Charakteristik $char(\mathbb{F}_p)$ ist p.
Erweiterungskörper sind endliche Körper der Ordnung p^m, p prim, $m \in \mathbb{N}$. Ist $p = 2$, so spricht man von binären Erweiterungskörpern. Ein Erweiterungskörper $\mathbb{F}_{p^m}$ lässt sich darstellen als die Menge von Polynomen, deren Koeffizienten aus dem Körper $\mathbb{F}_p$ sind und deren Grad maximal $m - 1$ ist:

$$\mathbb{F}_{p^m} = \{a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + ... + a_2x^2 + a_1x^1 + a_0 \mid a_i \in \mathbb{F}_p, 0 \leq i \leq m-1\}.$$

Dazu wählt man ein normiertes irreduzibles Polynom $f(x)$ mit Grad m. Ein Polynom $f(x)$ heißt irreduzibel, wenn es nicht in ein Produkt zweier Polynome vom Grad kleiner m zerlegbar ist. Die Addition der Körperelemente ist eine Polynomaddition, bei der die Koeffizienten modulo p addiert werden. Multiplikation der Körperelemente basiert auf der Multiplikation der Polynome modulo des Reduktionspolynoms $f(x)$. Für jedes Polynom $a(x) \in \mathbb{F}_{p^m}$ exisitert ein eindeutiges Restpolynom $r(x) := a(x) \mod f(x)$ mit Grad kleiner m, das man durch die Division von $a(x)$ durch $f(x)$ erhält. Diese Operation wird als Polynomreduktion modulo $f(x)$ bezeichnet.

Sei $\mathbb{F}$ ein Körper. Als $\mathbb{F}[x]$ bezeichnen wir die Menge aller Polynome über $\mathbb{F}$ in x, d.h. $\mathbb{F}[x]$ enthält alle Polynome

$$f(x) = \Sigma_{k \geq 0}\ a_k x^k$$

für $a_k \in \mathbb{F}$. Falls $f \neq 0$, so heißt der höchste Exponent k mit $a_k \neq 0$ der Grad von f, $deg(f)$.
Ein Körper $\mathbb{F}$ heißt **algebraisch abgeschlossen**, wenn sich jedes Polynom $f(x) \in \mathbb{F}[x]$ von positivem Grad als Produkt von Polynomen vom Grad 1 schreiben lässt, d.h. wenn

$$f(x) = d(x - c_1)...(x - c_m)$$

für Elemente c_i, d aus $\mathbb{F}$ gilt. In diesem Fall ist $m = deg(f)$ und d der Koeffizient vor x^m [Wer02].

Man kann jeden Körper $\mathbb{F}$ in einen algebraisch abgeschlossenen Körper einbetten. Es gibt einen kleinsten algebraisch abgeschlossenen Erweiterungskörper von $\mathbb{F}$. Dieser heißt **algebraischer Abschluss** von $\mathbb{F}$ und wird mit $\overline{\mathbb{F}}$ bezeichnet.
Ein Beispiel für einen algebraisch abgeschlossenen Körper ist der Körper der komplexen Zahlen $\mathbb{C}$. $\mathbb{C}$ ist der algebraische Abschluss von $\mathbb{R}$. Das Polynom $f(x) = x^2 + 1$ lässt sich in $\mathbb{R}[x]$ nicht als Produkt von Polynomen vom Grad

1 schreiben. In $\mathbb{C}[x]$ hingegen gilt $f(x) = x^2 + 1 = (x - i)(x + i)$.
Der algebraische Abschluss $\overline{\mathbb{F}}_p$ eines Primkörpers $\mathbb{F}_p$ enthält für alle $r \geq 1$ einen endlichen Körper $\mathbb{F}_{p^r}$ mit p^r Elementen [Wer02].

2.6 Diskreter-Logrithmus-Problem

Sei G eine endliche zyklische Gruppe mit einem Erzeuger g und der Ordunung $ord(g) = n$. Sei h ein beliebiges Element aus G. Das **Diskreter-Logarithmus-Problem** ist das Problem eine Zahl $1 \leq x \leq n$ zu finden, sodass

$$g^x = h.$$

x wird bezeichnet als der diskrete Logarithmus von h zur Basis g. Man schreibt $x = \log_g h$.

Eine Gruppe G heißt kryptographisch stark, wenn das DLP in G praktisch nicht lösbar ist. Nach aktuellem Stand ist dies der Fall, wenn mindestens 2^{100} Rechenschritte benötigt werden, um den diskreten Logarithmus zu berechnen [BSI16]. Man spricht dann von einem Sicherheitsniveau von 100 Bit.

In der Praxis werden häufig Untergruppen von $\mathbb{Z}_p^*$, p prim, deren Ordnung eine Primzahl ist, verwendet. Die Gruppe $\mathbb{Z}_p^*$ hat bekanntlich die gerade Ordnung $p - 1$. Wählt man ein Element $a \in \mathbb{Z}_p^*$, so kann es passieren, dass a Teil einer kleinen Untergruppe ist, in der das DLP leichter lösbar ist. Um diesen Fall zu verhindern, wählt man eine Untergruppe $G \subset \mathbb{Z}_p^*$ mit primer Ordnung q. Da q keine nichttrivialen Teiler hat, enthält G keine echten Untergruppen, für die das DLP leicht berechenbar wäre.

Kapitel 3

Kryptographische Grundlagen

Als Kryptographie bezeichnet man die Lehre mathematischer Verfahren im Bereich der Informationssicherheit [MvOV96, S. 4].
In diesem Kapitel sollen grundlegende Begriffe und Techniken der Kryptographie eingeführt werden. Dazu werden zunächst die Schutzziele der Informationssicherheit erläutert. Abschnitt 3.2 gibt einen Überblick kryptographischer Maßnahmen, die zur Erreichung der Schutzziele eingesetzt werden.

3.1 Schutzziele

Die wesentlichen Schutzziele der Informationssicherheit sind in [MvOV96, S. 4] zusammengefasst:

Vertraulichkeit Der Inhalt einer Nachricht soll nur für Personen zugänglich sein, die zum Lesen der Nachricht autorisiert sind.

Integrität Integrität betrifft die Manipulation von Daten. Eine unberechtigte Veränderung, Löschung oder das Hinzufügen von Daten soll detektiert werden können.

Nachrichtenauthentizität Der Empfänger einer Nachricht soll deren Sender zweifelsfrei identifizieren können. Dies erfordert die Integrität der Daten.

Teilnehmerauthentizität Ein Teilnehmer soll die Identität seines aktuellen Kommunikationspartners feststellen können.

Verbindlichkeit/Nichtabstreitbarkeit Der Empfänger einer Nachricht kann deren Sender zweifelsfrei identifizieren und die Identität des Senders gegenüber Dritten nachweisen. Dies erfordert die Integrität und Authentizität der Nachricht.

Verfügbarkeit Innerhalb eines festgelegten Zeitrahmens soll der Zugriff auf die Daten ermöglicht werden.

3.2 Kryptographische Verfahren

Mit Ausnahme der Verfügbarkeit lassen sich diese Schutzziele durch Anwendung kryptographischer Maßnahmen erreichen. Kryptographische Verfahren werden in symmetrische und asymmetrische Verfahren unterteilt.

Secret-Key-Kryptographie *Symmetrische Kryptographie* bezeichnet man auch als *Secret-Key-Kryptographie*. Die Teilnehmer erhalten dabei alle gleiche geheime Schlüssel, die zum Ver- und Entschlüsseln oder zum Authentifizieren von Nachrichten verwendet werden.

Public-Key-Kryptographie Bei der *asymmetrischen* oder *Public-Key-Kryptographie* erhalten die Teilnehmer jeweils zwei Paare bestehend aus privatem und öffentlichem Schlüssel. Das eine Paar wird zur Verschlüsselung verwendet, das andere für digitale Signaturen. Der private Schlüssel ist geheim und wird zum Entschlüsseln bzw. zum Signieren von Nachrichten verwendet. Der öffentliche Schlüssel wird von den Kommunikationspartnern zum Verschlüsseln bzw. Verifizieren genutzt.

Abhängig von den zu erfüllenden Schutzziel kommen folgende Verfahren zum Einsatz:

Verschlüsselung Die Verschlüsselung von Daten macht sie für Dritte unlesbar. Nur Inhaber der kryptographischen Schlüssel können sie entschlüsseln und somit lesbar machen. Mit der Verschlüsselung lässt sich also die Vertraulichkeit gewährleisten. Es gibt sowohl symmetrische, als auch asymmetrische Verschlüsselungstechniken. Symmetrische Algorithmen werden vermehrt eingesetzt, da sie im Allgemeinen schneller sind und mit kürzeren Schlüsseln arbeiten, als asymmetrische Verfahren. Um die Vorteile der Secret-Key-Kryptographie nutzen zu können, muss jedoch zunächst ein Schlüsselaustausch stattfinden.

Kryptographische Hashfunktionen Unter kryptographischen Hashfunktionen versteht man Funktionen, die als Eingabe eine Datenmenge erhalten und einen Hash (eine Bitfolge fester Länge) zurückgeben. Es muss praktisch unmöglich sein, verschiedene Datenmengen zu finden, die auf den gleichen Hash abgebildet werden. Zudem handelt es sich bei einer Hashfunktion um eine Einwegfunktion, d.h. mithilfe eines gegebenen Hashs, ist es praktisch unmöglich die ursprünglich eingegebenen Daten zu berechnen. Den Hash einer Datenmenge m bezeichnen wir mit $h(m)$. Mit Hashs lässt sich die Integrität von Daten nachweisen. Vergleicht man bspw. den Hash einer Nachricht vor und nach dem Versand, so kann man feststellen, ob die Nachricht während des Versands verändert wurde.

Message-Authentication-Code (MAC) Ein MAC wird zur Überprüfung der Authentizität von Nachrichten eingesetzt. Dabei wird zu einer Nachricht m, unter Verwendung eines symmetrischen Schlüssels k, eine eindeutige Bitfolge fester Länge erzeugt. Diese Bitfolge wird als MAC bezeichnet. Im Unterschied zu einem Hash, ist der MAC nicht nur von der Nachricht, sondern außerdem vom Schlüssel des Senders abhängig. Wie der Name schon sagt, gewährleistet ein MAC neben der Integriät auch die Authentizität der Nachricht.

Digitale Signaturen Die Signaturen der Public-Key-Kryptographie bezeichnet man als digitale Signaturen. Der Sender einer Nachricht signiert

diese mit seinem privaten Schlüssel. Der Empfänger kann die Signatur mittels des öffentlichen Schlüssels des Senders verifizieren. Da der private Schlüssel nur einem einzigen Teilnehmer bekannt ist, kann so neben der Nachrichtenauthentizität und der Integrität auch die Verbindlichkeit gewährleistet werden.

Challenge-Response-Verfahren Um die Authentizität eines Teilnehmers zu überprüfen, kann man Challenge-Response-Verfahren einsetzen. Diese gibt es sowohl in der Secret-Key- als auch in der Public-Key-Kryptographie. Der entsprechende Teilnehmer authentisiert sich, indem er die Kenntnis des geheimen bzw. privaten Schlüssels nachweist. Man sendet dazu eine *Challenge* an den zu authentifizierenden Teilnehmer. Dieser muss die Challenge entweder signieren, mit einem MAC authentifizieren oder entschlüsseln und zurückschicken. Lässt sich die *Response* erfolgreich verifizieren, so ist damit die Authentizität des Kommunikationspartners sichergestellt.

Diese Arbeit lässt sich in den Bereich der digitalen Signaturen einordnen. Wie die vorausgegangen Erklärungen zeigen, steht dieses Thema auch im Bezug zu anderen Basistechniken der Kryptographie. Zum Beispiel werden kryptographische Hashfunktionen verwendet, um die Länge digitaler Signaturen zu begrenzen. Zudem können digitale Signaturen für ein Challenge-Response-Verfahren eingesetzt werden.

Kapitel 4

Digitale Signaturverfahren

Digitale Signaturen sichern Integrität, Nachrichtenauthentizität und Verbindlichkeit und sind damit ein wichtiger Bestandteil der sicheren digitalen Kommunikation.
In diesem Kapitel behandeln wir die Definition sowie den Sicherheitsbegriff digitaler Signaturen. Danach werden zwei bekannte Signaturverfahren beschrieben.
In [BSI16] wird die Verwendung der Signaturverfahren RSA, DSA, DSA-Varianten auf elliptischen Kurven sowie Merkle-Signaturen empfohlen.
Wir betrachten den DSA sowie Schnorr-Signaturen, die dem DSA in wesentlichen Teilen ähneln. In Kaptiel 6 und 7 werden wir zwei Signaturverfahren auf elliptischen Kurven vorstellen, die nach dem Vorbild von DSA und Schnorr-Signaturen konstruiert wurden. RSA- und Merkle-Signaturen unterscheiden sich wesentlich von den anderen genannten Signaturverfahren und werden daher in dieser Arbeit nicht behandelt. Eine Erläuterung dieser Verfahren ist in [Buc16] und [MvOV96] zu finden.

4.1 Definition

Ein Teilnehmer kann mithilfe eines Signaturalgorithmus unter Verwendung seines privaten Schlüssels eine Nachricht m signieren. Der Empfänger der Nachricht nutzt dann den öffentlichen Schlüssel, um zu verifizieren, ob die

Signatur der Nachricht korrekt ist.
Bevor man jedoch eine digitale Signatur erzeugen kann, müssen die entsprechenden Schlüssel generiert werden und die öffentlichen Schlüssel müssen auf einem zuverlässigen Weg zwischen den Teilnehmern ausgetauscht werden.
Ein Signaturverfahren besteht daher aus mehreren Algorithmen [Buc16]:

1. einem Schlüsselerzeugungsalgorithmus,
2. einem Signaturalgorithmus und
3. einem Verifikationsalgorithmus.

Üblicherweise wird nicht die tatsächliche Nachricht signiert, sondern lediglich ein Hash der Nachricht. Diese Methode ermöglicht es, beliebig lange Nachrichten zu signieren. Dazu verwendet man kryptographische Hashfunktionen $H : \{0,1\}^* \to \{0,1\}^n$, die eine Nachricht m auf einen Hash $H(m)$ mit fester Länge $n \in \mathbb{N}$ abbilden [Buc16].

Formal betrachtet ist ein Signaturverfahren eine Tupel $(\mathcal{P}, \mathcal{A}, \mathcal{K}, \mathcal{S}, \mathcal{V})$ bestehend aus [Sti06]

- $\mathcal{P}$, einer endlichen Menge von Nachrichten,
- $\mathcal{A}$, einer endlichen Menge von Unterschriften,
- $\mathcal{K}$, einer endlichen Menge von Schlüsseln,
- $\mathcal{S}$, einer endlichen Menge von Signaturalgorithmen und
- $\mathcal{V}$, einer endlichen Menge von Verifikationsalgortihmen.

Zu jedem Schlüssel $K \in \mathcal{K}$ existiert ein Signaturalgorithmus $sig_K \in \mathcal{S}$ und ein entsprechender Verifikationsalgorithmus $ver_K \in \mathcal{V}$. Die Funktionen $sig_K : \mathcal{P} \to \mathcal{A}$ und $ver_K : \mathcal{P} \times \mathcal{A} \to \{true, false\}$ erfüllen für jede Nachricht $x \in \mathcal{P}$ und jede Signatur $y \in \mathcal{A}$:

$$ver_K(x,y) = \begin{cases} true & \text{if } y = sig_K(x) \\ false & \text{if } y \neq sig_K(x). \end{cases}$$

sig_K und ver_K müssen effizient berechenbar sein.
Es darf praktisch nicht möglich sein, ohne Kenntnis des privaten Schlüssels für eine Nachricht x eine Signatur y zu berechnen, sodass $ver_K(x, y) = true$. Dazu muss insbesondere gewährleistet sein, dass der private Schlüssel praktisch nicht aus dem öffentlichen Schlüssel konstruiert werden kann [BSI16].

4.2 Sicherheitsbegriff für digitale Signaturen

Um einen Sicherheitsbegriff für digitale Signaturverfahren zu definieren, betrachten wir zunächst verschiedene Angriffsszenarien und Angriffsziele.
Mögliche Angriffsszenarien werden anhand der Ressourcen, die ein Angreifer zur Verfügung hat, unterschieden [GMR88]:

Key-Only-Angriff (KOA) Der Angreifer ist nur im Besitz des öffentlichen Schlüssels.

Known-Message-Angriff (KMA) Der Angreifer kennt eine Menge von Nachrichten und die entsprechenden Signaturen.

Chosen-Message-Angriff (CMA) Um eine Signatur für eine Nachricht m zu fälschen, kann der Angreifer beliebige Nachrichten $m' \neq m$ durch ein Signaturorakel signieren lassen.

Ziele eines Angreifers können sein [GMR88]:

Vollständiges Brechen Der Angreifer kann den geheimen Schlüssel ermitteln. Falls dies nicht möglich ist, so heißt das Signaturverfahren sicher gegen vollständiges Brechen bzw. *unbreakable* (UB).

Universelle Fälschung Der Angreifer kann Signaturen für jede beliebige Nachricht fälschen. Ist dies nicht möglich, so nennen wir das Verfahren sicher gegen universelle Fälschung bzw. *unviersal unforgeable* (UUF).

Existentielle Fälschung Der Angreifer kann für wenigstens eine Nachricht m eine gültige Signatur erzeugen, wobei er nicht notwendigerweise Kontrolle über m hat. Ein Verfahren, das dies nicht zulässt, heißt sicher gegen existentielle Fälschung bzw. *existential unforgeable* (EUF).

Wir bezeichnen ein Signaturverfahren als sicher bzw. EUF-CMA[1]-sicher, wenn es sicher gegen existentielle Fälschung unter einem Chosen-Message-Angriff ist, d.h. wenn ein Angreifer mit dem stärksten Angriff nicht das schwächste Ziel erreichen kann [HVM03, S. 183].

4.3 DSA

1991 wurde der Digital Signature Algorithm (DSA) vom US-amerikanischen National Institute of Standards and Technology (NIST) vorgeschlagen und später als Digital Signature Standard (DSS) in [NIS00] standardisiert. Die Sicherheit des DSA beruht auf dem DLP in Untergruppen von $\mathbb{Z}_p^*$ mit Primzahlordnung.

Schlüsselgenerierung Um die Schlüssel des DSA zu erstellen, benötigen wir zwei zyklische Gruppen. Wir wählen zunächst die endliche zyklische Gruppe $\mathbb{Z}_p^*$ mit p prim. Die Ordnung der Gruppe ist folglich $|\mathbb{Z}_p^*| = \Phi(p) = p - 1$. Im nächsten Schritt erzeugen wir eine zyklische Untergruppe G von $\mathbb{Z}_p^*$ deren Ordnung eine Primzahl q ist. Dazu wählen wir q als Teiler von $p - 1$ und wählen zufällig ein Element x aus $\mathbb{Z}_p^*$. Dann definieren wir den Erzeuger der Untergruppe als $g := x^{(p-1)/q} \mod p$. Da nach dem Satz von Euler (Theorem 5) gilt $x^{\Phi(p)} = x^{p-1} = 1 \mod p$, gilt auch $g^q = (x^{(p-1)/q})^q = x^{p-1} = 1 \mod p$. Damit ist gezeigt, dass $ord(g) \leq q$.
Tatsächlich gilt auch $ord(g) = q$. Da $g^q = 1 \mod p$ muss nach Theorem 1 q ein Vielfaches von $ord(g)$ sein. Da q eine Primzahl ist, gilt also $ord(g) = q$.
Die Untergruppe $\langle g \rangle$ von $\mathbb{Z}_p^*$ ist also eine zyklische Gruppe und hat die Ordnung q. Wir bezeichnen (p, q, g) als Systemparameter. Diese sind öffentlich.

Nach der Bestimmung der Systemparameter, können öffentlicher und privater Schlüssel des DSA generiert werden. Dazu wählen wir ein zufälliges a kleiner q. a ist der private Schlüssel. Der öffentliche Schlüssel A wird dann berechnet, indem der Erzeuger g mit a potenziert wird.

[1] *existential unforgeable under chosen message attack* (EUF-CMA)

Algorithmus 1 : DL-Systemparametergenerierung

Input : Bitlängen l, t

Output : Systemparameter (p, q, g)

1 Wähle Primzahlen q und p mit Bitlängen t und l, sodass gilt $q|(p-1)$.

2 Wähle ein beliebiges $x \in [1, p-1]$ und berechne $g = x^{(p-1)/q} \mod p$.

3 **if** $g = 1$ **then** Gehe zu Schritt 2.

4 **return** (p, q, g)

Algorithmus 2 : DSA-Schlüsselgenerierung

Input : Systemparameter (p, q, g)

Output : Öffentlicher Schlüssel A, privater Schlüssel a

1 Wähle ein zufälliges $a \in [1, q-1]$.

2 Berechne $A = g^a \mod p$.

3 **return** (A, a)

Signaturerzeugung Eine DSA-Signatur besteht aus zwei Teilen, r und s. Zunächst wird eine zufällige Zahl $k \in [1, q-1]$ zur einmaligen Verwendung gewählt. Durch Potenzieren des Erzeugers g mit k erhält man dann r und damit den ersten Teil der Signatur.

Der zweite Teil, s, errechnet sich aus dem multiplikativen Inversen der Zufallszahl k, dem Hash h der zu signierenden Nachricht m und r.

Algorithmus 3 : DSA-Signaturerzeugung

Input : Systemparameter (p, q, g), privater Schlüssel a, Nachricht m

Output : Signatur (r, s)

1 Wähle ein zufälliges $k \in [1, q-1]$.

2 Berechne $r = (g^k \mod p) \mod q$.

3 **if** $r = 0$ **then** Gehe zu Schritt 1.

4 Berechne $h = H(m)$, wobei H eine kryptographische Hashfunktion ist.

5 Berechne $k^{-1} \mod q$.

6 Berechne $s = k^{-1}(h + ar) \mod q$.

7 **if** $s = 0$ **then** Gehe zu Schritt 1.

8 **return** (r, s)

Signaturverifikation Um die DSA-Signatur (r, s) einer Nachricht m zu verifizieren, wird wie folgt vorgegangen. Im ersten Schritt wird überprüft, ob r und s innerhalb des zulässigen Intervalls $[1, q-1]$ liegen. Falls nicht wird die Signatur abgelehnt.
Dann wird das Inverse von s bzgl. q berechnet und mit dem Hash h der Nachricht m bzw. mit r multipliziert. Daraus erhält man zwei Exponenten u_1, u_2. Mittels dieser Exponenten, dem Erzeuger g und dem öffentlichen Schlüssel A wird daraufhin r' berechnet. Die Signatur wird akzeptiert, falls gilt $r = r'$.

Algorithmus 4 : DSA-Signaturverifikation

Input : Systemparameter (p, q, g), öffentlicher Schlüssel A, Nachricht m, Signatur (r, s)

Output : `true`, wenn die Signatur akzeptiert wird, `false`, wenn sie abgelehnt wird

1 **if** $r, s \notin [1, q-1]$ **then return** `false`
2 Berechne $h = H(m)$.
3 Berechne $w = s^{-1} \mod q$.
4 Berechne $u_1 = hw \mod q$ und $u_2 = rw \mod q$.
5 Berechne $r' = (g^{u_1} A^{u_2} \mod p) \mod q$.
6 **if** $r = r'$ **then return** `true`
7 **else return** `false`

Sei (r, s) eine korrekte Signatur. Dann gilt

$$s = k^{-1}(h + ar) \mod q \Leftrightarrow k = s^{-1}(h + ar) \mod q.$$

Wir zeigen nun, dass eine korrekte Signatur durch die Verifikation akzeptiert wird:

$$\begin{aligned} r' &= (g^{u_1} A^{u_2} \mod p) \mod q = (g^{u_1} (g^a)^{u_2} \mod p) \mod q \\ &= (g^{hw} g^{arw} \mod p) \mod q = (g^{hs^{-1} + ars^{-1}} \mod p) \mod q \\ &= (g^{s^{-1}(h+ar)} \mod p) \mod q = (g^k \mod p) \mod q \\ &= r \end{aligned}$$

4.4 Schnorr-Signaturverfahren

Das Signaturverfahren von Schnorr basiert, sowie der DSA, auf dem DLP. Es wurde 1990 in [Sch90] veröffentlicht und zugleich patentiert.

Schlüsselgenerierung Die Generierung eines Schlüsselpaares für eine Schnorr-Signatur gleicht der Bestimmung von Systemparametern und Schlüsseln beim DSA. Man erhält also, wie in Abschnitt 4.3, die Systemparameter (p, q, g), den öffentlichen Schlüssel A und den privaten Schlüssel a.

Signaturerzeugung Um eine Nachricht zu signieren, wird zunächst eine zufällige Zahl k kleiner q gewählt. Im nächsten Schritt wird der Erzeuger g mit k potenziert. Das Ergebnis wird zusammen mit der Nachricht m gehasht. Dieser Hash bildet einen Teil der Signatur. Der zweite Teil ergibt sich aus der Summe von k und dem Produkt aus dem Hash und dem privaten Schlüssel a.

Algorithmus 5 : Schnorr-Signaturerzeugung

Input : Systemparameter (p, q, g), privater Schlüssel a, Nachricht m

Output : Signatur (h, s)

1 Wähle ein zufälliges $k \in [1, q-1]$.
2 Berechne $r = g^k \mod p$,
3 $h = H(m, r)$, wobei H eine kryptographische Hashfunktion ist, und
4 $s = ah + k \mod q$.
5 **return** (h, s)

Signaturverifikation Der Prüfer einer Signatur potenziert g mit dem Teil s der Signatur und das Inverse des öffentlichen Schlüssels mit der Hash h. Das Produkt dieser Potenzen wird dann zusammen mit der Nachricht gehasht. Wenn der neue Hash dem in der Signatur enthalten Hash h entspricht, so wird die Signatur aktzeptiert.

Algorithmus 6 : Schnorr-Signaturverifikation

Input : Systemparameter (p, q, g), öffentlicher Schlüssel A, Nachricht m, Signatur (h, s)

Output : `true`, wenn die Signatur akzeptiert wird, `false`, wenn sie abgelehnt wird

1 Berechne $v = g^s A^{-h} \mod p$ und
2 $h' = H(m, v)$.
3 **if** $h = h'$ **then return** `true`
4 **else return** `false`

Unter der Voraussetzung, dass (h, s) eine korrekt erzeugte Schnorr-Signatur ist, gilt

$$s = ah + k \mod q \Leftrightarrow k = s - ah \mod q.$$

Damit lässt sich zeigen, dass eine korrekte Signatur durch die Verifikation akzeptiert wird:

$$\begin{aligned} h' &= H(m, v) = H(m, g^s A^{-h} \mod p) \\ &= H(m, g^s (g^a)^{-h} \mod p) = H(m, g^{s-ah} \mod p) \\ &= H(m, g^k \mod p) = H(m, r) \\ &= h \end{aligned}$$

Kapitel 5

Elliptische Kurven

1985 schlug sowohl Neal Koblitz [Kob87] als auch Victor Miller [Mil85] die Verwendung von elliptischen Kurven für die Public-Key-Kryptographie vor. Koblitz und Miller vermuteten, dass das Diskreter-Logarithmus-Problem für elliptische Kurven schwerer sei als das klassische Diskreter-Logarithmus-Problem [Kob87] und zudem auch schnellere Laufzeiten ermögliche [Mil85].
Im Folgenden sollen verschiedene Varianten elliptischer Kurven und die auf ihnen basierende Arithmetik vorgestellt werden. Außerdem wird das Diskreter-Logarithmus-Problem für elliptische Kurven formuliert.

5.1 Weierstraß-Kurven

Eine elliptische Kurve definiert über einem Körper besteht aus der Menge aller Punkte, die eine sogenannte Kurvengleichung lösen. Spricht man allgemein von elliptischen Kurven, so entspricht die zugrunde liegende Kurvengleichung der *allgemeinen Weierstraßgleichung.*

Definition 1 (Elliptische Kurve, [HVM03, S. 76])**.** *Eine elliptische Kurve E über einem Körper $\mathbb{F}$ ist definiert durch die Gleichung*

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \tag{5.1}$$

mit $a_1, a_2, a_3, a_4, a_6 \in \mathbb{F}$ und $\Delta \neq 0$. Δ ist die Diskriminante der Kurvenglei-

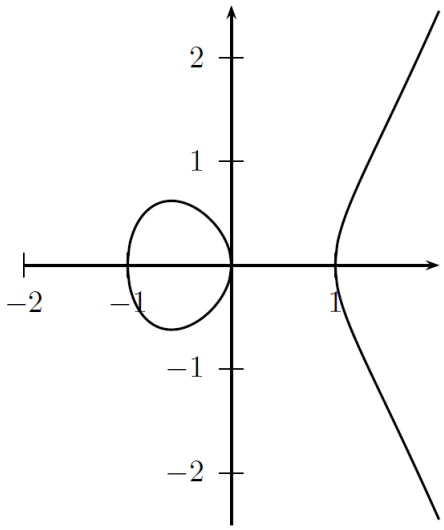


Abbildung 5.1: Weierstraß-Kurve $E : y^2 = x^3 - x$ über $\mathbb{R}$

chung und ist wie folgt definiert:

$$\begin{aligned}
\Delta &= -d_2^2 d_8 - 8d_4^3 - 27d_6^2 + 9d_2 d_4 d_6 \\
d_2 &= a_1^2 + 4a_2 \\
d_4 &= 2a_4 + a_1 a_3 \\
d_6 &= a_3^2 + 4a_6 \\
d_8 &= a_1^2 a_6 + 4a_2 a_6 - a_1 a_3 a_4 + a_2 a_3^2 - a_4^2.
\end{aligned}$$

Die Menge der rationalen Punkte auf E über $\mathbb{F}$ ist

$$E(\mathbb{F}) = \{(x, y) \in \mathbb{F}^2 \mid y^2 + a_1 xy + a_3 y - x^3 - a_2 x^2 - a_4 x - a_6 = 0\} \cup \{\infty\},$$

wobei ∞ der unendlich ferne Punkt ist.

Die Gleichung 5.1 heißt *allgemeine Weierstraßgleichung*. Man spricht daher auch von *Weierstraß-Kurven* oder elliptischen Kurven in Weierstraß-Form. Abbildung 5.1 zeigt eine Weierstraß-Kurve mit Kurvengleichung $y^2 = x^3 - x$ über $\mathbb{R}$.

Aus der Bedingung $\Delta \neq 0$ folgt, dass eine elliptische Kurve nicht-singulär ist.

Der Beweis ist in [Sil09, S. 45f] nachzulesen. Ein Punkt auf einer Kurve wird als *singulär* bezeichnet, wenn beide partiellen Ableitungen der Kurvengleichung in diesem Punkt verschwinden. Eine Kurve $E(\mathbb{F})$ heißt *nicht-singulär* oder *glatt*, wenn jeder Punkt auf $E(\overline{\mathbb{F}})$ nicht-singulär ist. Geometrisch betrachtet, bedeutet dies, dass die Kurve keine Knoten oder Spitzen hat [CF06, S. 64].

Definition 2 (Kurvenordnung, [HVM03, S. 82])**.** *Die Anzahl der Punkte auf einer elliptischen Kurve E definiert über einem Körper $\mathbb{F}$ bezeichnet man als die Ordnung bzw. die Kurvenordnung von E. Man schreibt $|E(\mathbb{F})|$ oder $\#E(\mathbb{F})$.*

In der Elliptische-Kurven-Kryptographie werden meist elliptische Kurven über endlichen Körpern $\mathbb{F}_q$ verwendet, wobei q eine Primzahl größer 3 oder eine Zweierpotenz $2^m, m \in \mathbb{N}$, ist. Bei $\mathbb{F}_q$ handelt es sich also entweder um einen Primkörper oder einen binären Erweiterungskörper. Diese Arbeit beschäftigt sich fast ausschließlich mit elliptischen Kurven über Primkörpern.

Für kryptographische Anwendungen ist es notwendig, dass eine elliptische Kurve eine große Ordnung hat (siehe Abschnitt 5.6.1). Die Kurvenordnung lässt sich mit dem Theorem von Hasse abschätzen:

Theorem 6 (Theorem von Hasse)**.** *Für die Anzahl der Punkte auf einer elliptischen Kurve E definiert über einem endlichen Körper $\mathbb{F}_q$ gilt:*

$$||E(\mathbb{F}_q)| - q - 1| \leq 2\sqrt{q}.$$

Beweis. Siehe [Sil09, S. 138]. □

Außerdem lässt sich die Kurvenordnung elliptischer Kurven über Primkörpern mittels eines 1985 von Schoof entwickelten Algorithmus effizient bestimmen [Sch85].

5.1.1 Vereinfachte Weierstraßgleichung

Definition 3 (Isomorphe elliptische Kurven, [CF06, S. 273])**.** *Zwei elliptische Kurven E und E' definiert über einem Körper $\mathbb{F}$ gegeben durch die allgemeine*

Weierstraßgleichung

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$
$$E' : y^2 + a'_1xy + a'_3y = x^3 + a'_2x^2 + a'_4x + a'_6$$

heißen isomorph, wenn $u, r, s, t \in \mathbb{F}, u \neq 0$ *existieren, sodass durch eine bijektive Abbildung der Form*

$$(x, y) \mapsto (u^2x + r, u^3y + u^2sx + t) \tag{5.2}$$

die Gleichung E in E′ überführt wird. Die Abbildung 5.2 wird als admissible change of varibles bezeichnet.
Falls gilt $u, r, s, t \in \overline{\mathbb{F}}$ *und* $u \neq 0$*, so sind E und E′ isomorph über* $\overline{\mathbb{F}}$*. Man sagt E und E′ sind Twists von einander.*

Der unendlich ferne Punkt ∞ bleibt durch eine solche Umformung unverändert.

In Abhängigkeit vom Körper, über dem eine Weierstraß-Kurve definiert wird, lässt sich die Kurvengleichung vereinfachen [HVM03, S. 78f].
Ist der zugrunde liegende Körper einer elliptischen Kurve E mit allgemeiner Weierstraßgleichung ein Primkörper $\mathbb{F}_p$ mit $p > 3$, so kann folgende Transformation der Variablen angewendet werden, um die Kurvengleichung zu vereinfachen:

$$(x, y) \mapsto (\frac{x - 3a_1^2 - 12a_2}{36}, \frac{y - 3a_1x}{216} - \frac{a_1^3 + 4a_1a_2 - 12a_3}{24}).$$

Durch diese Umformung erhält man eine zu E isomorphe elliptische Kurve

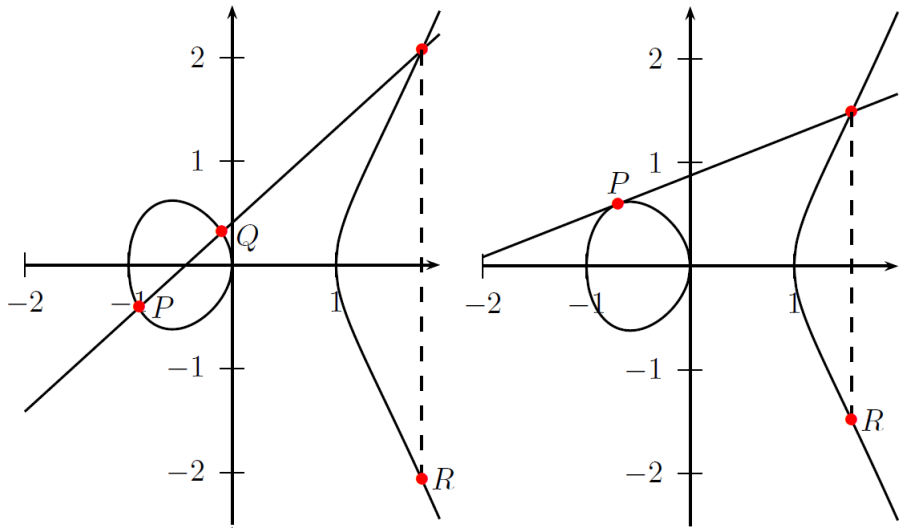
$$E' : y^2 = x^3 + ax + b \tag{5.3}$$

mit $a, b \in \mathbb{F}_p$. Gleichung 5.3 wird als *vereinfachte* oder *kurze Weierstraßgleichung* bezeichnet. Die entsprechende Diskriminante lautet $\Delta = 4a^3 + 27b^2$.

5.1.2 Addition

Um Berechnungen auf der Punktmenge einer elliptischen Kurve ausführen zu können, definieren wir eine Addition. Wir betrachten diese zunächst geometrisch anhand einer elliptischen Kurve über $\mathbb{R}$.
Seien $P = (x_P, y_P)$ und $Q = (x_Q, y_Q)$ zwei verschiedene Punkte auf der elliptischen Kurve $E(\mathbb{R})$. Den Punkt $R = P + Q$ erhält man wie folgt (siehe Abb. 5.2(a)):

1. Bilde eine Gerade g, durch die Punkte P und Q.
2. Finde den Schnittpunkt S der Geraden g und der elliptischen Kurve E.
3. Der Punkt R ergibt sich durch spiegeln von S an der x-Achse.



(a) Punktaddition (b) Punktverdopplung

Abbildung 5.2: Addition auf einer Weierstraß-Kurve

Wenn der Punkt P der Spiegelung von Q bzgl. der x-Achse entspricht, verläuft die Gerade g parallel zur y-Achse. In diesem Fall schneidet g die Kurve E nicht in einem dritten Punkt. Die Summe von P und Q entspricht dann dem unendlich fernen Punkt ∞.
Ein weiterer Fall tritt ein, wenn gilt $P = Q$. Dann lässt sich keine Gerade

durch P und Q konstruieren. Stattdessen berechnet man die Tangente t zu E im Punkt P und ermittelt deren zweiten Schnittpunkt S mit E. Dieser wird wiederum an der x-Achse gespiegelt, um $R = P + P = 2P$ zu erhalten. Man spricht in diesem Fall von Punktverdopplung (siehe Abb. 5.2(b)).
Aus dem geometrischen Vorgehen für die Addition kann eine Additionsvorschrift hergeleitet werden.

Wir betrachten diese für elliptische Kurven über Primkörpern $\mathbb{F}_p$ mit $p > 3$ gegeben durch die vereinfachte Weierstraßgleichung (Gleichung 5.3) [HVM03, S. 79ff]:

Definition 4 (Addition auf $E(\mathbb{F}_p)$)**.** *Sei $E(\mathbb{F}_p)$ eine elliptische Kurve über einem Primkörper $\mathbb{F}_p$ mit $p > 3$ gegeben durch die Kurvengleichung $E : y^2 = x^2 + ax + b$. Die Verknüpfung $+ : E(\mathbb{F}_p) \to E(\mathbb{F}_p)$ ist wie folgt definiert:*

1. *Neutrales Element. Für alle $P \in E(\mathbb{F}_p)$ gilt $P + \infty = \infty + P = P$.*

2. *Inverse. Falls $P = (x_P, y_P) \in E(\mathbb{F}_p)$, dann ist $(x_P, y_P)+(x_P, -y_P) = \infty$. Der Punkt $(x_P, -y_P) \in E(\mathbb{F}_p)$ wird mit $-P$ bezeichnet und ist der zu P inverse Punkt. Es gilt $-\infty = \infty$.*

3. *Punktaddition. Sei $P = (x_P, y_P) \in E(\mathbb{F}_p)$ und $Q = (x_Q, y_Q) \in E(\mathbb{F}_p)$, wobei $P \neq \pm Q$. Dann ist $R = (x_R, y_R) = P + Q \in E(\mathbb{F}_p)$ mit*

$$x_R = \left(\frac{y_Q - y_P}{x_Q - x_P}\right)^2 - x_P - x_Q \text{ und}$$
$$y_R = \left(\frac{y_Q - y_P}{x_Q - x_P}\right)(x_P - x_R) - y_P.$$

4. *Punktverdopplung. Sei $P = (x_P, y_P) \in E(\mathbb{F}_p)$, wobei $P \neq -P$. Dann ist $R = (x_R, y_R) = 2P \in E(\mathbb{F}_p)$ mit*

$$x_R = \left(\frac{3x_P^2 + a}{2y_P}\right)^2 - 2x_P \text{ und}$$
$$y_R = \left(\frac{3x_P^2 + a}{2y_P}\right)(x_P - x_R) - y_P.$$

Die Kommutativität der Addition geht aus der geometrischen Beschreibung hervor. Der Punkt ∞ dient als neutrales Element und zu jedem Punkt $(x, y) \in E(\mathbb{F}_p)$ existiert ein Inverses gegeben durch $(x, -y) \in E(\mathbb{F}_p)$. Für einen Beweis der Assoziativität sei auf [Was08, S. 20ff] verwiesen. Aus diesen Eigenschaften der Addition lässt sich folgern:

Theorem 7. *Sei E eine elliptische Kurve über $\mathbb{F}_p$ und sei $+$ die in Definition 4 definierte interne Verknüpfung auf E. Dann ist $(E(\mathbb{F}_p), +)$ eine additive abelsche Gruppe mit dem neutralen Element ∞.*

Eine Additionsvorschrift für elliptische Kurven mit allgemeiner Weierstraßgleichung ist in [CF06, S. 270] nachzulesen.
Der in Definition 3 beschriebene Isomorphismus bildet nicht nur die Punkte einer elliptischen Kurve auf eine andere ab. Es existiert offensichtlich auch ein bijektiver Homomorphismus bezüglich der Addition, sodass es sich um einen Gruppenisomorphismus handelt.

5.1.3 Projektive Koordinaten

Elliptische Kurven können in einer Reihe verschiedener Koordinatensysteme dargestellt werden. Bisher haben wir elliptische Kurven im affinen Koordinatensystem betrachtet. Sowohl die Punktaddition, als auch die Punktverdopplung erfordern hierbei eine Invertierung in $\mathbb{F}_p$. Bei der Darstellung in projektiven Koordinaten kann auf diese Invertierung verzichtet werden. Dadurch ist eine effizientere Arithmetik möglich.

In einem affinen Koordinatensystem wird ein Punkt einer elliptischen Kurve anhand seiner x- und y-Koordinate dargestellt mit $P = (x_P, y_P)$ oder als virtueller Punkt im Unendlichen ∞.
In projektiven Koordinaten wird P dargestellt durch $P = (X_P, Y_P, Z_P)$ mit $Z_P \neq 0$. Der zu P inverse Punkt ist $-P = (X_P, -Y_P, Z_P)$. Der Punkt im Unendlichen hat die Koordinaten $(0, 1, 0)$.
Zwei Punkte $P = (X_P, Y_P, Z_P)$ und $Q = (X_Q, Y_Q, Z_Q)$ werden als äquivalent bezeichnet, wenn ein $a \neq 0$ existiert, sodass $(X_P, Y_P, Z_P) = (aX_Q, aY_Q, aZ_Q)$.

Um den Punkt P vom affinen Koordinatensystem in ein projektives Koordinatensystem zu transformieren, wird eine zusätzliche Z-Koordinate mit $Z_P = 1$ hinzugefügt. P lautet dann $P = (X_P, Y_P, Z_P) = (x_P, y_P, 1)$. Die Umwandlung von projektiven in affine Koordinaten erhält man mit $P = (x_P, y_P) = (X_P/Z_P, Y_P/Z_P)$.

Durch Transformation der Punkte einer elliptischen Kurve $E(\mathbb{F}_p)$ vom affinen ins projektive Koordinatensystem ändert sich auch die Kurvengleichung und die Additionsvorschrift. Die projektive Kurvengleichung einer elliptischen Kurve $E(\mathbb{F}_p)$ lautet

$$E : Y^2Z = X^3 + aXZ^2 + bZ^3$$

mit $a, b \in \mathbb{F}_p$.

Für eine Übersicht der verschiedenen Koordinatensysteme die zur Darstellung elliptischer Kurven verwendet werden, sei auf [CF06] und [HVM03] verwiesen.

5.1.4 Skalarmultiplikation

Der Sonderfall $Q = P$ der Addition, die Punktverdopplung, lässt vermuten, dass sich neben der Addition auch eine Skalarmultiplikation auf elliptischen Kurven realisieren lässt.

Definition 5 (Skalarmultiplikation, [CF06, S. 271])**.** *Sei E eine elliptische Kurve definiert über einem Primkörper $\mathbb{F}_p$ und sei P ein Punkt auf E. Wir definieren die Skalarmultiplikation des Punktes P mit einem Skalar $d \in \mathbb{N}\backslash\{0\}$ als die d-fache Addition des Punktes P:*

$$dP := \underbrace{P + ... + P}_{d\ mal}.$$

Diese Definition lässt sich für alle $d \in \mathbb{Z}$ erweitern, indem man $0P = \infty$ und $nP = -n(-P)$ für $n < 0$ setzt.

In Anlehnung an den Square-and-Multiply-Algorithmus (siehe [Buc16, S. 51f]) für schnelles Potenzieren, wurde der Double-and-Add-Algorithmus zur effizienten Skalarmultiplikationen auf elliptischen Kurven entworfen.

Algorithmus 7 : Double-and-Add [HVM03, S. 96f]

Input : $P \in E(\mathbb{F}_p),\ d = (d_{t-1}, ..., d_1, d_0)_2$

Output : $Q = dP$

```
1 Q = ∞
2 for i = 0 to t − 1 do
3     if d_i = 1 then
4         Q = Q + P
5     P = 2P
6 return Q
```

Ebenso wie der Square-and-Multiply-Algorithmus ist auch das Double-and-Add-Verfahren angreifbar durch Zeitattacken [OKS00]. Dies werden wir in Abschnitt 5.6.2 genauer betrachten.

Wir haben bereits gesehen, dass man auf der Menge der rationalen Punkte einer elliptischen Kurve additive abelsche Gruppen definieren kann. Somit können wir auch die in Kapitel 2 beschriebenen Eigenschaften von Gruppen auf Gruppen, die auf den Punkten von elliptischen Kurven basieren, übertragen.
Definiert über einem Primkörper $\mathbb{F}_p$, ist auch die Gruppe $(E(\mathbb{F}_p), +)$ endlich. Die Ordnung eines Elements in $E(\mathbb{F}_p)$ wird als Punktordnung bezeichnet. Der Erzeuger B einer zyklischen Gruppe $\langle B \rangle$ heißt Basispunkt.

Definition 6 (Kofaktor, [Han10])**.** *Sei $E(\mathbb{F}_p)$ eine elliptische Kurve und B sei der Basispunkt einer Untergruppe von $E(\mathbb{F}_p)$. Dann wird*

$$h = \frac{|E(\mathbb{F}_p)|}{ord(B)}$$

als Kofaktor bezeichnet.

5.2 Montgomery-Kurven

1987 beschreibt Montgomery in [Mon87] eine neue Form der elliptischen Kurven, die Montgomery-Form. Okeya, Kurumatani und Sakurai beschreiben deren Vorzüge für kryptographische Anwendungen in [OKS00].

Definition 7 (Montgomery-Kurve, [OKS00])**.** *Sei $\mathbb{F}_p$ ein Primkörper mit $p > 3$. Eine elliptische Kurve $E_M(\mathbb{F}_p)$ mit der Kurvengleichung*

$$E_M : By^2 = x^3 + Ax^2 + x \tag{5.4}$$

mit $A, B \in \mathbb{F}_p, A \neq \pm 2$ und $B \neq 0$ heißt Montgomery-Kurve. Die Menge der rationalen Punkte auf E_M über $\mathbb{F}_p$ ist

$$E_M(\mathbb{F}_p) = \{(x, y) \in \mathbb{F}_p^2 \mid By^2 = x^3 + Ax^2 + x\} \cup \{\infty\},$$

wobei ∞ der unendlich ferne Punkt ist.

Die Klasse der Montgomery-Kurven ist beschränkt gegenüber elliptischen Kurven der Weierstraß-Form. Jede Montgomery-Kurve lässt sich in eine Weierstraß-Kurve überführen, die Umkehrung gilt jedoch nicht (siehe Abschnitt 5.5).
Offensichtlich liegt für jede beliebige Montgomery-Kurve der Punkt $(0, 0)$ auf der Kurve. Wie sich anhand der im Folgenden definierten Additionsvorschrift bestimmen lässt, hat er die Ordnung 2. Es existieren allerdings Weierstraß-Kurven, die keine Punkte der Ordnung 2 enthalten.
Eine weitere Besonderheit der Montgomery-Kurven ist, dass die Kurvenordnung in jedem Fall durch 4 teilbar ist. Diese Eigenschaft haben Okeya et al. in [OKS00] gezeigt.
Das NIST empfiehlt in [NIS99], dass der Kofaktor einer Gruppe basierend auf elliptischen Kurven möglichst klein sein sollte. Daher wird angestrebt nur Montgormery-Kurven, deren Kofaktor genau 4 ist, in der ECC zu verwenden. In [OKS00] wird ein Algorithmus zur Erzeugung solcher Kurven vorgestellt.

5.2.1 Addition

Die Additionsvorschrift für Montgomery-Kurven lautet wie folgt:

Definition 8 (Addition auf $E_M(\mathbb{F}_p)$, [OKS00])**.** *Sei $E_M(\mathbb{F}_p)$ eine Montgomery-Kurve über einem Primkörper $\mathbb{F}_p$ mit $p > 3$. Die Verknüpfung $+ : E_M(\mathbb{F}_p) \to E_M(\mathbb{F}_p)$ ist wie folgt definiert:*

1. *Neutrales Element. Für alle $P \in E_M(\mathbb{F}_p)$ gilt $P + \infty = \infty + P = P$.*

2. *Inverse. Falls $P = (x_P, y_P) \in E_M(\mathbb{F}_p)$, dann ist $(x_P, y_P) + (x_P, -y_P) = \infty$. Der Punkt $(x_P, -y_P) \in E_M(\mathbb{F}_p)$ wird mit $-P$ bezeichnet und ist der zu P inverse Punkt. Es gilt $-\infty = \infty$.*

3. *Punktaddition. Sei $P = (x_P, y_P) \in E_M(\mathbb{F}_p)$ und $Q = (x_Q, y_Q) \in E_M(\mathbb{F}_p)$, wobei $P \neq \pm Q$. Dann ist $R = (x_R, y_R) = P + Q \in E_M(\mathbb{F}_p)$ mit*

$$x_R = B\lambda^2 - A - x_P - x_Q \text{ und}$$
$$y_R = \lambda(x_P - x_R) - y_P$$

   mit $\lambda = (y_Q - y_P)/(x_Q - x_P)$.

4. *Punktverdopplung. Sei $P = (x_P, y_P) \in E_M(\mathbb{F}_p)$, wobei $P \neq -P$. Dann ist $R = (x_R, y_R) = 2P \in E_M(\mathbb{F}_p)$ mit*

$$x_R = B\lambda^2 - A - 2x_P \text{ und}$$
$$y_R = \lambda(x_P - x_R) - y_P$$

   mit $\lambda = (3x_P^2 + 2Ax_P + 1)/(2By_P)$.

Es ist anzumerken, dass sowohl für die Punktaddition, als auch für die Punktverdopplung λ definiert ist, da der Nenner in jedem Fall ungleich Null ist. Falls $x_Q - x_P = 0$ ist, so gilt für $P = \pm Q$. Es wird also entweder der Fall 2 oder 4 der Additionsvorschrift angewandt. Ist $2By_P = 0$ so ist entweder $B = 0$ oder $y_P = 0$. Laut Kurvendefinition gilt $B \neq 0$. Falls gilt $y_P = 0$, so ist der Punkt $P = (x_P, 0) = -P$ und der zweite Fall der Additionsvorschrift wird verwendet.

Die Menge $E_M(\mathbb{F}_p)$ bildet mit der oben beschriebenen Addition eine abelsche Gruppe mit neutralem Element ∞. Die Gruppeneingenschaften lassen sich durch simple jedoch etwas längliche Rechnungen nachweisen.

5.2.2 Montgomery-Leiter

Zusammen mit der Montgomery-Form für elliptische Kurven, hat Montgomery in [Mon87] auch ein Verfahren zur effizienten Skalarmultiplikation auf Montgomery-Kurven veröffentlicht, die Montgomery-Leiter.

Algorithmus 8 : Montgomery-Leiter [Mon87]

Input : $P \in E_M(\mathbb{F}_p),\ d = (d_{t-1}, ..., d_1, d_0)_2$

Output : $Q = dP$

```
1  Q = ∞
2  R = P
3  for i = t − 1 to 0 do
4      if d_i = 1 then
5          Q = Q + R
6          R = 2R
7      else
8          R = R + Q
9          Q = 2Q
10 return Q
```

Nach jedem Schleifendurchlauf gilt $Q = d'P$ und $R = (d'+1)P$ mit $0 \leq d' \leq d$ [OKS00]. Nach Ablauf des Algorithmus enthält Q den gesuchten Punkt dP. Die Korrektheit des Algorithmus lässt sich induktiv beweisen.

In einigen Anwendung der ECC wird die y-Koordinate der Punkte nicht betrachtet. In solchen Fällen empfiehlt sich eine Addition auf projektiven Koordinaten, die auf die Berechnung der y-Koordinaten verzichtet.
Sei $P = (X, Y, Z)$ ein Punkt auf einer Montgomery-Kurve $E_M(\mathbb{F}_p)$. Seien weiter $mP = (X_m, Y_m, Z_m)$ und $nP = (X_n, Y_n, Z_n)$ zwei Vielfache des Punktes P. Dann lässt sich $(m + n)P = mP + nP$ wie folgt berechnen [Mon87]:

- Für $m \neq n$:

$$X_{m+n} = Z_{m-n}[(X_m - Z_m)(X_n + Z_n) + (X_m + Z_m)(X_n - Z_n)]^2$$
$$Z_{m+n} = X_{m-n}[(X_m - Z_m)(X_n + Z_n) - (X_m + Z_m)(X_n - Z_n)]^2$$

- Für $m = n$:

$$4X_nZ_n = (X_n + Z_n)^2 - (X_n - Z_n)^2$$
$$X_{2n} = (X_n + Z_n)^2(X_n - Z_n)^2$$
$$Z_{2n} = (4X_nZ_n)((X_n - Z_n)^2 + ((A+2)/4)(4X_nZ_n))$$

Da bei der Montgomery-Leiter nach jedem Schritt gilt $Q = d'P$ und $R = (d'+1)P$ für ein $0 \leq d' \leq d$, ist die Differenz von R und Q gegeben durch $R - Q = (d'+1)P - d'P = P$. Die Koordinaten X_{m-n} und Z_{m-n} sind also genau die X- und Z-Koordinaten von P.
Falls benötigt, lässt sich auch bei Verwendung von projektiven Koordinaten die y-Koordinate des Ergebnispunkts berechnen [CF06, S. 286].

Brier und Joye haben die Skalarmultiplikation von Montgomery in [BJ02] für Kurven mit vereinfachter Weierstraßgleichung verallgemeinert.
Während die Berechnungen des Double-and-Add-Algorithmus (siehe Algortihmus 7) davon abhängig sind, ob ein bestimmtes Bit des Skalars 0 oder 1 ist, ist der Berechnungsaufwand der Montgomery-Leiter davon unabhängig. Für jedes Bit des Skalars wird eine Punktaddition und eine Punktverdopplung durchgeführt. Diese Eigenschaft macht die Montgomery-Leiter im Vergleich zum Double-and-Add-Algorithmus resistenter gegenüber Zeitattacken (siehe Abschnitt 5.6.2) [OKS00].

5.2.3 Curve25519

2006 veröffentlichte Bernstein in [Ber06] die Curve25519, eine Montgomery-Kurve, deren Parameter so gewählt sind, dass sie eine besonders schnelle Arithmetik erlauben. Mit seiner Implementierung konnte er neue Laufzeit-

Rekorde für das Schlüsselaustauschverfahren *Elliptic Curve Diffie-Hellman* (ECDH) erzielen.
Die Kurve ist definiert über dem Primkörper $\mathbb{F}_p$ mit $p = 2^{255} - 19$. Die Primzahl p verleiht der Kurve ihren Namen. Die Kurvengleichung lautet

$$y^2 = x^3 + 486662x^2 + x.$$

In seiner Implementierung verwendet Bernstein die Montgomery-Leiter für die Skalarmultiplikation. Dadurch und durch die Vermeidung von eingabeabhängigen Array-Indizes verhindert er bestimmte Seitenkanalangriffe (siehe Abschnitt 5.6.2) [Ber06].

5.3 Edwards-Kurven

Eine weitere Form der elliptischen Kurven sind Edwards-Kurven. Sie wurden erstmals 2007 von Edwards in [Edw07] beschrieben. Sie sind spezieller als Weierstraß- und Montgomery-Kurven, bieten jedoch eine schnelle und einfache Punktaddition.

Definition 9 (Edwards-Kurve, [BL07])**.** *Sei $\mathbb{F}_p$ ein Primkörper mit $p > 3$. Eine Edwards-Kurve E_{Ed} über $\mathbb{F}_p$ wird definiert durch die Kurvengleichung*

$$E_{Ed} : x^2 + y^2 = 1 + dx^2y^2$$

mit $d \in \mathbb{F}_p \setminus \{0, 1\}$.

Zu den Besonderheiten einer Edwards-Kurve zählt, dass der Punkt $(0, -1)$ mit der Ordnung 2 und die Punkte $(1, 0)$ und $(-1, 0)$ mit Ordnung 4 auf der Kurve liegen. Dies lässt sich leicht nachrechnen. Die Existenz von Punkten der Ordnung 4 beschränkt die Anzahl verschiedener Edwards-Kurven gegenüber der von Weierstraß- bzw. Montgomery-Kurven.

Definition 10 (Addition auf $E_{Ed}(\mathbb{F}_p)$, [BL07])**.** *Sei $E_{Ed}(\mathbb{F}_p)$ eine Edwards-Kurve über einem Primkörper $\mathbb{F}_p$ mit $p > 3$. Die Verknüpfung $+ : E_{Ed}(\mathbb{F}_p) \to E_{Ed}(\mathbb{F}_p)$ ist wie folgt definiert:*

Sei $P = (x_P, y_P) \in E_{Ed}(\mathbb{F}_p)$ *und* $Q = (x_Q, y_Q) \in E_{Ed}(\mathbb{F}_p)$. *Dann ist* $R = (x_R, y_R) = P + Q \in E_{Ed}(\mathbb{F}_p)$ *mit*

$$x_R = \frac{x_P y_Q + y_P x_Q}{1 + d x_P x_Q y_P y_Q}$$
$$y_R = \frac{y_P y_Q - x_P x_Q}{1 - d x_P x_Q y_P y_Q},$$

wobei $d \in \mathbb{F}_p \setminus \{0, 1\}$.

Das Inverse zu einem Punkt $P = (x_P, y_P)$ auf $E(\mathbb{F}_p)$ ist $-P = (-x_P, y_P)$. Wenn d kein Quadrat in $\mathbb{F}_p$ ist, so sind die Nenner ungleich 0. Die Addition ist in diesem Fall *vollständig*, d.h. sie ist für alle möglichen Eingaben definiert, wie in [BL07] bewiesen.
$(E_{Ed}(\mathbb{F}_p), +)$ ist dann eine Gruppe mit dem Punkt $(0, 1)$ als neutrales Element der Addition, wie sich leicht zeigen lässt.
Falls d ein Quadrat in $\mathbb{F}_p$ ist, so ist die Addition nicht vollständig, da Punkte $P = (x_P, y_P)$ und $Q = (x_Q, y_Q)$ existieren können, sodass $1 + d x_P x_Q y_P y_Q = 0$ oder $1 - d x_P x_Q y_P y_Q = 0$. Für kryptographische Anwendungen wird daher ein d gewählt, das kein Quadrat in $\mathbb{F}_p$ ist.

Die Vorteile dieser Arithmetik sind offensichtlich. Die einheitliche Additionsvorschrift erfordert keine Fallunterscheidungen. Dadurch wird die Addition nicht nur übersichtlicher und somit weniger anfällig für Programmierfehler, sondern auch resistent gegen gewisse Seitenkanalangriffe, die eine Unterscheidung von Punktaddition und Punktverdopplung ausnutzen.

5.4 Twisted-Edwards-Kurven

Bernstein et al. entwickelten 2008 eine Kurven-Form, die auf der der Edwards-Kurven basiert und deren Vorzüge bzgl. der Addition nutzt, gleichzeitig aber die gleiche Vielfalt von Kurven unterstützt wie die Montgomery-Form [BBJ+08]. Im Vergleich zur Kurvengleichung der Edwards-Kurven wird nur ein zusätzlicher Koeffizient a hinzugefügt, der einen Twist der Kurve bewirkt (siehe Abschnitt 5.5.3). Sie werden daher als Twisted-Edwards-Kurven bezeichnet.

Definition 11 (Twisted-Edwards-Kurve, [BBJ+08]). *Sei $\mathbb{F}_p$ ein Primkörper mit $p > 3$. Seien weiter $a, d \in \mathbb{F}_p$ mit $a, d \neq 0$ und $a \neq d$. Dann ist die Twisted-Edwards-Kurve $E_{TEd}(\mathbb{F}_p)$ die Menge aller Punkte (x, y), die die Gleichung*

$$E_{TEd} : ax^2 + y^2 = 1 + dx^2y^2$$

erfüllen.
Eine Edwards-Kurve ist eine Twisted-Edwards-Kurve mit $a = 1$.

Eine Twisted-Edwards-Kurve enthält in jedem Fall den Punkt $(0, -1)$, der die Ordnung 2 hat. Dies lässt sich anhand folgender Additionsvorschrift nachprüfen:

Definition 12 (Addition auf $E_{TEd}(\mathbb{F}_p)$, [BBJ+08]). *Sei $E_{TEd}(\mathbb{F}_p)$ eine Twisted-Edwards-Kurve über einem Primkörper $\mathbb{F}_p$ mit $p > 3$. Die Verknüpfung $+ : E_{TEd}(\mathbb{F}_p) \to E_{TEd}(\mathbb{F}_p)$ ist wie folgt definiert:*
Sei $P = (x_P, y_P) \in E_{TEd}(\mathbb{F}_p)$ und $Q = (x_Q, y_Q) \in E_{TEd}(\mathbb{F}_p)$. Dann ist $R = (x_R, y_R) = P + Q \in E_{TEd}(\mathbb{F}_p)$ mit

$$x_R = \frac{x_P y_Q + y_P x_Q}{1 + dx_P x_Q y_P y_Q}$$
$$y_R = \frac{y_P y_Q - a x_P x_Q}{1 - dx_P x_Q y_P y_Q}.$$

Das Inverse zu einem Punkt $P = (x_P, y_P)$ auf $E_{TEd}(\mathbb{F}_p)$ ist $-P = (-x_P, y_P)$. Die Addition ist vollständig, wenn a ein Quadrat in $\mathbb{F}_p$ ist und d kein Quadrat in $\mathbb{F}_p$ ist [BBJ+08]. In diesem Fall ist $(E_{TEd}(\mathbb{F}_p), +)$ eine Gruppe mit neutralem Element $(0, 1)$. Durch simple Rechnungen lässt sich beweisen, dass $(E_{TEd}(\mathbb{F}_p), +)$ die Gruppeneigenschaften erfüllt.

5.5 Transformationen zwischen elliptischen Kurven

In diesem Kapitel betrachten wir die Beziehungen zwischen den verschiedenen Formen elliptischer Kurven. Wir nutzen Isomorphismen und Twists (Definition 3), um die Kurven zu transformieren.
Dabei ist anzumerken, dass sich grundsätzlich jede der beschriebenen Kurven in Weierstraß-Kurven mit allgemeiner Weierstraßgleichung (Gleichung 5.1) umformen lassen. Da Montgomery-, Edwards- und Twisted-Edwards-Kurven spezielle Eigenschaften haben, ist die Umkehrung nur unter bestimmten Bedingungen möglich.

5.5.1 Weierstraß-Kurven und Montgomery-Kurven

Jede Kurve in Montgomery-Form lässt sich in eine elliptische Kurve mit vereinfachter Weierstraßgleichung (Gleichung 5.3) überführen, indem man für die Koeffizienten der Weierstraß-Kurve einsetzt:

$$a = \frac{3 - A^2}{3B^2} \text{ und } b = \frac{2A^3 - 9A}{27B^3}.$$

Die Transformation einer Weierstraß-Kurve in eine Montgomery-Kurve ist nicht in jedem Fall möglich. Eine elliptische Kurve in Weierstraß-Form $E : y^2 = x^3 + ax + b$ definiert über einem Primkörper $\mathbb{F}_p$ mit $p > 3$ ist genau dann in eine Montgomery-Kurve $E_M : Bv^2 = u^3 + Au^2 + u$ überführbar, wenn die folgenden Bedingungen erfüllt sind:

1. Die Gleichung $x^3 + ax + b = 0$ hat wenigstens eine Nullstelle α in $\mathbb{F}_p$.
2. Die Zahl $3\alpha^2 + a$ ist ein quadratischer Rest in $\mathbb{F}_p$.

Okeya et al. haben dies in [OKS00] bewiesen. Wie bereits erwähnt, enthält jede Montgomery-Kurve den Punkt $(0, 0)$ mit Ordnung 2. Also muss auch die transformierte Kurve E einen Punkt dieser Ordnung enthalten. Existiert die Nullstelle α, so liegt $(\alpha, 0)$ auf E und hat die Ordnung 2. Der Punkt $(\alpha, 0)$ auf E wird also auf den Punkt $(0, 0)$ auf E_M abgebildet.

Die Transformation der Kurvenpunkte lautet [CF06, S. 286]:

$$(x, y) \mapsto (u, v) = (s(x - \alpha), sy),$$

wobei s die Quadratwurzel von $(3\alpha^2+a)^{-1}$ ist. Als Parameter der Montgomery-Kurve erhält man

$$A = 3\alpha s \text{ und } B = s.$$

Erfüllt eine Weierstraß-Kurve E die genannten Bedingungen, so existiert also eine Montgomery-Kurve E_M die isomorph zu E über $\mathbb{F}_p$ ist.

5.5.2 Montgomery-Kurven und Twisted-Edwards-Kurven

Montgomery-Kurven und Twisted-Edwards-Kurven haben sehr ähnlichen Eigenschaften. Beide enthalten Punkte der Ordnung 2 und die Kurvenordnung ist durch 4 teilbar [BBJ+08].

Sei $E_{TEd} : ax^2 + y^2 = 1 + dx^2y^2$ eine Twisted-Edwards-Kurve. Die Kurve lässt sich in eine isomorphe Montgomery-Kurve überführen, falls a ein Quadrat und d kein Quadrat in $\mathbb{F}_p$ ist. Nur in diesem Fall ist die Addition auf E_{TEd} vollständig (siehe Abschnitt 5.4). Die Transformation

$$(x, y) \mapsto (u, v) = \begin{cases} \infty & \text{falls } (x, y) = (0, 1), \\ (0, 0) & \text{falls } (x, y) = (0, -1), \\ \left(\dfrac{1+y}{1-y}, \dfrac{1+y}{x(1-y)}\right) & \text{sonst} \end{cases} \tag{5.5}$$

bildet die Punkte von E_{TEd} auf die der isomorphen Montgomery-Kurve $E_M : Bv^2 = u^3 + Au^2 + u$ ab. Die Koeffizienten der Montgomery-Kurve lauten $A = 2(a + d)/(a - d)$ und $B = 4/(a - d)$.

Umgekehrt bildet die inverse Transformation

$$(u, v) \mapsto (x, y) = \begin{cases} (0, 1) & \text{falls } (u, v) = \infty, \\ (0, -1) & \text{falls } (u, v) = (0, 0), \\ \left(\frac{u}{v}, \frac{u-1}{u+1}\right) & \text{sonst} \end{cases} \tag{5.6}$$

die Punkte einer Montgomery-Kurve E_M auf die einer isomorphen Twisted-Edwards-Kurve ab, falls gilt:

1. $(A + 2)/B$ ist ein Quadrat in $\mathbb{F}_p$,
2. $(A - 2)/B$ ist kein Quadrat in $\mathbb{F}_p$.

Die Koeffizienten von E_{TEd} sind dann $a = (A + 2)/B$ und $d = (A - 2)/B$ [BL07, BBJ+08]. Damit ist also die Addition auf der isomorphen Twisted-Edwards-Kurve vollständig (siehe Abschnitt 5.4).

Die Transformationen ordnen die neutralen Elemente ($(0, 1) \in E_{TEd}$ und $\infty \in E_M$) einander zu. $(0, -1) \in E_{TEd}$ und $(0, 0) \in E_M$ sind die Punkte mit Ordnung 2, wie sich leicht nachrechnen lässt. Auch sie werden entsprechend auf einander abgebildet.

Isomorphie gilt nur unter den oben beschriebenen Bedingungen. Sind diese nicht gegeben, so ist die Addition auf der Twisted-Edwards-Kurve nicht vollständig. Die Montgomery-Kurve kann dann zusätzliche Punkte der Ordnung 2 und 4 enthalten, die nicht auf eine Twisted-Edwards-Kurve abbildbar sind. Allgemein gilt jedoch, dass jede Montgomery-Kurve birational äquivalent zu einer Twisted-Edwards-Kurve ist und umgekehrt (siehe [BBJ+08, BL07]). Wir werden jedoch nur solche Kurven betrachten, für die gilt $a = (A + 2)/B$ ist ein Quadrat und $d = (A - 2)/B$ ist kein Quadrat in $\mathbb{F}_p$.

Durch die Transformation einer Weierstraß-Kurve in eine Montgomery-Kurve und anschließende Umformung in eine Twisted-Edwards-Kurve lassen sich Weierstraß-Kurven in Twisted-Edwards-Kurven umwandeln [BBJ+08].

5.5.3 Edwards-Kurven und Twisted-Edwards-Kurven

Ein wesentlicher Nachteil der Edwards-Kurven ist, dass die Anzahl solcher Kurven begrenzt ist durch die Voraussetzung, dass sie einen Punkt der Ordnung 4 enthalten. Bernstein et al. haben daher in [BBJ+08] Twisted-Edwards-Kurven als eine Verallgemeinerung der Edwards-Kurven eingeführt. Damit erhält man eine größere Vielfalt an Kurven, die die Vorteile der Arithmetik von Edwards-Kurven nutzen.

Wir betrachten zunächst, wie wir aus einer Edwards-Kurve eine Twisted-Edwards-Kurve konstruieren können. Sei

$$E_{Ed} : u^2 + v^2 = 1 + \frac{d}{a}u^2v^2$$

eine Edwards-Kurve über einem Primkörper $\mathbb{F}_p$ mit $p > 3$. Dann ist

$$E_{TEd} : ax^2 + y^2 = 1 + dx^2y^2$$

ein quadratischer Twist von E_{Ed} über dem Körper $\mathbb{F}_p(\sqrt{a})$, dem kleinsten Erweiterungskörper von $\mathbb{F}_p$, in dem die Gleichung $x^2 = a$ eine Lösung hat.
Die Abbildung

$$(u, v) \mapsto (x, y) = (\frac{u}{\sqrt{a}}, v)$$

transformiert E_{Ed} nach E_{TEd}. Insbesondere gilt, wenn a ein Quadrat in $\mathbb{F}_p$ ist, so ist E_{TEd} isomorph zu E_{Ed} über $\mathbb{F}_p$ [BBJ+08].

5.6 Diskreter-Logarithmus-Problem für Elliptische Kurven

Analog zum allgemeinen Diskreter-Logarithmus-Problem, das wir in Kapitel 2.6 kennengelernt haben, lässt sich auch für elliptische Kurven ein Diskreter-Logarithmus-Problem (ECDLP) definieren.

Definition 13 (ECDLP, [HVM03, S. 153])**.** *Sei E eine elliptische Kurve definiert über einem Primkörper $\mathbb{F}_p$. Sei weiter B ein Punkt auf $E(\mathbb{F}_p)$ mit Ordnung n und $A \in \langle B \rangle$. Dann existiert eine Zahl $0 \leq d \leq n-1$, sodass gilt*

$$A = dB.$$

Die Zahl d wird bezeichnet als der diskrete Logarithmus von A zur Basis B. Man schreibt $d = log_B A$.

Es wird angenommen, dass es schwierig ist, eine solche Zahl zu finden, da bisher kein Algorithmus bekannt ist, der in subexponentieller Laufzeit das ECDLP für nicht-singuläre Kurven lösen kann [MvOV96]. Diskrete Logarithmen in $\mathbb{Z}_p$ können mithilfe des Zahlkörpersiebs bereits mit subexpoentieller Laufzeit berechnet werden. So konnten Adrian et al. 2015 das DLP in Gruppen mit einer Primzahlordnung von 512 Bit lösen [AVV+15]. Dadurch ermöglichen Signaturverfahren der ECC kürzere Schlüssellängen als beispielsweise der DSA [BSI16].

5.6.1 Bekannte Angriffe

Die vermutete Schwierigkeit des ECDLP ist wesentlich für Verfahren der ECC. So wie für das DLP in $\mathbb{Z}_p^*$ sind allerdings auch für das ECDLP Angriffe bekannt.
Der einfachste Algorithmus zum Lösen des ECDLP ist die Brute-Force-Methode. Dabei werden nacheinander die Punkte $B, 2B, 3B, ...$ berechnet, bis A gefunden ist. Im schlimmsten Fall werden dafür $n = ord(B)$ Schritte benötigt. Ein Angriff dieser Art lässt sich verhindern, indem man einen Basispunkt B mit großer Ordnung (z.B. $n > 2^{100}$) wählt.
Weitaus effizientere Angriffe können mit einer Kombination aus dem Pohlig-Hellman- und dem Pollard-Rho-Algorithmus erzielt werden. Damit erreicht man eine Laufzeit von $O(\sqrt{p})$, wobei p der größte Primfaktor von n ist. Um diesen Angriff zu verhindern, sollten die Kurvenparameter so gewählt werden, dass n durch eine große Primzahl (z.B. $p > 2^{200}$) teilbar ist, was die Berechnung von $\sqrt{p}$ Schritten praktisch unmöglich macht.

Außerdem gibt es Angriffe, die das ECDLP auf bestimmten elliptischen Kurven lösen. Dazu wird die entsprechende Kurve auf eine isomorphe Gruppe abgebildet, in der das DLP leicht oder zumindest in subexponentieller Laufzeit berechenbar ist. Man spricht daher von Isomorphismus-Angriffen [HVM03]. Zu dieser Art von Angriffen zählen der MOV-Algorithmus und der SSSA-Algorithmus [Wer02, S. 82ff]. Um diese Angriffe zu verhindern, fordert man für Gruppen auf elliptischen Kurven, dass $|E(\mathbb{F}_p)| \neq p$ und dass die Ordnung n des Basispunkts B für alle $1 \leq k \leq 20$ die Zahl $p^k - 1$ nicht teilt [JMV01].

5.6.2 Seitenkanalangriffe

Während die oben betrachteten Angriffe die mathematischen Eigenschaften eines Kryptoverfahrens ausnutzen, lassen sich auch über spezielle Eigenschaften der Implementierung oder Betriebsumgebung Kenntnisse über geheime Informationen gewinnen. Solche Angriffe werden als Seitenkanalangriffe bezeichnet. Durch Beobachtung von Schwankungen

- der Berechnungszeiten,
- des Stromverbrauchs oder
- der elektromagnetischen Abstrahlung

können Rückschlüsse auf einen privaten Schlüssel gezogen werden [KLLT11]. Außerdem können durch absichtliches Herbeiführen von Fehlern geheime Informationen gewonnen oder der Programmfluss beeinflusst werden. Da bei Seitenkanalangriffen meist ein physikalischer Zugriff auf das Kryptosystem notwendig ist, eignen sich insbesondere Smartcards als Angriffsziel [PP10].

Die Rechenoperationen eines Algorithmus nehmen unterschiedlich viel Zeit in Anspruch. Ein Angriff, der Informationen durch Analyse von Rechenzeiten oder der Gesamtlaufzeit eines Algorithmus gewinnt, wird als *Zeitattacke* bezeichnet. Um die kleinen Unterschiede in der Berechnungszeit zu messen, wird eine Zeitattacke meist mit einer Analyse des Stromverbrauchs (*Simple*

Power Analysis) kombiniert. Neben der Zeit kann man auch durch die Höhe des Stromverbrauchs beurteilen, welche Operation gerade ausgeführt wird.
Bei manchen Operationen ergeben sich abhängig vom Eingabewert minimale Unterschiede in der Laufzeit oder dem Stromverbrauch. Durch mehrfache Messungen und Anwendung von statistischen Methoden können auch solche Unterschiede Aufschluss über die verarbeiteten Bits geben. Solche Analysen werden als *Differential Power Analysis* bezeichnet.

In Verfahren der ECC werden oftmals diskrete Logarithmen als private Schlüssel verwendet. Die Skalarmultiplikation mit einem privaten Schlüssel ist daher ein attraktiver Angriffspunkt für Seitenkanalangriffe. Eine ausführlichere Betrachtung von möglichen Angriffen auf Verfahren der ECC bieten [CF06], [HVM03] und [KLLT11].

Im Folgenden werden wir auf einige typische Angriffspunkte in der Skalarmultiplikation auf elliptischen Kurven eingehen. In Kapitel 5.1.4 haben wir den Double-and-Add-Algorithmus (Algorithmus 7) für die Skalarmultiplikation auf elliptischen Kurven kennengelernt.
Der Algorithmus führt in Abhängigkeit von den Bits des Skalars d entweder eine Punktverdopplung oder eine Punktverdopplung und eine Punktaddition durch. Je nach verwendeter Kurve und Koordinatendarstellung ist die Laufzeit von Punktadditon und -verdopplung unterschiedlich lang. Meist dauert eine Addition länger als eine Verdopplung. Eine Analyse der Berechnungszeiten kann daher Aufschluss über den Wert von d geben.

Betrachten wir die Sequenz aus Punktadditionen und -verdopplungen in Abbildung 5.3 als Teil eines Double-and-Add-Algorithmus. Man sieht, dass die Spitzen des Stromverbrauchs unterschiedlich breit sind. Die breiten Spitzen entsprechen also einer Punktaddition und die schmaleren einer Punktverdopplung. Da eine Addition nur ausgeführt wird, wenn das betrachtete Bit des Skalars 1 ist, können wir somit die Binärdarstellung von d bestimmen.

An dieser Stelle wird deutlich, dass eine konstante, von d unabhängige Re-

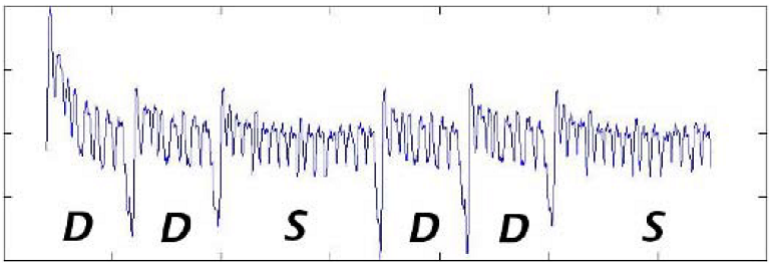


Abbildung 5.3: Stromverbrauchsprofil einer Sequenz von Punktadditionen (S) und -verdopplungen (D) auf einer Weierstraß-Kurve [HVM03, S. 240]

chenzeit notwendig ist, um den Skalar d geheim zu halten. Eine Modifikation des Double-and-Add-Algorithmus ist Algorithmus 9. Er führt in jedem Schleifendurchlauf eine Punktverdopplung und eine Punktaddition aus und lässt somit keine Interpretation der Abfolge von Addition und Verdopplung mehr zu. Auch die Montgomery-Leiter (Algorithmus 8) ist gegen solche Angriffe resistent [CF06, S. 676].

Algorithmus 9 : Double-and-Add-Always [CF06, S. 676]

Input : $P \in E(\mathbb{F}_p),\ d = (d_{t-1}, ..., d_1, d_0)_2$

Output : $Q_0 = dP$

1 $Q_0 = \infty$
2 **for** $i = 0$ *to* $t - 1$ **do**
3 $\quad Q_0 = 2Q_0$
4 $\quad Q_1 = Q_0 + P$
5 $\quad Q_0 = Q_{d_i}$
6 **return** Q_0

Neben unterschiedlichen Laufzeiten der Rechenoperationen können auch Sprünge ein Programm angreifbar machen. Wird beispielsweise eine *If*-Konstruktion verwendet, so wird einfach die nächste Instruktion ausgeführt, falls die Bedingung wahr ist. Falls die Bedingung falsch ist wird ein Sprung ausgeführt. Da ein Sprung für gewöhnlich länger dauert als kein Sprung, lässt sich mit Blick auf das Stromprofil schließen, ob die entsprechende Bedingung wahr oder falsch war. Daher ist es ratsam auf Verzweigungen, die von geheimen Schlüsseln abhängig sind, zu verzichten [CF06, S. 676].

Das Gleiche gilt für geheime Array-Indizes. Die Zeit, die benötigt wird um auf die jeweilige Adresse zuzugreifen, kann charakteristisch sein für den verwendeten Array-Index [Ber05].

Kapitel 6

ECDSA

Der Elliptic Curve Digital Signature Algorithm (ECDSA) ist eine digitales Signaturverfahren basierend auf der Arithmetik elliptischer Kurven. Das Protokoll gleicht dem in Kapitel 4.3 beschriebenen DSA.
Das Verfahren wurde 1998 als International Standards Organization (ISO) Standard (ISO 14888-3 [ISO98]) und später auch als American National Standards Institute (ANSI) Standard (ANSI X9.62 [ANS99]), Institute of Electrical and Electronics Engineers (IEEE) Standard (IEEE 1363-200 [IEE00]) und NIST Standard (FIPS 186-2 [NIS00]) akzeptiert. Das im Folgenden beschriebenen Verfahren basiert auf dem Standard ANSI X9.62 [ANS99].

6.1 Das Signaturverfahren

6.1.1 Systemparameter

Die Systemparameter für den ECDSA bestehen aus folgenden Komponenten:

1. Ein Sicherheitsparameter l.

2. Eine kryptographische Hashfunktion H (z.B. SHA-1).

3. Die Ordnung p eines Primkörpers $\mathbb{F}_p$ mit $p > 3$ und Bitlänge l.

4. Zwei Koeffizienten $a, b \in \mathbb{F}_p$, die die Gleichung einer Weierstraß-Kurve

$E(\mathbb{F}_p)$ definieren. Die Kurvengleichung lautet

$$E : y^2 = x^3 + ax + b.$$

5. Ein Basispunkt $B \in E(\mathbb{F}_p)$ primer Ordnung.

6. Die Ordnung n von B, mit $n > 2^{160}$ und $n > 4\sqrt{p}$.

Um die ECDSA-Schlüssel gegenüber bekannten Angriffen resistent zu machen, müssen einige Bedingungen bei der Wahl der Systemparameter beachtet werden.
Durch den Punkt B wird eine Untergruppe $\langle B \rangle$ von $E(\mathbb{F}_p)$ erzeugt. Wie beim DSA werden Berechnungen nur in der Untergruppe ausgeführt. Um gegenüber Angriffen durch den Pohlig-Hellman- oder Pollard-Rho-Algorithmus sicher zu sein, muss die Ordnung n des Basispunktes eine große Primzahl sein. Die Kurvenordnung $|E(\mathbb{F}_p)|$ sollte also durch eine große Primzahl teilbar sein (siehe Kapitel 5.6). Mit $n > 2^{160}$ sind solche Angriffe praktisch unmöglich[1].
Für einen gegebenen Körper $\mathbb{F}_p$ sollte n so groß wie möglich gewählt werden, $n \approx p$. Man fordert daher zusätzlich, dass $n > 4\sqrt{p}$. Gleichzeitig wird der Kofaktor $|E(\mathbb{F}_p)|/n$ also klein.
Weitere Bedingungen an die Ordnung von $\langle B \rangle$ sind, dass $p^k - 1$ für alle $k \in [1, 20]$ nicht durch n teilbar sein und $n \neq p$ sein soll. Damit sollen die in Kapitel 5.6.1 genannten Isomorphismus-Angriffe ausgeschlossen werden.
Für die Wahl der Kurvenparameter gibt es verschiedene Techniken. Entweder man wählt eine spezielle als sicher angesehene Kurve oder man erzeugt die Kurvenparameter zufällig. Die erste Möglichkeit hat den Vorteil, dass die Kurvenparameter so gewählt werden können, dass sich die Berechnungen auf der Kurve besonders schnell durchführen lassen. Empfohlene Parameter für elliptische Kurven über Primkörpern sind im NIST-Standard FIPS 186-4 [NIS13] und im Brainpool Standard RFC 5639 [LM10] zu finden. Wählt man die Parameter zufällig, so wird das Verfahren resistent gegenüber Angriffen, die auf bestimmte Kurven spezialisiert sind [JMV01].

[1] Nach heutigem Stand der Technik sollte $n > 2^{200}$ gelten.

Algorithmus 10 : ECDSA-Systemparametergenerierung

Input : Bitlängen l

Output : Systemparameter (p, a, b, B, n)

1 Wähle eine Primzahl p mit Bitlänge l.

2 Wähle (zufällig) die Kurvenparameter $a, b \in \mathbb{F}_p$, welche die elliptische Kurve $E(\mathbb{F}_p)$ definieren. Falls $\Delta = 0$, gehe zu Schritt 1.

3 Berechne $N = |E(\mathbb{F}_p)|$.

4 Prüfe, dass N durch eine große Primzahl n ($n > 2^{160}$ und $n > 4\sqrt{p}$) teilbar ist. Falls nicht, gehe zu Schritt 1.

5 Prüfe, dass $p^k - 1$ für jedes $k \in [1, 20]$ nicht durch n teilbar ist. Falls nicht, gehe zu Schritt 1.

6 Prüfe, dass $n \neq p$. Falls nicht, gehe zu Schritt 1.

7 Wähle einen beliebigen Punkt $B' \in E(\mathbb{F}_p)$ und setzte $B = (N/n)B'$. Falls $B = \infty$ gehe zu Schritt 6.

8 **return** (p, a, b, B, n)

Die Wahl der zufälligen Kurvenparameter lässt sich so realisieren, dass ein anderer Teilnehmer verifizieren kann, ob a und b tatsächlich zufällig erzeugt wurden. Diese Nachprüfbarkeit stellt sicher, dass nicht absichtlich schwache Kurven eingeschleust werden. Für eine Beschreibung dieses Konzepts sei auf [JMV01] verwiesen.

6.1.2 Schlüsselpaare

So wie die Sicherheit des privaten Schlüssels beim DSA auf dem DLP in $\mathbb{Z}_p^*$ basiert, basiert sie beim ECDSA auf dem ECDLP. Der private Schlüssel ist also der diskrete Logarithmus d des öffentlichen Schlüssels A. Das heißt der Punkt A ist das d-fache Vielfache des Basispunkts B. Da $\langle B \rangle$ eine zyklische Gruppe der Ordnung n ist, muss d aus dem Intervall $[1, n-1]$ gewählt werden.

Algorithmus 11 : ECDSA-Schlüsselgenerierung

Input : Systemparameter (p, a, b, B, n)

Output : Öffentlicher Schlüssel A, privater Schlüssel d

1 Wähle ein zufälliges $d \in [1, n-1]$.
2 Berechne $A = dB$.
3 **return** (A, d)

6.1.3 Signatur

Eine Signatur (r, s) für eine Nachricht m wird wie folgt berechnet:

Algorithmus 12 : ECDSA-Signaturerzeugung

Input : Systemparameter (p, a, b, B, n), privater Schlüssel d, Nachricht m

Output : Signatur (r, s)

1 Wähle ein zufälliges $k \in [1, n-1]$.
2 Berechne $R = (x_R, y_R) = kB$.
3 Berechne $r = x_R \mod n$.
4 **if** $r = 0$ **then**
5 Gehe zu Schritt 1.
6 Berechne $h = H(m)$.
7 Berechne $s = k^{-1}(h + dr) \mod n$.
8 **if** $s = 0$ **then**
9 Gehe zu Schritt 1.
10 **return** (r, s)

6.1.4 Verifikation

Algorithmus 13 beschreibt die Verfikation der Signatur (r, s) einer Nachricht m.

Algorithmus 13 : ECDSA-Signaturverifikation

Input : Systemparameter (p, a, b, B, n), öffentlicher Schlüssel A, Nachricht m, Signatur (r, s)

Output : `true`, wenn die Signatur akzeptiert wird, `false`, wenn sie abgelehnt wird

1 **if** $r, s \notin [1, n-1]$ **then**
2 **return** `false`
3 Berechne $h = H(m)$.
4 Berechne $w = s^{-1} \mod n$.
5 Berechne $u_1 = hw \mod n$ and $u_2 = rw \mod n$.
6 Berechne $R' = (x_{R'}, y_{R'}) = u_1 B + u_2 A$.
7 **if** $R' = \infty$ **then**
8 **return** `false`
9 Berechne $r' = x_{R'} \mod n$.
10 **if** $r = r'$ **then**
11 **return** `true`
12 **else**
13 **return** `false`

Dass eine korrekte Signatur (r, s) erfolgreich verifiziert wird, macht folgende Umformung deutlich:
Da $s = k^{-1}(h + dr) \mod n$, gilt

$$k = s^{-1}(h + dr) = s^{-1}h + s^{-1}dr = wh + wrd = u_1 + u_2 d \mod n.$$

Daraus folgt

$$R' = u_1 B + u_2 A = (u_1 + u_2 d)B = kB = R.$$

Da r' aus der x-Koordinate von R' und r aus der x-Koordinate von R hervorgeht, ist damit auch $r = r'$ erfüllt.

6.2 Sicherheitsanalyse

Die Sicherheit des ECDSA beruht auf dem ECDLP, also der Annahme, dass es schwer ist für ein gegebenes kB den Skalar k zu finden. Es ist offensichtlich, dass das Lösen des ECDLP ein vollständiges Brechen des ECDSA zur Folge hat. Es konnte jedoch nicht bewiesen werden, dass die Sicherheit des ECDSA äquivalent zu der des ECDLP ist [Wes15]. Versuche die EUF-CMA-Sicherheit des ECDSA zu beweisen, hatten bisher nur unter stärkeren Annahmen oder für Varianten des ECDSA Erfolg (siehe [Bro00] und [PS96]).

Die Sicherheit des ECDSA kann durch Lösen des ECDLP gebrochen werden. Angriffe auf das ECDLP haben wir in Kapitel 5.6 betrachtet.
Ein weiterer Angriffspunkt ist die Zufallszahl k. Für diese gelten die gleichen Sicherheitsanforderungen wie für den privaten Schlüssel d. Wenn ein Angreifer k ermitteln kann, so kann er $d = r^{-1}(ks - h) \mod n$ berechnen. k muss also sicher generiert, gespeichert und gelöscht werden.
Ebenso kann, falls das gleiche k für mehrere Signaturen verwendet wird, der private Schlüssel ermittelt werden. Seien (r, s_1) und (r, s_2) zwei Signaturen für verschiedene Nachrichten m_1 und m_2. Dann ist

$$s_1 = k^{-1}(h_1 + dr) \mod n \quad \text{und } s_2 = k^{-1}(h_2 + dr) \mod n$$

mit $h_1 = H(m_1)$ und $h_2 = H(m_2)$. Durch Multiplikation mit k erhält man

$$ks_1 = h_1 + dr \mod n \quad \text{und } ks_2 = h_2 + dr \mod n.$$

Eine Subtraktion dieser Gleichungen ergibt $k(s_1 - s_2) = h_1 - h_2 \mod n$. Mit hoher Wahrscheinlichkeit gilt $s_1 \neq s_2 \mod n$, sodass

$$k = (h_1 - h_2)(s_1 - s_2)^{-1} \mod n$$

berechnet werden kann. Folglich kann ein Angreifer k und somit auch d ermitteln.
Ein Angriff dieser Art führte 2010 dazu, dass der private Schlüssel der Sony

PlayStation 3 bekannt wurde [BMSS10].

Der ECDSA kann außerdem durch das Ausnutzen von Schwächen der verwendete Hashfunktion gebrochen werden. Ist es einem Angreifer möglich, zwei verschiedene Nachrichten m und m' zu finden, für die gilt $H(m) = H(m')$, so bezeichnet man die Hashfunktion als nicht kollisionsresistent. Verfügt ein Angreifer über solche Nachrichten m und m' und kann er die Nachricht m signieren lassen, so ist diese Signatur auch für m' gültig. Der Angreifer hat somit eine existentielle Fälschung unter einer Chosen-Message-Attacke erreicht.

Kapitel 7

EdDSA

2012 wurde der Edwards-curve Digital Signature Algorithm (EdDSA) von Bernstein et al. in [BDL+12] veröffentlicht. Das Signaturverfahren ist eine Variante des Schnorr-Signaturverfahrens (s. Abschnitt 4.4), die die Verwendung elliptischer Kurven unterstützt.
Das Verfahren nutzt die schnelle Addition auf Twisted-Edwards-Kurven. Dadurch ist das Verfahren deutlich effizienter als der ECDSA [BDL+12].

7.1 Das Signaturverfahren

7.1.1 Systemparameter

Im Gegensatz zum ECDSA verwendet der EdDSA keine Weierstraß-Kurven sondern Twisted-Edwards-Kurven. Dadurch ergeben sich einige Unterschiede bezüglich der Systemparameter. Folgende Parameter werden benötigt:

1. Ein Sicherheitsparameter $l \geq 10$.

2. Eine kryptographische Hashfunktion $H : \{0,1\}^* \to \{0,1\}^{2l}$.

3. Ein Primkörper $\mathbb{F}_p$ der Ordnung p. p ist eine Primzahl, für die gilt $p = 1 \mod 4$ und $p < 2^l$. Die Elemente aus $\mathbb{F}_p$ werden durch einen Bitstring der Länge $l - 1$ ($(l-1)$-Bit-Codierung) dargestellt.

4. Der Koeffizient $d \in \mathbb{F}_p$, der die Gleichung der Twisted-Edwards-Kurve definiert. Die Kurvengleichung lautet

$$E : -x^2 + y^2 = 1 + dx^2y^2. \tag{7.1}$$

   Für d gilt $d \notin \{0, -1\}$ und d ist kein Quadrat in $\mathbb{F}_p$.

5. Ein Basispunkt $B \in E(\mathbb{F}_p)$, wobei B verschieden vom neutralen Element $(0, 1)$ ist.

6. Eine Primzahl n mit $2^{l-4} < n < 2^{l-3}$ für die gilt $nB = (0, 1)$.

Die Kurvengleichung E ist die einer Twisted-Edwards-Kurve mit den Parametern $a = -1$ und d. Da -1 ein Quadrat in $\mathbb{F}_p$ ist, ist die Kurve isomorph zu einer Edwards-Kurve. Setzt man a in die Additionsvorschrift für Twisted-Edwards-Kurve ein, so erhält man:

$$(x_P, y_P) + (x_Q, y_Q) = (\frac{x_Py_Q + y_Px_Q}{1 + dx_Px_Qy_Py_Q}, \frac{y_Py_Q + x_Px_Q}{1 - dx_Px_Qy_Py_Q}).$$

Die Addition ist vollständig, da a ein Quadrat und d kein Quadrat in $\mathbb{F}_p$ ist.

Ein Punkt (x, y) wird als l-Bit-String $(\underline{x, y})$ codiert, bestehend aus der $(l-1)$-Bit-Codierung von y und einem Vorzeichen-Bit für x. Das Vorzeichen-Bit ist 1, falls x negativ ist. Da $x = \pm\sqrt{(y^2 - 1)/(dy^2 + 1)}$, kann x mittels y und dem Vorzeichen eindeutig bestimmt werden. Wir bezeichnen ein Element in $\mathbb{F}_p$ als negativ, falls die $(l-1)$-Bit-Darstellung von x lexikographisch größer ist als die von $-x$. Wir betrachten dabei zuerst das kleinstwertigste Bit (Little-Endian). Für $x = 3 \in \mathbb{F}_{17}$ ist $-x = 14 \in \mathbb{F}_{17}$ und da 00011 > 01110, ist 3 negativ. Da p ungerade ist, sind $\{1, 3, 5, ..., p-2\}$ die negativen Elemente in $\mathbb{F}_p$.

Als Systemparameter empfehlen Bernstein et al. eine spezielle Twisted-Edwards-Kurve, die isomorph zur Curve25519 ist, mit festem Basispunkt B und festem n. Unter Verwendung dieser Kurve heißt das Signaturverfahren Ed25519. Auf die besonderen Eigenschaften von Ed25519 werden wir in Abschnitt 7.3 eingehen.

Ein Algorithmus zur Bestimmung sicherer Systemparameter ist für den EdDSA nicht angegeben. Da Weierstraß-Kurven sich jedoch unter bestimmten Bedingungen in isomorphe Twisted-Edwards-Kurven überführen lassen, kann auch hier Algorithmus 10 zur Erstellung der Systemparameter verwendet werden. Danach kann man mit den in Kapitel 5.5 gegebenen Transformationen die Kurvengleichung und den Basispunkt in passende Parameter für den EdDSA übertragen. Da jedoch nicht für jede Weierstraß-Kurve eine isomorphe Twisted-Edwards-Kurve existiert, braucht man dazu unter Umständen mehrere Durchläufe des Algorithmus.

7.1.2 Schlüsselpaare

Ein Schlüsselpaar des EdDSA besteht aus dem privaten Schlüssel a und dem öffentlichen Schlüssel $\underline{A}$. Zusätzlich existiert ein geheimer Bitstring b. Zur Generierung der Schlüssel wird wie folgt vorgegangen:

Algorithmus 14 : EdDSA-Schlüsselgenerierung

Input : Systemparameter (p, d, B, n), Sicherheitsparameter l

Output : Öffentlicher Schlüssel $\underline{A}$, privater Schlüssel a, geheimer Bitstring b

1 Wähle ein zufälliges z mit Bitlänge l.
2 Berechne $H(z) = (h_0, h_1, ..., h_{2l-1})$.
3 Setze $b = (h_l, ..., h_{2l-1})$.
4 Berechne $a = 2^{l-2} + \Sigma_{i=3}^{l-3} 2^i h_i \in \{2^{l-2}, 2^{l-2} + 8, ..., 2^{l-1} - 8\}$.
5 Berechne $A = aB$ und codiere A als $\underline{A}$.
6 **return** $(\underline{A}, a, b)$

7.1.3 Signatur

Die Signatur $(\underline{R}, \underline{s})$ einer Nachricht m wird wie folgt generiert:

Algorithmus 15 : EdDSA-Signaturerzeugung

Input : Systemparameter (p, d, B, n), öffentlicher Schlüssel $\underline{A}$, privater Schlüssel a, geheimer Bitstring b, Nachricht m

Output : Signatur $(\underline{R}, \underline{s})$

1 Berechne $k = H(b, m)$.

2 Berechne $R = kB$ und codiere R als $\underline{R}$.

3 Berechne $s = (k + H(\underline{R}, \underline{A}, m) \cdot a) \mod n$ und codiere s als l-Bit-String $\underline{s}$.

4 **return** $(\underline{R}, \underline{s})$

Die Signatur $(\underline{R}, \underline{s})$ ist demzufolge ein String mit $2l$ Bit.

7.1.4 Verifikation

Um die Signatur $(\underline{R}, \underline{s})$ einer Nachricht m zu verifizieren, sind folgende Schritte nötig:

Algorithmus 16 : EdDSA-Signaturverifikation

Input : Systemparameter (p, d, B, n), öffentlicher Schlüssel $\underline{A}$, Nachricht m, Signatur $(\underline{R}, \underline{s})$

Output : `true`, wenn die Signatur akzeptiert wird, `false`, wenn sie abgelehnt wird

1 Decodiere $\underline{A}, \underline{R}$ und $\underline{s}$.

2 **if** $A \notin E(\mathbb{F}_p) \ \vee \ R \notin E(\mathbb{F}_p) \ \vee \ s \notin \{0, 1, ..., n-1\}$ **then**

3 **return** `false`

4 Berechne $H(\underline{R}, \underline{A}, m)$.

5 **if** $8sB = 8R + 8H(\underline{R}, \underline{A}, m) \cdot A \in E(\mathbb{F}_p)$ **then**

6 **return** `true`

7 **else**

8 **return** `false`

Um zu zeigen, dass eine korrekte Signatur erfolgreich verifiziert wird, multiplizieren wir die Gleichung $s = (k + H(\underline{R}, \underline{A}, m) \cdot a) \mod n$ mit dem Basispunkt B. Da $nB = (0, 1)$, ergibt sich daraus

$$sB = kB + H(\underline{R}, \underline{A}, m) \cdot aB = R + H(\underline{R}, \underline{A}, m) \cdot A. \tag{7.2}$$

Da $2^{l-4} < n < 2^{l-3}$, sind die drei höchstswertigsten Bits von s immer Null. Wir können daher zusätzlich die Gleichung mit 8 multiplizieren, ohne das wir dadurch an Sicherheit verlieren. Statt der Gleichung in Zeile 5 kann jedoch auch Gleichung 7.2 verifiziert werden.

7.2 Anmerkungen zum Design

Um die Besonderheiten des EdDSA zu verdeutlichen, vergleichen wir das Verfahren mit dem in Kapitel 4.4 vorgestellten Schnorr-Signaturen. Da das Schnorr-Verfahren als Vorlage für das Design des EdDSA diente, gleichen sich die Verfahren in wesentlichen Punkten.
Tabelle 7.1 zeigt den EdDSA im Vergleich mit dem Schnorr-Signaturverfahren. Die Notation ist dabei der des EdDSA angepasst.

Auch wenn der EdDSA Gruppen basierend auf elliptischen Kurven verwendet, so sind die Rahmenbedingungen von EdDSA- und Schnorr-Signaturen sehr ähnlich. In beiden Fällen werden durch die Systemparameter zyklische Untergruppen mit Primzahlordnung definiert.

Der erste wesentliche Unterschied, der beiden Verfahren liegt in der Erzeugung der Schlüssel. Beim Schnorr-Verfahren wird eine zufällig generierte Zahl a als privater Schlüssel verwendet. Um den öffentlichen Schlüssel zu berechnen, wird der Erzeuger g mit a potenziert.
Auch beim EdDSA wird zunächst eine Zufallszahl z generiert. Diese wird jedoch nicht direkt zum Berechnen des öffentlichen Schlüssels und zum Signieren verwendet. Stattdessen werden aus dem Hash $H(z) = (h_0, h_1, ..., h_{2l-1})$ von z zwei weitere geheime Schlüssel abgeleitet.
Aus den Bits h_3 bis h_{l-3} des Hashs wird der private Schlüssel a bestimmt. Dieser wird zum Berechnen des öffentlichen Schlüssels A genutzt, $A = aB$.
Das Bestimmen von a durch den Hash von z sorgt dafür, dass a eine angemessene Zufälligkeit hat [Ber14].

Der zweite Teil des Hashs $(h_l, ..., h_{2l-1})$ ist ein geheimer Bitstring b, der beim

EdDSA	Schnorr
Systemparameter	
Gruppe $E(\mathbb{F}_p)$	Gruppe $\mathbb{Z}_p^*$
Punkt $B \in E(\mathbb{F}_p)$	Element $g \in \mathbb{Z}_p^*$
Untergruppe $G = \langle B \rangle \subset E(\mathbb{F}_p)$	Untergruppe $G = \langle g \rangle \subset \mathbb{Z}_p^*$
Ordnung $n = \|G\| = ord(B)$	Ordnung $n = \|G\| = ord(g),\ n \| p-1$
Hashfunktion $H : \{0,1\}^* \to \{0,1\}^{2l}$	Hashfunktion $H : \{0,1\}^* \to \mathbb{Z}_n$
Schlüsselpaare	
Zufallszahl z	Zufallszahl a
$H(z) = (h_0, h_1, ..., h_{2l-1})$	
$b = (h_l, ..., h_{2l-1})$	
$a = 2^{l-2} + \Sigma_{i=3}^{l-3} 2^i h_i$	
Geheimer Bitstring b	
Privater Schlüssel a	Privater Schlüssel a
Öffentl. Schlüssel $A = aB$	Öffentl. Schlüssel $A = g^a \mod n$
Signatur	
$k = H(b, m)$	Zufallszahl k
$R = kB$	$r = g^k \mod n$
$s = (k + H(\underline{R}, \underline{A}, m)a) \mod n$	$s = (k + H(r, m)a) \mod n$
	$h = H(r, m)$
Signatur $(\underline{R}, \underline{s})$	Signatur (h, s)
Verifikation	
$sB \stackrel{?}{=} R + H(\underline{R}, \underline{A}, m)A$	$h \stackrel{?}{=} H((g^s A^{-h} \mod p), m)$

Tabelle 7.1: Vergleich von EdDSA und Schnorr-Signaturverfahren

Signieren verwendet wird. Hier liegt der zweite wichtige Unterschied der Signaturverfahren.

Beim EdDSA wird k nicht zufällig erzeugt, sondern basiert auf dem Hash des geheimen Bitstrings b und der Nachricht m. k ist also immer eindeutig für eine Nachricht m und den geheimen Bitstring b.

Das Schnorr-Verfahren erzeugt zunächst eine Zufallszahl k. Dabei ist es wichtig, dass für jede neue Nachricht m eine neue, geheime Zufallszahl verwendet wird.

Falls, durch Fehler in der Implementierung, das gleiche k für das Signieren mehrerer verschiedener Nachrichten genutzt wird, kann ein Angreifer so wie auch beim ECDSA den private Schlüssel ermitteln.

Da k beim EdDSA deterministisch in Abhängigkeit der Nachricht m und dem

geheimen Bitstring bestimmt wird, ist ein solcher Angriff hier nicht möglich. Für verschiedene Nachrichten ist, aufgrund der Verwendung einer kryptographischen Hashfunktion, mit hoher Wahrscheinlichkeit auch k verschieden. Außerdem ist hierbei kein sicherer Zufallszahlengenerator notwendig. Ein solcher wird nur einmalig für die Erzeugung des privaten Schlüssels und des geheimen Bitstrings gebraucht. Bei ECDSA und Schnorr hingegen kann auch die Verwendung eines schwachen Zufallszahlengenerators für die Generierung von k zum Bruch des Signaturverfahrens führen.

Während für die Berechnung von s beim EdDSA der Hash von $\underline{R}, \underline{A}$ und m gebildet wird, wird bei Schnorr nur R und m gehasht. Das Einschließen des öffentlichen Schlüssels in den Hash, führt zu einem zusätzlichen Schutz vor Angriffen, die Schwächen der Hashfunktion ausnutzen. Ein Angreifer, der viele Hashs berechnet, um damit eine Signatur zu fälschen, muss seinen Angriff somit auf einen einzigen öffentlichen Schlüssel beschränken. Damit wird, ohne großen zusätzlichen Rechenaufwand, ein simultaner Angriff auf mehrere öffentliche Schlüssel erschwert. Dieser Schutzmechanismus wird relevant, falls es einem Angreifer gelingt, zu einer Nachricht m und entsprechendem Hash $H(\underline{R}, \underline{A}, m)$ eine weitere Nachricht m' zu finden, sodass $H(\underline{R}, \underline{A}, m') = H(\underline{R}, \underline{A}, m)$. Man bezeichnet dies als Zweites-Urbild-Angriff (*second preimage attack*) auf H.

Auch die Komponenten der Signatur unterscheiden sich bei EdDSA und dem Schnorr-Verfahren. Eine Schnorr-Signatur besteht aus dem Hash $h = H(R, m)$ und s. Eine Signatur beim EdDSA besteht hingegen aus $\underline{R}$ und $\underline{s}$. Das Schnorr-Verfahren wurde hier aus zwei Gründen angepasst. Zum einen hat ein Hash beim EdDSA die Länge $2l$, während der codierter Punkt $\underline{R}$ nur l Bit lang ist. $\underline{s}$ ist mit l Bit codiert. Durch das ersetzen des Hashs durch $\underline{R}$ wird die Signatur also um ein Drittel kürzer.
Der zweite Grund ist, dass die Übermittlung von $\underline{R}$ in der Signatur Batch Verifikation erlaubt. Batch Verfikation bietet die Möglichkeit, eine Menge von Signaturen kombiniert zu verifizieren. Damit kann die Verifikation schneller durchgeführt werden, als wenn jede der Signaturen einzeln verifiziert wird.

Auf diesen Aspekt werden wir jedoch nicht genauer eingehen.
Das Anpassen der Signaturkomponenten hat keinen Einfluss auf die Sicherheit. Auch bei Schnorr-Signaturen ist R nicht geheim, denn R lässt sich mittels h und s berechnen: $R = g^s A^{-h} \mod p$.
Die Unterschiede im Verifikationsalgorithmus ergeben sich durch die veränderte Signatur.

7.3 Ed25519

Die Kurve Curve25519 ist definiert über dem Primkörper $\mathbb{F}_p$ mit $p = 2^{255} - 19$ und hat die Kurvengleichung

$$E_M : y^2 = x^3 + 486662x^2 + x$$

in Montgomery-Form.
Bernstein et al. empfehlen die Verwendung des EdDSA mit einer zur Curve25519 isomorphen Twisted-Edwards-Kurve. Das Signaturverfahren trägt in diesem Fall den Namen *Ed25519*.
Um die Curve25519 in eine Twisted-Edwards-Kurve zu transformieren, gehen wir wie folgt vor. Dazu verwenden wir die in Kaptiel 5.5 beschriebenen Abbildungen und erhalten die Twisted-Edwards-Kurve

$$E_{TEd} : 486664x^2 + y^2 = 1 + 486660x^2y^2. \tag{7.3}$$

Diese Transformation lässt sich anwenden, da $486662 + 2$ ein Quadrat und $486662 - 2$ kein Quadrat in $\mathbb{F}_p$ ist[1]. Die resultierende Kurve hat jedoch nicht die für den EdDSA geforderte Form ($a = -1$). Da 486664 ein Quadrat in $\mathbb{F}_p$ ist, können wir die Kurve 7.3 in eine über $\mathbb{F}_p$ isomorphe Edwards-Kurve überführen:

$$E_{Ed} : x^2 + y^2 = 1 + (121665/121666)x^2y^2. \tag{7.4}$$

[1]Dies lässt sich mit dem Legendre-Symbol (siehe [CF06, S. 36]) überprüfen.

Da -1 ein Quadrat in $\mathbb{F}_p$ ist[2], lässt sich die Edwards-Kurve 7.4 wiederum in folgende Twisted-Edwards-Kurve umwandeln:

$$E_{TEd} : -x^2 + y^2 = 1 - (121665/121666)x^2y^2. \tag{7.5}$$

Für die Systemparameter des Ed25519 gilt also $l = 265$, $p = 2^{255} - 19$, $d = -121665/121666$. Als Hashfunktion wird SHA-512 verwendet. Als Basispunkt wird der Punkt $B = (x, 4/5)$ verwendet, wobei x positiv ist. Die Ordnung von B ist $n = 2^{252} + 27742317777372353535851937790883648493$.

Neben den Kurvenparametern beschreiben Bernstein et al. auch eine Skalarmultiplikation für den EdDSA, die nicht nur effizient sondern auch resistent gegenüber bekannten Seitenkanalangriffen ist [Ber14].
Für das Signieren einer Nachricht m wird $R = kB$ berechnet, wobei B der Basispunkt und r ein Hash basierend auf m und dem geheimen Bitstring $(h_l, ..., h_{2l-1})$ ist.
Um kB zu bestimmen, wird wie folgt vorgegangen. Zunächst berechnen wir $k \mod n$ und reduzieren damit k auf eine Zahl mit 253 Bit. $k \mod n$ wird dann dargestellt als die Summe

$$k_0 + 16k_1 + ... + 16^{63}k_{63} \text{ mit}$$
$$k_i \in \{0, 1, ..., 15\},\ 0 \leq i \leq 63.$$

Für jedes i entnehmen wir $16^i k_i B$ einem vorab berechneten Array A_i, das die Punkte $16^i 0B, 16^i 1B, ..., 16^i 15B$ enthält. Danach können wir kB berechnen

$$kB = \Sigma_{0 \leq i \leq 63}\ 16^i k_i B.$$

Falls ein Angreifer Kenntnis über den Skalar k erlangt, so kann er den privaten Schlüssel a berechnen: $a = (s - k)/H(\underline{R}, \underline{A}, m) \mod n$.
Daher ist es wichtig sicherzustellen, dass ein Angreifer k nicht über Seitenkanalangriffe ermitteln kann. Dies ist möglich, wenn k bzw. die Bits von

[2]Dies lässt sich mit dem Legendre-Symbol (siehe [CF06, S. 36]) überprüfen.

k für *If*-Bedingungen oder als Array-Indizes verwendet werden oder wenn in Abhängigkeit von k Operationen mit verschiedenen Laufzeiten ausgeführt werden (siehe Kapitel 5.6.2). Solche Seitenkanalangriffe werden im beschriebenen Algorithmus durch folgende Maßnahmen verhindert:

1. Für die Berechnung von kB erfolgt in jedem Fall eine Reduktion modulo n, 64 Array-Abfragen aus 64 verschiedenen Arrays mit jeweils 16 Einträgen und die Addition von 64 Punkten der Kurve 7.5. Man sieht leicht, dass der Algorithmus eine konstante Laufzeit unabhängig von k hat. Dies gilt insbesondere, da es bei der Addition auf Twisted-Edwards-Kurven keine Fallunterscheidungen oder Ausnahmefälle gibt.

2. Es existieren keine *If*-Bedingungen, die abhängig von k sind.

3. Die naive Herangehensweise, um für ein i den Punkt $16^i k_i B$ aus dem vorab berechneten Array A_i auszulesen, ist das Abfragen von $A_i[k_i]$. In diesem Fall verwenden wir jedoch Teile von k als Array-Indizes und bieten somit einen Angriffspunkt für Seitenkanalangriffe. Um dies zu verhindern, laden wir zunächst das komplette Array A_i und kombinieren die Einträge durch folgende simple Idee:
   Sei a ein Bit. Dann ist $X[a] = X[0] + a(X[0] - X[1])$, wobei an dieser Stelle Addition und Subtraktion einer XOR-Verknüpfung und die die Multiplikation einer AND-Verknüpfung entspricht. So erhalten wir $X[a]$ ohne a direkt als Index eines Arrays zu verwenden. Für zwei Bits a und b gilt $X[2b+a] = Y[0]+b(Y[0]-Y[1])$ mit $Y[0] = X[0]+a(X[0]-X[1])$ und $Y[1] = X[2] + a(X[2] - X[3])$.
   Diese Technik lässt sich auf beliebig viele Bits erweitern. Statt k_i als Index des Arrays A_i zu verwenden, nutzt man also die oben beschriebene Idee und ermittelt so $A_i[8k_{i3} + 4k_{i2} + 2k_{i1} + k_{i0}]$, wobei $k_i = (k_{i3}, k_{i2}, k_{i1}, k_{i0})_2$.

7.4 Sicherheitsanalyse

Auch beim EdDSA beruht die Sicherheit auf der angenommenen Schwierigkeit des ECDLP. Die Schlüssellänge des Ed25519 liegt bei 251 Bit. Unter Verwendung des bisher effizientesten Angriffs auf das ECDLP, den Pollard-Rho-Angriff, werden etwa 2^{125} Schritte benötigt, um den diskreten Logarithmus zu berechnen (siehe Kapitel 5.6.1). Die Curve25519 bzw. eine isomorphe Twisted-Edwards-Kurve wird daher als sicher betrachtet [Ber06]. Eine Pohlig-Hellman-Attacke lässt sich auf den Ed25519 nicht anwenden, da die Ordnung des Basispunkts eine Primzahl ist (siehe Kapitel 5.6.1). Angriffe, die eine effiziente Berechnung von diskreten Logarithmen speziell für diese Kurven ermöglichen, sind bisher nicht bekannt.

Das Schnorr-Signaturverfahren gilt als EUF-CMA-sicher. Pointcheval und Stern haben in [PS96] die Sicherheit im Random-Oracle-Modell unter der Annahme, dass das Berechnen diskreter Lorgarithmen in $\mathbb{Z}_p^*$ schwer ist, bewiesen. Später haben auch Koblitz und Menezes einen solchen Beweis für Schnorr-Signaturen veröffentlicht [KM07].
Im Folgenden werden wir zeigen, dass sich der Beweis von Koblitz und Menezes für das Schnorr-Verfahren auch auf den EdDSA anwenden lässt. Wir wollen also beweisen, dass unter der Annahme, dass das ECDLP schwer ist, der EdDSA im Random-Oracle-Modell EUF-CMA-sicher ist. Dazu werden wir zeigen, dass eine existentielle Fälschung einer EdDSA-Signatur die Berechnung des diskreten Logarithmus des öffentlichen Schlüssels ermöglicht. Zunächst müssen wir jedoch einige Begriffe erläutern:

Random-Oracle-Modell Im Random-Oracle-Modell wird eine Hashfunktion als ein Hashorakel betrachtet, das für jede neue Anfrage eine echte Zufallszahl (bzw. zufällige Bitstrings) zurückgibt. Bei gleicher Anfrage, sind auch die Rückgabewerte identisch. Beweise in diesem Modell zeigen die Sicherheit des Designs eines Signaturverfahrens unter der Annahme, dass die verwendete Hashfunktion ideal ist [PS00].

Existentieller Fälscher Für unseren Beweis nehmen wir an, dass der EdD-

SA nicht EUF-CMA-sicher ist. Es ist also für einen Angreifer, der gültige Signaturen für von ihm gewählte Nachrichten $m_i, 1 \leq i \leq n$, bei einem Signaturorakel anfragen kann, möglich, für eine Nachricht verschieden von m_i eine gültige Signatur zu erzeugen. Einen solchen Angreifer bezeichnen wir als existentiellen Fälscher $\mathcal{F}$. $\mathcal{F}$ ist als ein Programm mit polynomieller Laufzeit zu verstehen, das einen deterministischen Algorithmus ausführt. Die Abläufe des Programms kennen wir nicht. Als Eingabe erhält es einen öffentlichen Schlüssel und die entsprechenden Systemparameter. Es hat Zugriff auf ein Hashorakel H und ein Signaturorakel Σ. Die Ausgabe des Fälschers ist entweder eine gültige Signatur für eine Nachricht, die er zuvor nicht von Σ angefragt hat, oder `false` falls er keine Signatur fälschen konnte.
Wir sprechen von einem probabilistischen existentiellen Fälscher, falls $\mathcal{F}$ eine Sequenz zufälliger Bits zur Verfügung hat, die er beliebig für seine Berechnungen einsetzen kann.

Außerdem benötigen wir folgendes Lemma:

Lemma 1 (Splitting-Lemma, [KM07])**.** *Seien A und B zwei endliche Mengen mit $|A| = a$ und $|B| = b$. Angenommen ϵab aller Paare $(\alpha, \beta) \in A \times B$ haben eine bestimmte Eigenschaft. Wir bezeichnen ein Paar mit dieser Eigenschaft als „gut“. Sei $A_0 \subset A$ definiert als die Menge der Elemente α_0, für die das Paar (α_0, β) (mit festem $\alpha_0 \in A$ und variablem $\beta \in B$) mit Wahrscheinlichkeit $\geq \epsilon/2$ „gut“ ist. Dann sind mindestens $\epsilon a/2$ Elemente in A_0.*

Beweis. Um das Splitting-Lemma zu beweisen, nehmen wir zunächst das Gegenteil an: $|A_0| < \epsilon a/2$. Wir ermitteln nun die Anzahl der guten Paare (α, β) mit $\alpha \in A_0$ und die Anzahl derer mit $\alpha \notin A_0$.
Gemäß unserer Annahme, ist die Anzahl der guten Paare mit $\alpha \in A_0$ höchstens $|A_0|b < (\epsilon a/2)b$.
Aus der Definition von A_0 folgt, dass ein Paar $(\alpha_0, \beta) \in (A \setminus A_0) \times B$ (mit festem $\alpha_0 \in A$ und variablem $\beta \in B$) nur mit Wahrscheinlichkeit $< \epsilon/2$ ein gutes Paar ist. Die Anzahl der guten Paare mit $\alpha \notin A_0$ ist daher höchstens $|A \setminus A_0| \cdot b \cdot \epsilon/2 \leq a \cdot \epsilon b/2$.

Da $A = A_0 \cup (A \setminus A_0)$ die Vereinigung zwei disjunkter Mengen ist, ist die Gesamtanzahl der guten Paare in $A \times B$ echt kleiner als $\epsilon ab/2 + \epsilon ab/2 = \epsilon ab$. Das ergibt einen Widerspruch zu unserer Annahme, dass genau ϵab Paare in $A \times B$ gut sind. Also muss $|A_0| \geq \epsilon a/2$ sein. □

Wir werden folgendes Theorem beweisen:

Theorem 8. *Sei $\mathcal{F}$ ein probabilistischer existentieller Fälscher. Falls $\mathcal{F}$ mit nicht-vernachlässigbarer Wahrscheinlichkeit in polynomieller Zeit eine existentielle Fälschung einer EdDSA-Signatur erreichen kann, so lassen sich diskrete Logarithmen für Twisted-Edwards-Kurven mit Kurvengleichung $E : -x^2 + y^2 = 1 + dx^2y^2$ in polynomieller Zeit berechnen.*

Beweis. Wir nehmen an, es existiert ein probabilistischer existentieller Fälscher $\mathcal{F}$ für EdDSA-Signaturen mit Erfolgswahrscheinlichkeit ϵ. Unter Eingabe eines öffentlichen Schlüssels A, stellt $\mathcal{F}$ Anfragen an das Hashorakel H und das Signaturorakel Σ, greift dabei auf eine gegebene Sequenz zufälliger Bits ω zu und gibt letztendlich mit nicht-vernachlässigbarer Wahrscheinlichkeit ϵ eine gültige Signatur für eine Nachricht, die nicht bei Σ angefragt wurde, aus.

Sei $A = aB$ ein öffentlicher EdDSA-Schlüssel, für den wir den diskreten Logarithmus berechnen wollen. Wir führen $\mathcal{F}$ mit Eingabe A und einer zufälligen Bitfolge ω aus. Bei einer Anfrage auf das Hashorakel H antworten wir mit einem zufälligen Hashwert h. Bei einer Anfrage auf das Signaturorakel Σ wählen wir zufällige Werte h und s und berechnen damit $R = sB - hA$. Dann antworten wir mit der Signatur $(\underline{R}, \underline{s})$. Die Wahrscheinlichkeit, dass die zufälligen Antworten der Orakel zu einem Widerspruch führen, ist gering[3].

[3] Sei $(\underline{R}, \underline{A}, m)$ eine Hashanfrage, die wir mit h beantworten. Später erhalten wir eine Signaturanfrage für die gleiche Nachricht m, die wir mit $(\underline{R}', \underline{s})$ beantworten, wobei R' durch s und einen zufälligen Hash h' bestimmt wurde. Mit hoher Wahrscheinlichkeit gilt $h \neq h' \mod l$. Es kann vorkommen, dass für die beiden Anfragen gilt $\underline{R}' = \underline{R}$. In diesem Fall erhalten wir einen Widerspruch, da wir zwei verschiedene Hashwerte h und h' für gleiche Eingaben $(\underline{R}, \underline{A}, m)$ verwendet haben.
Auch folgendes Szenario kann zu einem Widerspruch führen. Sei m eine Nachricht für die zweimal eine Signatur von Σ angefragt wird. Wir antworten erst mit $(\underline{R}, \underline{s})$, wobei R durch s und einen zufälligen Hash h bestimmt wurde, und später mit $(\underline{R}', \underline{s}')$, wobei R' durch s' und einen zufälligen Hash h' bestimmt wurde. Wieder gilt $h \neq h' \mod l$ mit hoher

Sei q_h die maximale Anzahl an Anfragen, die $\mathcal{F}$ an H stellen darf. Wir wählen vorab einen zufälligen Index $j \leq q_h$. Um eine Signatur zu fälschen, muss $\mathcal{F}$ für die entsprechende Nachricht m und den Punkt R einen Hash $H(\underline{R}, \underline{A}, m)$ anfragen. Wir hoffen, dass Hashanfrage j einen Hash $h_j = H(\underline{R}_j, \underline{A}, m_j)$ anfordert und $\mathcal{F}$ für das Paar (m_j, R_j) eine Signatur fälschen wird ($\mathcal{F}$ könnte allerdings auch eine andere Nachricht fälschen oder `false` ausgeben).
Nach unserer Definition von $\mathcal{F}$, liegt die Wahrscheinlichkeit, dass er eine Signatur für irgendein Paar (m, R), für das es $H(\underline{R}, \underline{A}, m)$ anfragt, fälscht, bei ϵ. Demnach ist die Wahrscheinlichkeit, dass er eine Signatur für (m_j, R_j) fälscht, mindestens ϵ/q_h.

Unser Ziel ist es, zwei gültige Signaturen für (m_j, R_j) zu finden, für die $h_j \neq h'_j$ mod l und damit $\underline{s}_j \neq \underline{s}'_j$ gilt. Da $(\underline{R}_j, \underline{s}_j)$ und $(\underline{R}_j, \underline{s}'_j)$ gültige Signaturen sind, können sie erfolgreich verifiziert werden. Also gilt

$$\begin{aligned} s_j B &= R_j + h_j A \text{ und} \\ s'_j B &= R_j + h'_j A. \end{aligned}$$

Subtraktion der beiden Gleichungen ergibt

$$\begin{aligned} s_j B - s'_j B &= R_j - R_j + h_j A - h'_j A \\ (s_j - s'_j) B &= (h_j - h'_j) A \\ (s_j - s'_j)(h_j - h'_j)^{-1} B &= A. \end{aligned}$$

Daraus ergibt sich der diskrete Logarithmus des öffentlichen Schlüssel $A = aB$:

$$a = (s_j - s'_j)(h_j - h'_j)^{-1} \mod l.$$

Wahrscheinlichkeit. Es kann vorkommen, dass $\underline{R} = \underline{R}'$. In diesem Fall erhalten wir einen Widerspruch, da wir zwei verschiedene Hashwerte h und h' für gleiche Eingaben $(\underline{R}, \underline{A}, m)$ verwendet haben.
Falls eines dieser Szenarien eintritt, müssen wir den Angriff von neuem beginnen, da die Erfolgswahrscheinlichkeit ϵ von $\mathcal{F}$ dadurch nicht mehr garantiert ist. Die Wahrscheinlichkeit, dass diese Fälle eintreten, ist jedoch sehr gering.

Nun stellt sich die Frage, wie wir solche Signaturen finden können. Dazu verwenden wir das sogenannte *Forking-Lemma*.

Lemma 2 (Forking-Lemma [PS96])**.** *Sei $\mathcal{F}$ ein probabilistischer existentieller Fälscher für EdDSA-Signaturen. Falls $\mathcal{F}$ mit nicht-vernachlässigbarer Wahrscheinlichkeit eine gültige Signatur $(\underline{R}, \underline{s})$ finden kann, dann führt ein Wiederholen des Angriffs mit verschiedenen Zufallsbits und Orakelantworten mit nicht-vernachlässigbarer Wahrscheinlichkeit zu zwei Signaturen $(\underline{R}, \underline{s})$ und $(\underline{R}, \underline{s}')$ mit $h \neq h'$.*

Wir erstellen eine Kopie unseres Fälschers $\mathcal{F}$, $\mathcal{F}'$. Dieser bekommt die gleichen Eingaben wie $\mathcal{F}$, also den öffentlichen Schlüssel A und die zufällige Bitfolge ω. Da $\mathcal{F}'$ eine Kopie von $\mathcal{F}$ ist, verhalten sich die Programme bei gleicher Eingabe und gleichen Antworten von H und Σ genau gleich. Wir werden $\mathcal{F}$ und $\mathcal{F}'$ bis zur Anfrage j die selben Antworten auf ihre Hashanfragen geben. Für Anfrage j und die darauffolgenden Hashanfragen erhält $\mathcal{F}'$ andere Rückgabewerte von H als $\mathcal{F}$. Auch die Antworten von Σ und die Sequenz von zufälligen Bits für $\mathcal{F}'$ sind ab diesem Punkt unabhänging von denen, die $\mathcal{F}$ erhält.
Wir hoffen, dass beide Fälscher eine Signatur für (m_j, R_j) erzeugen. R_j ist für beide Signaturen gleich, da R_j auf jeden Fall vor der Anfrage für $H(\underline{R}_j, \underline{A}, m_j)$ bestimmt wird und sich $\mathcal{F}$ und $\mathcal{F}'$ bis zu diesem Zeitpunkt gleich verhalten. Mit hoher Wahrscheinlichkeit gilt dann für beide Signaturen $h_j \neq h'_j \mod l$ und wir können a berechnen. Wir werden nun ermitteln, mit welcher Wahrscheinlichkeit dieses Ereignis eintritt.

Sei A die Menge aller möglichen zufälligen Bits und Antworten der Orakel, die die Fälscher erhalten, bevor beide die Anfrage j an H stellen. Sei B die Menge aller möglichen zufälligen Bits und Antworten der Orakel ab dieser Anfrage. Weiter sei S die Menge aller möglichen zufälligen Bits und Antworten der Orakel für die gesamte Prozedur. Demnach gilt $S = A \times B$. Außerdem sei $J = \{1, ..., q_h\}$. Für jedes $j \in J$ bezeichne ϵ_j die Wahrscheinlichkeit, dass eine Sequenz $s \in S$ zu einer Fälschung der Signatur für (m_j, R_j) führt. Gemäß unserer Annahme ist $\Sigma_{j \in J}\ \epsilon_j = \epsilon$, wobei ϵ die Erfolgswahrscheinlichkeit des

Fälschers bezeichnet.
Nehmen wir an, dass a die Anzahl der Elemente in A und b die Anzahl der Elemente in B ist. Demnach ist $|S| = ab$. Es gibt also $\epsilon_j ab$ Sequenzen $s \in S$ für die ein Fälscher eine Signatur für (m_j, R_j) erzeugt.
Unter Verwendung des Splitting-Lemmas können wir nun schließen, dass es mindestens $\epsilon_j a/2$ Elemente in A gibt, die folgende Eigenschaft haben: die übrigen Schritte (angefangen mit der Anfrage j an H) des Fälschers führen mit einer Wahrscheinlichkeit $\epsilon_j/2$ zu einer Fälschung der Signatur für (m_j, R_j). Für jedes dieser Elemente aus A ist die Wahrscheinlichkeit, dass beide Fälscher eine Signatur für (m_j, R_j) erzeugen, mindestens $(\epsilon_j/2)^2$. Die Wahrscheinlichkeit, dass ein Element aus $A \times (B \times B)$, zu zwei verschiedenen Signaturen für (m_j, R_j) führt, ist daher mindestens $(\epsilon_j/2)^3$.

Da $j \in J$ zufällig gewählt wird, ist die Wahrscheinlichkeit, dass das beschriebene Vorgehen zu zwei verschiedenen Signaturen der gleichen Nachricht führt, mindestens

$$\frac{1}{q_h} \Sigma_{j \in J} \, (\epsilon_j/2)^3 = \frac{1}{8q_h} \Sigma_{j \in J} \, \epsilon_j^3.$$

Die Summe $\Sigma_{j \in J} \, \epsilon^3$ wird minimal, wenn alle ϵ_j gleich sind [4]. Da $\Sigma_{j \in J} \, \epsilon_j = \epsilon$, gilt dann $\epsilon_j = \epsilon/q_h$ für alle $j \in J$.
Die Erfolgswahrscheinlichkeit für eine Iteration dieses Verfahrens ist daher mindestens

$$\frac{1}{8q_h} \cdot q_h \cdot (\epsilon/q_h)^3 = (\epsilon/2q_h)^3.$$

Die Wahrscheinlichkeit, dass wir den privaten Schlüssel nach k Iterationen nicht finden, ist höchstens $(1 - (\epsilon/2q_h)^3)^k$. Bei steigender Anzahl der Iterationen, konvergiert die Wahrscheinlichkeit gegen 0. Damit ist unser Theorem bewiesen. □

Es ist anzumerken, dass dieser Beweis nicht automatisch vor Angriffen in der realen Welt schützt. Setzt man für ϵ und q_h angemessene Werte ein, so ist die notwendige Schlüssellänge unter Umständen nicht für praktische Zwecke

[4] Die lässt sich nachprüfen, indem man (mithilfe der partiellen Ableitungen) das Minimum der Summe $\epsilon_1^3 + \epsilon_2^3 + .. + \epsilon_{q_h-1}^3 + (\epsilon - \epsilon_1 - \epsilon_2 - ... - \epsilon_{q_h-1})^3$ berechnet.

anwendbar.
Sei t die Laufzeit (Anzahl der Schritte), die ein Fälscher $\mathcal{F}$ braucht, um eine Signatur zu fälschen. Sei T eine untere Schranke für die Zeit, die wir brauchen, um damit einen diskreten Logarithmus zu berechnen. Wenn wir unser Vorgehen k-mal wiederholen müssen, um mit hoher Wahrscheinlichkeit den diskreten Logarithmus zu finden, so ist unsere untere Schranke $T \approx kt$. Die Wahrscheinlichkeit, dass unser Angriff scheitert, liegt bei $(1 - (\epsilon/2q_h)^3)^k$. $(1 - (\epsilon/2q_h)^3)^k$ wird klein für ein $k = O(q_h^3/\epsilon^3)$ [5]. Wir sollten also $T \approx tq_h^3/\epsilon^3$ setzen.
Nehmen wir an, die Erfolgswahrscheinlichkeit von $\mathcal{F}$ liegt bei 100%. Die Anzahl der Hashanfragen q_h ist begrenzt durch die Anzahl der Schritte von $\mathcal{F}$. Es gelte also $q_h = t$ und $\epsilon = 1$ und wir erhalten $T \approx t^4$.
Um unter diesen Bedingungen ein Sicherheitsniveau von 100 Bit zu garantieren, müssen wir die Parameter des EdDSA also so wählen, dass mindestens 2^{400} Schritte benötigt werden, um einen diskreten Logarithmus zu berechnen. Unter der Annahme, dass der Pollard-Rho-Angriff der effizienteste Angriff auf den ECDLP ist, müssten wir eine Schlüssellänge von 800 Bits fordern. Aktuell empfohlene Schlüssellängen für die ECC liegen hingegen bei nur 200 Bit für ein Sicherheitsniveau von 100 Bit [BSI16].

Des Weiteren können Implementierungsfehler, Seitenkanalangriffe, Angriffe auf die verwendete Hashfunktion oder verbesserte Angriffe auf das ECDLP, zur Fälschung von Signaturen führen.

[5] Aus der Analysis wissen wir, dass $\forall x \in \mathbb{R} \lim_{n\to\infty}(1+\frac{x}{n})^n = e^x$ [Bla96, S. 161]. Da die Folge von unten gegen e^x strebt, folgt daraus $(1+\frac{x}{n})^n \leq e^x, \forall n \in \mathbb{N}$. Setzen wir $n = q_h^3/\epsilon^3$ und $x = -1/8$, so gilt $(1 - \frac{1}{8} \cdot \frac{\epsilon^3}{q_h^3})^{q_h^3/\epsilon^3} = (1 - (\epsilon/2q_h)^3)^{q_h^3/\epsilon^3} \leq e^{-1/8} \approx 0,882$.

Kapitel 8

Gegenüberstellung von ECDSA und EdDSA

In den vorausgegangen Kapiteln haben wir den ECDSA und den EdDSA kennengelernt, ihre Schwachstellen und Sicherheitsmaßnahmen analysiert. Wir werden nun beide Verfahren im Hinblick auf ihre Sicherheit vergleichen.

Beweisbare Sicherheit Wir konnten zeigen, dass der EdDSA unter der Annahme, dass das ECDLP schwer ist, im Random-Oracle-Modell EUF-CMA-sicher ist.
Ein entsprechender Beweis für den ECDSA existiert nicht. Bisherige Beweise für die Sicherheit des ECDSA basieren auf stärkeren Annahmen.
Das in Kapitel 7.4 beschriebene Verfahren lässt sich auf den ECDSA nicht anwenden. Der Hash $H(m)$ einer Nachricht m ist unabhängig von r. Daher kann ein Fälscher r auch nach dem Anfragen des Hashs bestimmen. Es ist also nicht gewährleistet, dass die zwei Fälscher $\mathcal{F}$ und $\mathcal{F}'$ zwei Signaturen $(r, s), (r', s')$ für m erzeugen, für die gilt $r = r'$. Damit lässt sich der private Schlüssel (bzw. der diskrete Logarithmus) nicht bestimmen.

Hashfunktion Betrachten wir nun die Anforderungen an die Hashfunktion. Für den ECDSA ist es wichtig, dass die verwendete Hashfunktion kollisionsresistent ist. Ist sie es nicht, so kann ein Angreifer zwei Nachrichten m, m' finden, für die gilt $H(m) = H(m')$. Fragt der Angreifer ein Signaturorakel

nach einer Signatur für m, so ist diese gleichzeitig auch für die Nachricht m' gültig. In diesem Fall ist der ECDSA also nicht EUF-CMA-sicher.
Beim EdDSA ist der Hash nicht nur abhängig von der Nachricht m, sondern auch vom Punkt R und dem öffentlichen Schlüssel A. Um eine existentielle Fälschung zu erreichen, muss der Angreifer daher zu einem gegebenen Hash $H(\underline{R}, \underline{A}, m)$ eine Nachricht m' finden, sodass gilt $H(\underline{R}, \underline{A}, m) = H(\underline{R}, \underline{A}, m')$. Wenn ein Angreifer dies erreicht, so ist die Hashfunktion nicht resistent gegen einen Zweites-Urbild-Angriff. Gleichzeitig ist sie dadurch auch nicht kollisionsresistent.
Ein zweites Urbild zu einem gegebenen Hash zu finden, ist schwerer als zwei beliebige Eingaben mit gleichem Hash zu finden. Daher sind die Anforderungen an die Hashfunktion beim ECDSA höher als beim EdDSA.

Zufallszahlengenerator Der Zufallsgenerator wird beim ECDSA verwendet, um den privaten Schlüssel zu generieren und um bei jeder Signaturerzeugung eine neue geheime Zufallszahl k zu generieren. Die Verwendung eines schwachen Zufallszahlengenerators, dessen Werte vorhersagbar oder gar gleich sind, führt zum vollständigen Brechen des Verfahrens (siehe Kapitel 6.2).
Auch beim EdDSA wird zur Schlüsselgenerierung eine Zufallszahl z benötigt. Die Erzeugung von k erfolgt jedoch deterministisch, sodass beim Signieren keine Zufallszahlen nötig sind. Da es sich bei k um einen Hash handelt, der von $H(z)$ und der Nachricht abhängig ist, ist es praktisch nicht möglich, dass für verschiedene Nachrichten das gleiche k verwendet wird.
Ein sicherer Zufallszahlengenerator wird beim EdDSA also lediglich zur Schlüsselgenerierung verwendet. Man kann also Signaturen auch auf Umgebungen ohne Zufallszahlengenerator erstellen, wenn man das Schlüsselpaar vorher extern auf einem Rechner mit sicherem Zufallszahlengenerator erzeugt.

Kurvenauswahl Beim ECDSA können sowohl empfohlene Kurven, die als sicher gelten, als auch zufällige Kurven verwendet werden. Die zufälliger Wahl der Kurvenparameter schützt vor Angriffen auf spezielle Kurven.
Für den EdDSA wird die Curve25519 empfohlen. Ein Algorithmus zur zufälligen Wahl der Kurvenparameter wurde von Bernstein et al. nicht angegeben. Es

kann jedoch der gleiche Algorithmus wie beim ECDSA verwendet werden (siehe Kapitel 7.1.1).

Seitenkanalangriffe Der ECDSA verwendet Weierstraß-Kurven und deren Arithmetik. Die komplizierte Fallunterscheidung bei der Addition auf Weierstraß-Kurven macht sie anfällig für Implementierungsfehler. Dadurch kann das Verfahren angreifbar für Seitenkanalangriffe werden.
Die Addition auf Twisted-Edwards-Kurven, wie sie beim EdDSA verwendet wird, ist aufgrund ihrer Kompaktheit (es gibt keinerlei Fallunterscheidungen) weniger anfällig für Seitenkanalangriffe.
Die Skalarmultiplikation enthält keine Sprünge oder Array-Indizes die unmittelbar von dem Wert eines geheimen Schlüssels abhängen. Auch dies dient der Verhinderung von Seitenkanalangriffen.

Kapitel 9

Fazit und Ausblick

Diese Arbeit beschäftigt sich mit zwei digitalen Signaturverfahren der Elliptische-Kurven-Kryptographie (ECC). Das erste vorgestellte Verfahren ist der Elliptic Curve Digital Signature Algorithm (ECDSA), der bereits 1998 standardisiert wurde. Er wurde nach den Vorbild des DSA entworfen und basiert auf zyklischen Gruppen, erzeugt durch einen Punkt großer Ordnung auf einer Weierstraß-Kurve.
Ein neues Signaturverfahren auf Basis elliptischer Kurven wurde 2012 durch Bernstein et al. veröffentlicht. Anstelle von Weierstraß-Kurven werden hierbei Twisted-Edwards-Kurven verwendet. Das Verfahren wurde dem Schnorr-Signaturverfahren nachempfunden.

Die Arbeit führt zunächst in die Grundlagen der ECC ein und beschreibt die Zusammenhänge von Weierstraß-Kurven und Twisted-Edwards-Kurven.
Daraufhin werden die Verfahren in Hinblick auf ihre Sicherheit untersucht und verglichen. Diese beruht bei beiden Signaturverfahren auf der vermuteten Schwierigkeit des Diskreter-Logarithmus-Problem für elliptische Kurven (ECDLP).
Die Vorteile des EdDSA gegenüber dem ECDSA liegen in den niedrigeren Anforderungen an die verwendete Hashfunktion. Zudem wird kein Zufallszahlengenerator zum Signieren benötigt. Beim Design des EdDSA wurde ein besonderes Augenmerk auf das Verhindern von Seitenkanalangriffen gerichtet.

Zum einen verzichtet die Addition auf Twisted-Edwards-Kurven auf jegliche Fallunterscheidung. Zum anderen enthält die entsprechende Skalarmultiplikation eine Reihe von Maßnahmen, die den Algorithmus resistent gegen Seitenkanalangriffe machen sollen.
Ein weiterer Vorteil ist die beweisbare Sicherheit des Verfahrens. Der Reduktionsbeweis in Kapitel 7.4 zeigt, dass der EdDSA unter Annahme, dass der ECDLP schwer ist, im Random-Oracle-Modell EUF-CMA-sicher ist.

Neben den Sicherheitsaspekten bietet der EdDSA ein effizientes Verfahren zur Batch Verifikation. Der ECDSA ist nicht ohne weiteres zur Batch Verifikation geeignet. Es gibt jedoch bereits Varianten, die auch dies ermöglichen. Ein Vergleich dieser Verfahren im Hinblick auf ihre Anwendung in der Batch Verifikation bietet Anlass für weitere Untersuchungen.

Literaturverzeichnis

[ANS99] ANSI X9.62: *Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA).* American National Standards Institute, 1999.

[AVV+15] Adrian, David, Luke Valenta, Benjamin VanderSloot, Eric Wustrow, Santiago Zanella-Béguelin, Paul Zimmermann, Karthikeyan Bhargavan, Zakir Durumeric, Pierrick Gaudry, Matthew Green, J. Alex Halderman, Nadia Heninger, Drew Springall und Emmanuel Thomé: *Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice.* In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, Seiten 5–17. ACM Press, 2015.

[BBJ+08] Bernstein, Daniel J., Peter Birkner, Marc Joye, Tanja Lange und Christiane Peters: *Twisted Edwards Curves.* In: *Progress in Cryptology–AFRICACRYPT 2008*, Seiten 389–405. Springer, 2008.

[BDL+12] Bernstein, Daniel J., Niels Duif, Tanja Lange, Peter Schwabe und Bo-Yin Yang: *High-speed High-security Signatures.* Journal of Cryptographic Engineering, 2:77–89, 2012.

[Ber05] Bernstein, Daniel J.: *Cache-timing Attacks on AES.* Technischer Bericht, 2005.

[Ber06] Bernstein, Daniel J.: *Curve25519: New Diffie-Hellman Speed Records.* In: Yung, Moti, Yevgeniy Dodis, Aggelos Kiayias

und Tal Malkin (Herausgeber): *Public Key Cryptography - PKC 2006*, Nummer 3958 in *Lecture Notes in Computer Science*, Seiten 207–228. Springer, April 2006.

[Ber14] Bernstein, Daniel J.: *cr.yp.to: 2014.03.23: How to Design an Elliptic-curve Signature System*, März 2014. `http://blog.cr.yp.to/20140323-ecdsa.html`, 17.09.2016.

[BJ02] Brier, Eric und Marc Joye: *Weierstraß Elliptic Curves and Side-Channel Attacks*. In: *International Workshop on Public Key Cryptography*, Seiten 335–345. Springer, 2002.

[BL07] Bernstein, Daniel J. und Tanja Lange: *Faster Addition and Doubling on Elliptic Curves*. In: Kurosawa, Kaoru (Herausgeber): *Advances in Cryptology – ASIACRYPT 2007*, Nummer 4833 in *Lecture Notes in Computer Science*, Seiten 29–50. Springer, Dezember 2007.

[Bla96] Blatter, Christian: *Ingenieur Analysis 1*. Springer-Lehrbuch. Springer, 1996.

[BMSS10] Bushing, Marcan, Segher und Sven: *Console Hacking 2010*, 2010. `https://events.ccc.de/congress/2010/Fahrplan/attachments/1780_27c3_console_hacking_2010.pdf`, 18.10.2016.

[Bro00] Brown, Daniel R. L.: *The Exact Security of ECDSA*. Technischer Bericht CORR 2000-54, Universtiy of Waterloo, Dezember 2000.

[BSI12] BSI: *Elliptic Curve Cryptography*. Technischer Bericht TR-03111, Bundesamt für Sicherheit in der Informationstechnik, Juni 2012.

[BSI16] BSI: *Kryptographische Verfahren: Empfehlungen und Schlüssellängen*. Technischer Bericht TR-02102-1, Bundesamt für Sicherheit in der Informationstechnik, Februar 2016.

[Buc16] Buchmann, Johannes: *Einführung in die Kryptographie*. Springer-Lehrbuch. Springer, 2016.

[CF06] Cohen, Henri und Gerhard Frey: *Handbook of elliptic and hyperelliptic curve cryptography*. Discrete mathematics and its applications. Chapman & Hall/CRC, 2006.

[Edw07] Edwards, Harold: *A Normal Form for Elliptic Curves*. Bulletin of the American Mathematical Society, 44(3):393–422, 2007.

[GMR88] Goldwasser, Shafi, Silvio Micali und Ronald L. Rivest: *A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks*. SIAM Journal on Computing, 17(2):281–308, April 1988.

[Han10] Hanser, Christian: *New Trends in Elliptic Curve Cryptography*. Technische Universität Graz, 2010.

[HVM03] Hankerson, Darrel R., Scott A. Vanstone und A. J. Menezes: *Guide to elliptic curve cryptography*. Springer, 2003.

[IEE00] IEEE Std 1363-2000: *Standard Specifications for Public-Key Cryptography*. Institute of Electrical and Electronics Engineers, 2000.

[ISO98] ISO 14888-3:1998: *Information Technology – Security Techniques – Digital Signatures with Appendix – Part 3: Certificate Based-Mechanisms*. International Organization for Standardization, 1998.

[JMV01] Johnson, Don, Alfred Menezes und Scott Vanstone: *The Elliptic Curve Digital Signature Algorithm (ECDSA)*. International Journal of Information Security, 1(1):36–63, 2001.

[KAF+10] Kleinjung, Thorsten, Kazumaro Aoki, Jens Franke, Arjen K. Lenstra, Emmanuel Thomé, Joppe W. Bos, Pierrick Gaudry, Alexander Kruppa, Peter L. Montgomery, Dag Arne Osvik, Herman te Riele, Andrey Timofeev und Paul Zimmermann: *Factorization of a 768-Bit RSA Modulus*. In: Rabin, Tal (Herausgeber): *Advances in Cryptology – CRYPTO 2010*, Nummer 6223 in *Lecture Notes in Computer Science*, Seiten 333–350. Springer, August 2010.

[KLLT11] Killmann, Wolfgang, Tanja Lange, Manfred Lochter und Wolfgang Thumser: *Minimum Requirements for Evaluating Side-Channel Attack Resistance of Elliptic Curve Implementations.* Technischer Bericht AIS 46, Bundesamt für Sicherheit in der Informationstechnik, Juli 2011.

[KM07] Koblitz, Neal und Alfred J. Menezes: *Another Look at „Provable Security“.* J. Cryptol., 20(1):3–37, Januar 2007.

[Kob87] Koblitz, Neal: *Elliptic Curve Cryptosystems.* Mathematics of Computation, 48(177):203–209, 1987.

[LM10] Lochter, Manfred und Johannes Merkle: *Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation.* RFC 5639. 2010.

[Mil85] Miller, Victor S.: *Use of Elliptic Curves in Cryptography.* In: Williams, Hugh C. (Herausgeber): *Advances in Cryptology — CRYPTO '85 Proceedings*, Nummer 218 in *Lecture Notes in Computer Science*, Seiten 417–426. Springer, August 1985.

[Mon87] Montgomery, Peter L.: *Speeding the Pollard and Elliptic Curve Methods of Factorization.* Mathematics of Computation, 48(177):243–264, 1987.

[MvOV96] Menezes, Alfred J., Paul C. van Oorschot und Scott A. Vanstone: *Handbook of Applied Cryptography.* CRC press, 1996.

[NIS99] NIST: *Recommended Elliptic Curves for Federal Government Use.* National Institute of Standards and Technology, 1999.

[NIS00] NIST FIPS 186-2: *Digital Signature Standard (DSS).* National Institute of Standards and Technology, Januar 2000.

[NIS13] NIST FIPS 186-4: *Digital Signature Standard (DSS).* National Institute of Standards and Technology, Juli 2013.

[OKS00] Okeya, Katsuyuki, Hiroyuki Kurumatani und Kouichi Sakurai: *Elliptic Curves with the Montgomery-form and their Cryptographic Applications*. In: *Public Key Cryptography*, Seiten 238–257. Springer, 2000.

[PP10] Paar, Christof und Jan Pelzl: *Understanding Cryptography*. Springer, 2010.

[PS96] Pointcheval, David und Jacques Stern: *Security Proofs for Signature Schemes*. In: Maurer, Ueli (Herausgeber): *Advances in Cryptology — EUROCRYPT '96*, Nummer 1070 in *Lecture Notes in Computer Science*, Seiten 387–398. Springer, Mai 1996.

[PS00] Pointcheval, David und Jacques Stern: *Security Arguments for Digital Signatures and Blind Signatures*. Journal of cryptology, 13(3):361–396, 2000.

[Sch85] Schoof, Rene: *Elliptic Curves Over Finite Fields and the Computation of Square Roots mod p*, Band 44 der Reihe *Mathemetics of Computation*. American Mathematical Society, 1985.

[Sch90] Schnorr, C. P.: *Efficient Identification and Signatures for Smart Cards*. In: Brassard, Gilles (Herausgeber): *Advances in Cryptology — CRYPTO' 89 Proceedings*, Nummer 435 in *Lecture Notes in Computer Science*, Seiten 239–252. Springer, 1990.

[Sil09] Silverman, Joseph H.: *The Arithmetic of Elliptic Curves*, Band 106 der Reihe *Graduate Texts in Mathematics*. Springer, 2009.

[Sti06] Stinson, Douglas: *Cryptography: Theory and Practice*. Chapman & Hall/CRC, 2006.

[Was08] Washington, Lawrence C.: *Elliptic curves: Number Theory and Cryptography*. Discrete Mathematics and its Applications. Chapman & Hall/CRC, 2008.

[Wer02] Werner, Annette: *Elliptische Kurven in der Kryptographie*. Springer, 2002.

[Wes15] Wesolowski, Michael: *Batch Verification of Elliptic Curve Digital Signatures*. University of Waterloo, April 2015.