ingenieur
wissenschaften
htw saar

Hochschule für
Technik und Wirtschaft
des Saarlandes
University of
Applied Sciences

Projektbericht

Projektarbeit
an der Hochschule für Technik und Wirtschaft des Saarlandes
im Studiengang Praktische Informatik
der Fakultät für Ingenieurwissenschaften

Implementierung elliptischer Kurven für die Kryptographie

vorgelegt von
Annick Aboa
Hendrik Haas
Mai Manh-Khang

betreut und begutachtet von
Prof. Dr. Peter Birkner

Saarbrücken, 31. März 2021

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

In den letzten Jahren hat die dramatische Zunahme von elektronisch übertragenen Informationen zu einer zunehmenden Abhängigkeit von kryptographischen Verfahren geführt. In unserer modernen vernetzten Welt ermöglicht Kryptographie es Menschen, nicht nur geheime Nachrichten über öffentliche Kanäle auszutauschen, sondern auch Online-Banking, Online-Handel und Online-Einkäufe zu tätigen, ohne befürchten zu müssen, dass die persönlichen Informationen kompromittiert werden [5].
Daher ist unter dem Begriff **Kryptographie**, die Lehre mathematischer Techniken in Bezug auf Aspekte der Informationssicherheit wie Vertraulichkeit, Datenintegrität, Datensauthentifizierung, zu verstehen [3]. Die Dringlichkeit eines sicheren Austauschs digitaler Daten zu gewähren, hat daher in den letzten Jahren zu großen Mengen unterschiedlicher Verschlüsselungsverfahren geführt. Diese können in zwei Gruppen eingeteilt werden nähmlich: symmetrische (mit privaten Schlüsselalgorithmen) und asymmetrische Verschlüsselungsverfahren (mit öffentlichen Schlüsselalgorithmen) [3].

In dieser Arbeit wird der Fokus hauptsächlich auf eine asymmetrische Verschlüsselungstechnik liegen: die **Elliptische-Kurven-Kryptographie** (engl. **Elliptic Curve Cryptography: ECC**) aufgrund ihrer zahlreichen Vorteile gegenüber herkömmlichen kryptographischen Algorithmen. Gemäß den Richtlinien des Nationalen Instituts für Standards und Technologie (NIST) kann eine ECC-Schlüsselgröße von 163 Bit eine gleichwertige bzw. höhere Sicherheit wie ein 1024-Bit des RSA-Algorithmus gewährleisten. Mit guten ECC-Schlüsselgrößen sind nur eine geringere Rechenleistung, sowie ein geringerer Speicher- und Stromverbrauch erforderlich ([10]; [32]). Zudem kann die Technologie in Verbindung mit den meisten Verschlüsselungsmethoden mit öffentlichen Schlüsseln wie RSA und Diffie-Hellman verwendet werden. ECC ist ideal für den Einsatz in eingeschränkten Umgebungen wie Personal Digital Assistenten, Mobiltelefonen und Smartcards.

Im Allgemeinen lässt sich die Elliptische-Kurven-Kryptographie als ein Public-Key- bzw. ein asymmetrisches Verschlüssselungsverfahren definieren, das auf der elliptischen Kurventheorie basiert und zur Erstellung schnellerer, kleinerer und effizienterer kryptografischer Schlüssel verwendet werden kann. Im asymmetrischen Verfahren mit öffentlichen Schlüsseln verfügt jeder Benutzer oder das Gerät, das an der Kommunikation teilnimmt, über ein Schlüsselpaar: einen öffentlichen Schlüssel und einen privaten Schlüssel sowie eine Reihe von Operationen, die den Schlüsseln zugeordnet sind, um kryptographische Operationen wie die Verschlüsselung einer Nachricht auszuführen. Nur der bestimmte Benutzer kennt den privaten Schlüssel, während der öffentliche Schlüssel an alle an der Kommunikation beteiligten Benutzer verteilt wird. Zudem können die Daten, die mit öffentlichen Schlüsseln verschlüsselt sind, nur mit dem privaten Schlüssel entschlüsselt werden ([10]; [13]).

1.2 Problematik

Da ECC dazu beiträgt, eine gleichwertige Sicherheit bei geringerer Rechenleistung und geringerem Ressourcenverbrauch zu erreichen, hat sich ECC zu einem attraktiven und sehr effizienten Public-Key-Krytosystem entwickelt [32]. Ihre Sicherheit basiert jedoch auf die Komplexität, das diskrete Logarithmusproblem in der Gruppe von Punkten auf einer elliptischen Kurve zu berechnen [11], da dieses Problem in nur exponentieller Zeit gelöst werden kann.

Außerdem ist auch die Art der zu verwendeten elliptischen Kurven unter Betrachtung der verwendeten Parameter (z.B. den Koeffizienten der Kurve) optimal auszuwählen [16]. Es existiert bereits mehrere Kurven die vom amerikanischen Standardinstitut NIST festgelegt wurden, obwohl deren Erzeugung allerdings nicht vollständig nachvollziehbar ist, was in amerikanischen Krypto-Standards zu erheblicher Kritik geführt hat [1].
Zudem gibt es eine erhebliche Anzahl potenzieller Schwachstellen für elliptische Kurven, wie z. B. Seitenkanalangriffe und Twist-Security-Angriffe, die bedrohen, die Sicherheit von angebotenen ECC privaten Schlüsseln, ungültig zu machen [34].
Also unabhängig davon, wie sicher ECC theoretisch ist, muss der Algorithmus ordnungsgemäß implementiert werden, da fehlgeschlagene Implementierung von ECC-Algorithmen zu erheblichen Sicherheitslücken in der kryptografischen Software führen können [34].

1.3 Zielsetzung

Ziel dieser Projektarbeit ist es einen Überblick über elliptischen Kurven in der Kryptographie zu geben. Zudem wird eine auf Modular- und Kurvenarithmetik basierende Implementierung der elliptischen Kurven für die kryptographische Anwendung bereitgestellt, die dann zur Erzeugung von Schlüsseln eines ECC-basierten Kryptosystems verwendet wird. Java wurde als bevorzugte Sprache in dieser Arbeit für die Implementierung von elliptischen Kurven gewählt, weil sie gut lesbar und mit einfachen Mitteln eine gute Schnittstelle für viele Webanwendungen bietet, wodurch unsere Implementierung auch von anderen Programmen genutzt werden kann.

1.4 Projektaufbau

Die vorliegende Arbeit ist wie folgt aufgebaut: Nach diesem einleitenden Abschnitt, gibt der zweite Abschnitt einen Überblick über das Thema Im Abschnitt 2 bietet eine Einfuhrung in elliptische Kurven und deren Arithmetik. //TODO

2 Grundlagen von elliptischen Kurven

2.1 Begriffsabgrenzung

Nach Dietmer ist unter dem Begriff **Kryptographie** „die Wissenschaft von geheimen Schreiben zu verstehen." Grundkonzept eines kryptografischen Systems ist also die Verschlüsselung von Informationen bzw. von Daten, um die Vertraulichkeit der Informationen zu gewährleisten [37].

Daten, die über einen unsicheren Kanal wie das Internet übertragen werden, werden mit Hilfe moderner Kryptosysteme so verschlüsselt, dass Unbefugte in einem Szenario, in dem sie auf die Informationen zugegriffen haben, diese nicht frei lesen und verstehen können [22].
Die unverschlüsselte Information wird als *Klartext* (Plaintext, Cleartext), die Verschlüsselte als *Chiffretext* (Ciphertext, Cryptotext) bezeichnet und der Verschlüsselungsprozess des Klartextes wird *chiffrieren* (engl. encryption) genannt. Umgekehrt wird unter *dechriffrieren* (engl. decryption), der Entschlüsselungsprozess eines Chiffretextes, verstanden.

Im Allgemeinen werden die beiden Prozesse durch eine Reihe von Regeln erreicht, sogenannte Verschlüsselungs- und Entschlüsselungsalgorithmen. Der Verschlüsselungsprozess basiert auf einem Schlüssel, der dann zusammen mit den Informationen als Eingabe an einen Verschlüsselungsalgorithmus übergeben wird. Danach können unter Verwendung eines Entschlüsselungsalgorithmus die Informationen mit dem entsprechenden Schlüssel abgerufen werden. Wer einen geheimen Schlüssel besitzt, kann die Informationen in Klartext entschlüsseln [22]. Die folgende Abbildung 2.1 verdeutlicht die Zusammenhänge.

In den 70er und 80er Jahren war die Kryptographie vor allem auf dem militären und diplomatischen Sektor beschränkt. Um geheime Nachrichten zu verschlüsseln, wurden sogenannte symmetrische Verschlüsselungsverfahren (z.B. Caesar-Verschlüsselung ([22])) verwendet, wo nur ein gemeinsamer geheimer Schlüssel sowohl für die Ver- als auch für die Entschlüsselung benutzt wird.

Aus diesem Grund ist es bei der symmetrischen Verschlüsselung sehr wichtig, dass der geheime Schlüssel auf einem sicheren Übertragungsweg an den Empfänger weitervermittelt wird, bevor die verschlüsselten Nachrichten übermittelt werden können. Früher wurde der Schlüssel meist persönlich, in Form eines Botens, übergeben. Da das persönliche Übergeben des Schlüssels sehr umständlich ist und das Risiko besteht, dass der

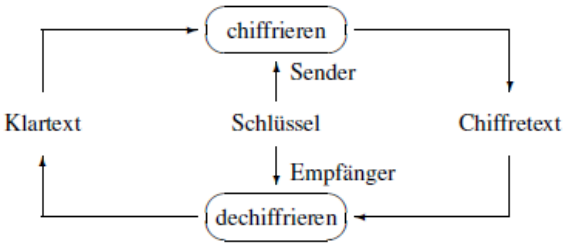

Abbildung 2.1: Grundkonzept der Kryptographie

Schlüssel belauscht oder gestohlen werden könnte, wurden weitere Verschlüsselungsverfahren vorgeschlagen. Dies sind die asymmetrischen Verschlüsselungsverfahren (auch Public-Key-Verfahren genannt) [5].

Im Gegensatz zu einem symmetrischen Verschlüsselungsverfahren erfordern asymmetrische Verschlüsselungsverfahren, nicht nur einen Schlüssel, sondern ein Schlüsselpaar bestehend aus einem öffentlichen Schlüssel und einem privaten Schlüssel [37]. Die Kommunikation erfolgt hier ohne vorhergehenden Schlüsselaustausch und ist jedoch viel langsamer als die Kryptografie mit privaten Schlüsseln. Mit dem privaten Schlüssel werden Daten entschlüsselt oder digitale Signaturen erzeugt, während mit dem öffentlichen Schlüssel Daten verschlüsselt und die Authentizität von erzeugten Signaturen überprüft werden ([30]; [37]). Die Abbildung 2.2 veranschaulicht diese Schlüssel.

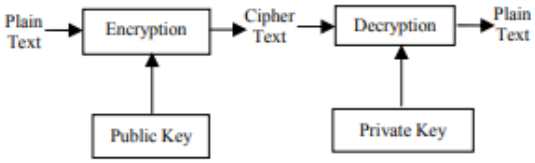

Abbildung 2.2: Public-Key Verschlüsselung

Das erste asymmetrische Kryptosystem, Rivest-Shamir-Adleman-Verfahren (RSA-Verfahren) wurde im Jahr 1977 von Ronald Rivest, Adi Shamir und Leonard Adleman gefunden, danach wurden weitere Kryptosysteme wie Rabin, Elgamal und Elliptische Kurven-Kryptogrphie vorgestellt.
Für unsere Arbeit liegt jedoch der Schwerpunkt auf der Kryptographie mit elliptischen Kurven und die Erzeugung von Schlüsseln, die für das Diffie-Hellman-Verfahren notwendig sind.

Die **elliptische Kurven-Kryptographie (ECC)** ist eine Verschlüsselungstechnik mit öffentlichem Schlüssel, die auf der algebraischen Struktur elliptischer Kurven über endlichen Körper basiert [20] und zur Erstellung schnellerer, kleinerer und effizienterer kryptografischer Schlüssel verwendet werden kann [6].

Die Verwendung einer elliptischen Kurve in der Kryptographie wurde im Jahr 1985 unabhängig voneinander von Miller [35] und Koblitz [24] vorgeschlagen. In den späten 1990er Jahren wurde ECC von einer Reihe von Organisationen wie ANSI [26], IEEE [33], ISO [14], NIST [2] standardisiert und erhielt kommerzielle Akzeptanz [12]. Das bekannteste Verschlüsselungsschema ist das Elliptic Curve Integrated Encryption Scheme (ECIES), das in IEEE- und auch in SECG SEC 1-Standards enthalten ist [28]. Beispiele für die Anwendung von ECC sind unter anderen Mehrzweck-Smartcards wie die neuen deutschen Ausweisdokumente [16].

Außerdem basieren Kryptosysteme mit öffentlichem Schlüssel auf der Lösung bestimmter mathematischen Probleme, z.B. beruht das RSA-Verfahren auf dem Integer Faktorisierungsproblem, DH und ECC basieren auf dem Diskreten Logarithmus-Problem. Bei der Lösung dieser Probleme stellt man im direkten Vergleich jedoch fest, dass herkömmliche Kryptosysteme mit öffentlichen Schlüsseln wie RSA Nachteile aufweisen.

Das Hauptproblem besteht darin, dass die Schlüsselgröße ausreichend groß sein muss, um die Sicherheitsanforderungen auf hohem Niveau zu erfüllen, was zu einer geringeren Geschwindigkeit und einem höheren Bandbreitenverbrauch führt.

Dies ist nicht den Fall, wenn elliptische Kurven (EC) für das Public Key Verfahren eingesetzt sind, da ECC im Vergleich zu RSA für den Benutzer eine gleichwertige Sicherheit bei kleineren Schlüsselgrößen bietet und für Angreifer schwerer exponentieller Zeitherausforderung, um in das System einzudringen [12].

Laut einigen Forschern kann ECC mit einem 164-Bit-Schlüssel ein Sicherheitsniveau erreichen, für dessen Erreichung andere Systeme einen 1.024-Bit-Schlüssel benötigen[6]. Es stellte sich heraus, dass ECC das effizienteste Kryptosystem mit öffentlichem Schlüssel ist [23]. Der Grund dafür ist, dass nach Miller und Koblitz das Diskreter-Logarithmus-Problem für elliptische Kurven schwerer sei als das klassische Diskreter-Logarithmus-Problem ist und zudem auch schnellere Laufzeiten ermögliche.
Bevor das Diskreter-Logarithmus-Problem vorgestellt wird, ist es wichtig zu definieren, was unter einer elliptischen Kurven zu verstehen ist.
Im Allgemeinen ist eine elliptische Kurve eine projektive, algebraische Kurve **C(K)**, auf der sich ein bestimmter Punkt O befindet, der als Punkt unendlich oder Nullpunkt bezeichnet wird [12].

Eine **elliptische Kurve E** über ein Körper K der Charakteristik $\neq 2$ oder 3, lässt sich formal definieren als die Menge der Punkte $(x, y) \in K$ der Weierstraß-Gleichung:

$$y^2 = x^3 + ax + b \quad \text{mit a, b} \in K \text{ [24].} \tag{2.1}$$

Elliptische Kurven in Form von Gleichungen können in **singuläre** und **nicht-singuläre** Gruppen unterteilt werden. In der ECC werden nicht-singuläre Kurven bevorzugt, damit eine Kurve frei von Spitzen oder Selbstüberschneidungen sein kann[13].

Eine elliptische Kurve wird als **nicht-singulär** bezeichnet, wenn sie keinen Punkt $P = (x, y)$ enthält, an dem ein mathematisches Objekt nicht definiert ist [5] und für die Werte a und b folgende Bedingung $\Delta = 4a + 27b \neq 0$ erfüllt, um eine endliche Abelsche Gruppe zu bilden.

In der abstrakten Algebra ist eine **abelsche** bzw. eine **kommutative** Gruppe, eine Gruppe G, in der das Ergebnis der Anwendung der Gruppenoperation auf zwei Gruppenelemente nicht von ihrer Reihenfolge abhängt. Also wenn $a \cdot b = b \cdot a \quad \forall a, b \in G$, wobei „ $\cdot$ “ eine Verknüpfung auf G.

Ein **Körper** ist eine Menge K mit zwei Verknüpfungen („ $+$ “, „ $\cdot$ “), sodass es gilt:

1. $(K, +)$ ist eine abelsche Gruppe. Das neutrale Element wird mit 0 und das inverse von $a \in K$ mit -a bezeichnet,
2. $(K \setminus 0, \cdot)$ ist eine abelsche Gruppe mit neutralem Element 1 und a^{-1} als inversem Element zu $a \in K$.
3. Distributivgesetze:

$$a \cdot (b + c) = (a \cdot b) + (a \cdot c) \quad \forall a, b, c \in K$$
$$(a + b) \cdot c = (a \cdot c) + (a \cdot b) \quad \forall a, b, c \in K$$

Ein Körper mit endlich vielen Elementen heißt endlicher Körper. Ein solcher Körper mit p Elementen wird mit $\mathbf{F_p}$ (auch: $\mathbf{GF(p)}$) bezeichnet.

Die **Charakteristik** eines Körpers **K** ist die kleinste positive Zahl **p** mit

$$\underbrace{1 \cdot 1 \cdot 1 \cdot \ldots \cdot 1}_{k \text{ mal}} = 0 \quad \text{, falls sie existiert.}$$

Dies ist beispielsweise für die Körper der rationalen Zahlen $\mathbb{Q}$, der reelen Zahlen $\mathbb{R}$ und der komplexen Zahlen $\mathbb{C}$ der Fall. Für jeden endlichen Körper ist die Charakteristik immer eine Primzahl [37].
K kann ein beliebiger Körper, also etwa $\mathbb{R}$, $\mathbb{Q}$, $\mathbb{C}$ oder ein endlicher Körper $\mathbb{F}$ sein [5]. Die obige Gleichung entspricht die Definition einer elliptischen Kurve über reelen Zahlen.

Obwohl eine elliptische Kurve über den reellen Zahlen ein guter Ansatz ist, um die Eigenschaften einer elliptischen Kurve zu verstehen, erfordert sie eine höhere Rechenzeit, um verschiedene Operationen auszuführen. Sie ist manchmal aufgrund von Rundungsfehlern ungenau. Kryptographie-Schemata erfordern jedoch eine schnelle und präzise Arithmetik [13]. Folglich werden in kryptografischen Anwendungen zwei Arten von elliptischen Kurven verwendet:

- Primzahlen über einem Feld $\mathbf{Z_p}$, wobei p eine Primzahl und p > 3 ist. Alle Variablen und Koeffizienten werden aus einer Menge von ganzen Zahlen von 0 bis p - 1 entnommen und Berechnungen werden über Modulo p durchgeführt [5].
- Binäre Kurve über dem Galois-Feld 2^m, auch bekannt als $GF(2^m)$, wobei alle Variablen und Koeffizienten in GF sind und Berechnungen über $GF(2^m)$ durchgeführt werden.

Da Primkurven analog zur Binärkurve keine erweiterte Bit-Fidding-Operation haben, sind sie für die Software-Implementierung geeignet [13].

In dieser Arbeit wurde die Implementierung von elliptischen Kurven über endliche Körper mit $\mathbf{K} = \mathbf{Z_p}$ berücksichtigt. Dazu wird mehr im Kapitel 4 erläutert. Eine elliptische Kurve über dem endlichen Feld $\mathbf{Z_p}$ enthält alle Punkte (x, y) in der $\mathbf{Z_p} \times \mathbf{Z_p}$ Matrix, die die folgende elliptische Kurvengleichung erfüllt:

$$y^2 = x^3 + ax + b \quad (mod p) \tag{2.2}$$

Dabei sind x und y Zahlen in $\mathbf{Z_p}$ und ähnlich wie im realen Fall ist $\Delta \neq 0$. Alle Punkte (x, y), die die obige Gleichung erfüllen, liegen auch auf der elliptischen Kurve. Der öffentliche Schlüssel ist ein Punkt der Kurve und der private Schlüssel ist eine Zufallszahl. Das Hinzufügen von Punkten auf der elliptischen Kurve ist jedoch kein einfacher Prozess, sondern an ein auf polynomzeit zu lösen Problem gebunden [21]: Das diskrete Logarithmusproblem.

2.2 Diskretes Logarithmusproblem

Bevor das Problem des diskreten Logarithmus erläutert wird, muss noch der Begriff der zyklischen Gruppe erklärt werden.

Eine **zyklische Gruppe** ist eine Gruppe, deren Elemente als Potenz eines ihrer Elemente dargestellt werden können. Also wenn es ein $a \in G$ gibt mit $\langle a \rangle = G$.

Nehmen wir nun an, dass (G, $\times$) eine multiplikative zyklische Gruppe mit Domainparametern g, h und n ist. Das **diskrete Logarithmusproblem** ist explizit definiert, als das

Problem der Bestimmung einer eindeutigen Ganzzahl x, die zufällig aus dem Intervall [1, p - 1] ausgewählt wird, so dass es gilt:

$$g^x = h \mod p$$

vorausgesetzt, dass eine solche ganze Zahl existiert. Der Parameter g ist die Basis des Logarithmus bzw. ein Erzeuger der Gruppe **G** ; n die Anzahl der Elemente in **G** ; der private Schlüssel bzw. das diskrete Logarithmusproblem von h zur Basis g ist die Ganzzahl x, und der öffentliche Schlüssel ist $h = g^x$ [8].

Das **Elliptic Curve Diskrete Logarithmus Problem (ECDLP)** ist ähnlich definiert, aber betrachtet die Weierstraß-Gleichung für eine elliptische Kurve $\mathbf{E} : y^2 = x^3 + ax + b$ über $\mathbb{Z}$ und zwei Punkte P und $Q \in \mathbf{F_p}$. Zu bestimmen ist, eine Zahl k $\in \mathbf{Z}$ mit

$$Q = kP \quad \text{, falls so ein k existiert.}$$

Die Primzahl p, die Gleichung der elliptischen Kurve E und der Punkt P und seine Ordnung n sind die Domainparameter. Der private Schlüssel ist die ganze Zahl k, die gleichmäßig zufällig aus dem Intervall [1, n - 1] ausgewählt wird, und der entsprechende öffentliche Schlüssel ist $Q = kP$.

Daher ist die Hauptoperation bei der ECC die Punktmultiplikation, also die Multiplikation eines Skalars k mit einem beliebigen Punkt P auf der Kurve, um einen anderen Punkt Q auf der Kurve zu erhalten.

Das Lösen von ECDLP ist viel schwieriger als DLP, da die Komplexität der Punktarithmetik in ECDLP im Vergleich zur Ganzzahlarithmetik in DLP zunimmt. Aus diesem Grund ist ECC in der Lage, ein ähnliches Sicherheitsniveau wie RSA bereitzustellen, jedoch mit einer kürzeren Schlüsselgröße, was es wiederum zur beliebtesten Wahl macht [21].

Damit ein auf $\mathbf{F_p}$ basierendes diskretes Logarithmus-System effizient ist, sollten schnelle Algorithmen zur Berechnung der Gruppenoperation bekannt sein. Aus Sicherheitsgründen sollte das Problem des diskreten Logarithmus in $\mathbf{Z_p}$ unlösbar sein [8].
Es gibt viele Algorithmen zur Lösung von **ECDLP**, aber der erfolgreichste ist die Kombination von Pollards Rho- und Pohlig-Hellman-Angriffen mit vollständig exponentieller Laufzeit ([21];[8]). Angriffe resultieren normalerweise aus den Schwächen bei der Auswahl der elliptischen Kurve und des endlichen Feldes (siehe Kapitel 4 und 5), aber die meisten Angriffe können durch korrekte Auswahl der Parameter der elliptischen Kurve vereitelt werden [21]. Daher sollten diese öffentlichen Parameter sicher ausgewählt werden, um alle erkannten Angriffe zu vermeiden [4]. Der nächste Abschnitt beschäftigt sich näher mit diesen Parametern.

2.3 Domänenparameter für elliptischen Kurven

Vor der Implementierung eines ECC-Systems müssen mehrere Entscheidungen getroffen werden. Dazu gehören die Auswahl von Domänenparametern für elliptische Kurven (zugrunde liegendes endliches Feld, Felddarstellung, elliptische Kurve) sowie Algorithmen für Feldarithmetik, elliptische Kurvenarithmetik und Protokollarithmetik.

Die Auswahl kann durch Sicherheitsaspekte, Anwendungsplattform (Software oder Hardware), Einschränkungen der jeweiligen Computer- und Kommunikationsumgebung

(z. B. Speichergröße, Bandbreite) beeinflusst werden [7].

Bevor der Verschlüsselungsprozess gestartet und der verschlüsselte Text transformiert wird, müssen Domänenparameter von beiden Parteien vereinbart werden, die an der sicheren und vertrauenswürdigen Kommunikation über ECC beteiligt sind [32]. Sie werden von einer Gruppe von Benutzern gemeinsam genutzt und können in einigen Anwendungen jedoch für jeden Benutzer spezifisch sein [17]. Es können zwei Arten von elliptischen Kurvendomänenparametern verwendet werden [29]:

- elliptische Kurvendomänenparameter über $\mathbf{Z_p}$ und
- elliptische Kurvendomänenparameter über $\mathbf{F_2^m}$.

2.3.1 Parameter über Z_p

Die Domänenparameter für die elliptische Kurve über $\mathbf{F_p}$ sind ein Sextuple

$$(p, a, b, G, n, h)$$

bestehend aus einer ganzen Zahl **p**, die das endliche Feld $\mathbf{Z_p}$ spezifiziert; der Kurve aus 2.2, den Parametern a und b, einem Basispunkt $G = (x_G, y_G)$ auf $E(\mathbf{Z_p})$, **n** die Ordnung der elliptischen Kurve und dem Cofaktor h, wobei

$$h = \frac{\#E(F_p)}{n}$$

und $\#E(\mathbf{F_p})$ die Anzahl der Punkte auf einer elliptischen Kurve ist ([29]; [12]).

2.3.2 Parameter über F_2^m

Die Domänenparameter für die elliptische Kurve über $\mathbf{F_2^m}$ sind ein Septuple

$$T = (m, f(x), a; b; G; n; h)$$

bestehend aus einer ganzen Zahl m, einem irreduziblen binären Polynom $f(x)$ vom Grad m, Parameter a, b $\in \mathbf{F_2^m}$ der durch die Gleichung der elliptischen Kurve $\mathbf{E}(\mathbf{F_2^m})$:

$$y^2 + xy = x^3 + ax^2 + b \quad \in F_2^m \tag{2.3}$$

definiert sind, einem Basispunkt $G = (xG; yG)$ auf $E(F_2^m)$, eine Primzahl n, die in der Größenordnung von G liegt und die Cofaktor h mit

$$h = \frac{\#E(F_p)}{n}$$

und $\#E(\mathbf{F_p})$ die Anzahl der Punkte auf einer elliptischen Kurve ist. ([29]; [12])

Um einen Angriff auf ECDLP zu vermeiden, muss $|E|$ eine ausreichend große Primzahl sein. Zumindest wird vorgeschlagen, dass $n > 2^{160}$ ist [17].

Für diese Arbeit wurden Parameter in $\mathbf{Z_p}$ bevorzugt. Die Verwendung der elliptischen Kurvenarithmetik macht ECC unter allen vorhandenen kryptografischen Schemata einzigartig. Je nach gewähltem Feld verwendet ECC für seine Operationen modulare Arithmetik oder Polynomarithmetik. Dies wird im nächsten Kapitel präsentiert.

3 Zahlentheoretische Grundlagen

Hoher Durchsatz und geringe Ressourcen sind in vielen Anwendungen die entscheidenden Entwurfsparameter des ECC-Prozessors [27]. Da die Effizienz von ECC-Prozessoren hauptsächlich von modularen arithmetischen Operationen wie modularer Addition, Subtraktion und Multiplikation abhängt, ist der effiziente Entwurf modularer Arithmetik eine sehr anspruchsvolle Aufgabe für die Implementierung eines Hochleistungs-ECC-Prozessors [27].

In diesem Kapitel werden die wichtigsten Arithmetikoperationen über dem Primfeld $\mathbf{Z_p}$, die für die Kryptographie von Bedeutung sind, zusammen mit Beispielen vorgestellt und beschrieben.

3.1 Modulararithmetik

Die modulare Arithmetik ist:

> „ein Prozess zum Reduzieren einer Zahl modulo einer anderen Zahl, wobei dieser Prozess zahlreiche Multi-Präzisions-Gleitkomma-Divisionsoperationen umfassen kann [9]. “

Die modulare Artihmetik bietet endliche Strukturen, die alle üblichen arithmetischen Operationen der ganzen Zahlen aufweisen und die mit vorhandener Computerhardware problemlos implementiert werden können [31]. Eine wichtige Eigenschaft dieser Strukturen ist, dass sie durch Operationen wie Exponentiation zufällig permutiert zu sein scheinen, aber die Permutation kann oft leicht durch eine andere Exponentiation umgekehrt werden [31]. In entsprechend ausgewählten Fällen ermöglichen diese Vorgänge die Ver- und Entschlüsselung oder die Generierung und Überprüfung von Signaturen [31].

Die Berechnung hier erfolgt genauso wie bei der normalen Arithmetik und der einzige Unterschied besteht darin, dass alle Operationen in der modularen Arithmetik in Bezug auf eine positive ganze Zahl, den Modul, ausgeführt werden. Ziel ist der kleinste positive Rest zu finden.

Im Allgemeinen lässt sich eine modulare Reduktion durch die folgende Gleichung definieren:

$$r = x \mod p = x - \lfloor \frac{x}{p} \rfloor \cdot p$$

, wobei x die zu reduzierende Anzahl modulo p ist, der gefundene Rest r, der im Bereich von [0, p - 1] liegt [9]. Daraus lässt sich die Äquivalenzrelation definieren und man sagt, dass r kongruent zu x modulo p ist.

Eine Zahl r ist **kongruent** bzw. **äquivalent** zu einer Zahl x modulo p, ausgedrückt durch $r \equiv x \pmod{p}$, falls p ein Teiler von (r - x) ist. 13

Dies bedeutet, dass r und x den gleichen Rest haben, wenn sie durch p geteilt werden und $r = k \cdot p + x$ mit $k \in \mathbf{Z}$.

In dieser Arbeit bildet die Klasse *BasicTheoreticMethod.java*, das Grundgerüst der Implementierung von elliptischen Kurven über das Primfeld $\mathbf{Z_p}$. In dieser Klasse wurden folgende Funktionen implementiert:

- *modCalculation*, die den Rest aus der Division zweier ganzen Zahlen bestimmt;
- *isKongruent*, die prüft ob zweier Zahlen den selben Rest nach der Division;
- *modAddition*
- *modSubtraction*
- *modMultiplikation*
- *gcdExtended*
- *hasInverse*
- *multiplicativeInverse*
- *modDivision*
- *modExponentiation*
- *phiFunction*
- *chineseremainder*

Diese Klasse enthält unter anderen zahlentheoretische Methoden, die eine wichtige Bedeutung für die Kryptographie haben. Die erste wichtigste Methode ist die Methode der Berechnung vom Modulo: $\mathbf{X} \pmod{P}$

3.1.1 Berechnung von X mod Y

Eine naivste Methode wurde implementiert, in dem das Finden von $r = x \pmod{p}$ zuerst durch wiederholte Berechnung des Quotients $q = \frac{x}{p}$ und dann durch wiederholtes Subtrahieren von x mit dem Ergebnis aus der Multiplikation von p mit q, bis das Ergebnis im Bereich von $[0, p-1]$ liegt. Die Leistung der Modulo-Operation hängt vor allem hier von der Leistung des Divisionsalgorithmus ab. //

Die naivste Art eine ganzzahlige Division zu implementieren, besteht darin, solange den Divisor p von der Dividende x zu subtrahieren, bis die Dividende x negativ wird, und dabei den Quotienten q hochzuzählen. Wenn die Dividende x negativ wird, wird sie vom Divisor addiert und der Quotient um eins verringert [15].

Die implementierte Methode ist jedoch sehr langsam und kostspielig, wenn die Anzahl der Iterationen berücksichtigt wird, wenn eine große Zahl durch eine kleine Zahl geteilt wird [25].

Aber Methoden wie die **Barrett-Reduktion** können verwendet werden, um die Modulo-Operation zu optimieren ([19], [36]).

Eine **Barrett-Reduktion** ist ein Verfahren zum Reduzieren einer Zahl modulo einer anderen Zahl, wobei die Division durch Multiplikationen und Verschiebungen ersetzt

werden können, da die beiden letzteren Operationen viel billiger sind [9].
Also der Quotient $q = \frac{x}{p}$ wird unter Verwendung kostengünstigerer Operationen mit Potenzen einer geeignet gewählten Basis b geschätzt. Für die Barett-Reduktion wird b häufig als Potenz von 2 gewählt werden.

Eine modulabhängige Größe $m = \left\lfloor \frac{b^{2k}}{p} \right\rfloor$ muss berechnet werden, wodurch der Algorithmus für den Fall geeignet ist, dass viele Reduktionen mit einem einzigen Modul durchgeführt werden [8]. Der Quotient q kann nun einmal nur für jedes Modul gleich dem Kehrwert von p berechnet werden [25].

Dies erklärt sich aus der Tatsache, dass beispielsweise jede RSA-Verschlüsselung für eine Entität ein Reduktionsmodul für den öffentlichen Schlüssel dieser Entität erfordert [3].

Das Modulares Reduktonsverfahren umfasst [9]:

- die Eingabe die zu reduzierenden Wert x und des Modulos p, wobei dies eine spezielle Form (z.B. NIST-Primzahl) hat,
- das Durchführen der modularen Reduktion von x mod p, wobei die modulare Reduktion umfasst:
  - Berechnen eines genäherten Basisquotienten m,
  - Berechnen einer Quotienten-Näherung q,
  - Berechnen eines genäherten Rests r, und
  - Berechnen einer geschätzten, reduzierten Form des Wert x auf der Basis der Quotienten-Näherung und des genäherten Rests.

Die Tabelle 4.1 veranschaulicht den Barett-Reduktion-Algortihmus.

Algorithmus: Modulo-Berechnung
Input: b > 3, p, $k = \lfloor log_b p \rfloor + 1, 0 \leq x < b^{2k}, m = \left\lfloor \frac{b^{2k}}{p} \right\rfloor$
Output: $x mod p$
1. Berechne $q = \left\lfloor \left\lfloor \frac{x}{b^{k-1}} \right\rfloor \frac{m}{b^{k+1}} \right\rfloor \cdot p$;
2. $r = (x \bmod b^{k+1}) - (q \cdot p \bmod b^{k+1})$;
3. If r < 0 then $r = r + b^{k+1}$;
4. While $r \geq p$ do r = r - p;
5. Return r.

Tabelle 3.1: Barett-Reduktion[2]

Weitere grundlegende und wesentliche Operationen für die Kryptographie sind die modulare Addition, modulare Subtraktion und modulare Multiplikation.

3.1.2 Modulare Addition

Die modulare Addition über $\mathbf{Z_p}$ kann mathematisch geschrieben werden als:

$$x + y \mod p$$

wobei x und y die gegebene Zahlen und p eine Primzahl sind. In dieser Arbeit wurde für die Implementierung der modularen Addition zuerst x und y addiert, dann wird das Ergebnis modulo p berechnet, wenn dies außerhalb des Bereichs von [0, p-1] liegt, sonst wird die Summe direkt ausgegeben.

Seien zum Beispiel x = 17, y = 4, p = 5, dann ist

$$x + y \pmod{p} = \quad 17 + 4 \pmod{5} = \quad 21 \equiv 1 \mod 5$$

Ein kostengünstiger bzw. schnellerer Algorithmus wäre zuerst x und y binär darzustellen und in einem Array von t-Bit zu speichern. Danach sind x und y wortweise bzw. bitweise zu addieren und das Ergebnis dann p subtrahieren, solange es p - 1 überschreitet.

Jede Wortaddition erzeugt zum Beispiel in einer 32-Bit Plattform-Architektur eine 32-Bit-Summe und eine 1-Bit-Übertragsziffer , die zur nächsthöheren Summe addiert werden [2]. Die einfache Addition und die *Addition mit Übertrag* können schnelle Einzeloperationen sein [27]. Die Tabellen 3.2 und 3.3 veranschaulichen die beiden Algorithmen.

Klassischer Algorithmus: Modulare Addition
Input: Modulo p und Zahlen x, y $\in$ [0, p - 1]
Output: $r = x + y \pmod{p}$
1. Berechne r = x + y;
2. If $r \neq 0$, then finde den Rest r = r mod p else r = 0;
3. return r.

(a) 1.Tabelle

Tabelle 3.2: Modulare Addition aus der Klasse *BasicTheoreticMethods*

Algorithmus: Modulare Addition
Input: Modulo p und Zahlen x, y $\in$ [0, p - 1]
$t = \lceil m/32 \rceil$ und $m = \lceil log_2 p \rceil$
Output: $c = (x + y) mod p$
1. $c_0 = x_0 + y_0$;
2. For i from 1 to t-1 do: $c = Add\_With\_Carry(x_i, y_i)$;
3. If the carry bit is set, then subtract p from $c = (c_{t\_1}, ..., c_2, c_1, c_0)$;
4. If $c \geq p$ then $c = c - p$;
5. Return c.

(a) 2.Tabelle

Tabelle 3.3: Empfohlene modulare Addition von amerikanischen Standard NIST für eine 32-Bit Architektur-Plattform [2].

3.1.3 Modulare Subtraktion

Mathematisch kann die modulare Subtraktion geschrieben werden als:

$$(x - y) \mod p$$

wobei x und y die gegebenen Zahlen und p die Primzahl sind. Ein einfachster Weg diese Operation durchführen, besteht darin, die beiden Zahlen x und y zuerst zu subtrahieren und dann der kleinste positive Rest im Bereich von [0, p-1] zu berechnen, in dem das Ergebnis der Subtraktion von (x - y) durch p geteilt wird. Der Algorithmus sieht genauso wie den Algorithmus der modularen Addition, wobei im ersten Schritt anstatt eine Addition, eine Subtraktion durchgeführt wird.

Seien zum Beispiel x = 15, y = 23, p = 5, dann ist

$$(x - y) \mod p = \quad (15 - 23) \mod 5 = \quad -8 \equiv 2 \mod 5$$

In der effizienteren Art, Modulo-Subtraktion durchzuführen, wird das Übertragungsbit nicht mehr als `Carry-Bit`, sondern als `Borrow-Bit` (Ausleih-Bit) interpretiert [2]. Die Operation wird dann ähnlich wie die modulare Addition implementiert.

3.1.4 Modulare Multiplikation

Die modulare Multiplikation ist eine der teuersten und zeitaufwändigsten Operationen der Kryptographie über Z_p. Aber um ein höheres, leistungsstarkes Kryptosystem zu entwickeln, muss eine effiziente Implementierung der modularen Multiplikation erfolgen [27]. Mathematisch kann die modulare Multiplikation-Operation ausgedrückt werden als:

$$x \cdot y \mod p$$

Um Modulo-Multiplikationen durchzuführen, wurde in dieser Arbeit die Zahlen x und y einfach multipliziert und dann die ganzzahlige Division durch p zu verwendet, um den Rest zu erhalten. Dies bedeutet auch, dass die Leistung des ganzen Algorithmus, genauso wie bei der modularen Addition und Subtraktion, von Modulo-Operationen bzw. von der Leistung des Divisionsalgorithmus abhängt.
Seien zum Beispiel: x = 35, y = 7, p = 5, dann gilt:

$$x \cdot y \mod p = \quad 35 \cdot 7 \mod 5 = \quad 245 \equiv 0 \mod 5$$

Neben der Barett-Reduktion Methode gibt es auch eine andere Methode, die die Modulo-Operationen viel schneller als die reguläre klassische Methode durchführen kann: Die **Montgomery-Multiplikation**.

Die **Montgomery-Multiplikation** ist ein Verfahren zur modularen Multiplikation, bei der die traditionelle Teilung-Operation vermeidet wird und anschließend nur Multiplikationen, Additionen und Verschiebungen verwendet werden[30]. Die Zahlen x und y werden binär dargestellt Der modulare multiplikative Algorithmus ist in Tabelle 3.4 angegeben.

Das Verfahren ist für eine einzelne modulare Multiplikation nicht effizient, kann jedoch effektiv bei Berechnungen wie der modularen Exponentiation verwendet werden, bei denen viele Multiplikationen für eine gegebene Eingabe durchgeführt werden [8]. Im nächsten Unterabschnitt werden Methoden zur Berechnung der modularen Exponentiation erläutert.

Algorithmus: Modulare Multiplikation

Input: p, x und y zwei positive k-Bit Ganzzahlen, x_i, y_i: i-te Bit in x und y
Output: $m = x \cdot y \bmod p$

1. m = 0;
2. For i = 0 to k-1
   2.1 $m = m + (x \cdot y_i)$;
   2.2 If $(m_0 = 1)$ then $m = m/2$ else $m = (m + p)/2$;
3. Return m.

Tabelle 3.4: Montgomery-Multiplikation [30]

3.1.5 Modulare Exponentiation

Das Problem der modularen Exponentiation lässt sich mathematisch definieren als:

$$A = x^k \mod p \quad mit \quad x \in \mathbf{Z_p} \quad und \quad 0 \leq k < p$$

Die modulare Exponentiation ist die kostspielige, aber entscheidende und bedeutungsvolle Methode für viele kryptographische Protokolle.

Algorithmen zur schnellen modularen Exponentiation wie das Square-and-Multiply-Verfahren, basieren vor allem auf der Auswertung der Binärdarstellung des Exponenten k [18]. Dort wird der Exponent k binär dargestellt, als k = $\sum_{i=0}^{t} k_i \cdot 2^i$ mit $k_i \in \{0, 1\}$ und die Auswertung kann dabei je nach Algorithmen entweder von links nach rechts oder von rechts nach links erfolgen [18].

Bei einem k-Bit-Exponenten werden dann für die Exponentiation höchstens k Quadrate und k Multiplikationen benötigt [18]. Selbst wenn sie effizient mit dem wiederholten Square-and-Multiply-Verfahren durchgeführt wird, erreicht sie eine Bit-Komplexität von $\mathcal{O}(\log n^3)$ [3].

Gegeben sei beispielsweise $x^k = x^{13}$. Zu berechnen ist, $2^{13} \mod 7$. Dann ist die binäre Darstellung von k = 1101. Der Exponent k lässt sich wie folgt auswerten:

$$((((0) \cdot 2 + 1) \cdot 2 + 1) \cdot 2 + 0) \cdot 2 + 1 = 13$$

Daraus lässt sich x^{13} bzw. 2^{13} wie folgt aufteilen:

$$x^{13} = ((((x^0)^2 \cdot x^1)^2 \cdot x^1)^2 \cdot x^0)^2 \cdot x^1$$

$$2^{13} = ((((2^0)^2 \cdot 2^1)^2 \cdot 2^1)^2 \cdot 2^0)^2 \cdot 2^1$$

Es ergibt sich, dass $2^{13} \equiv 2 \mod 7$.

Die schnelle modulare Exponentiation Kann jedoch auch in einfacher Weise realisiert werden [18]. Tatsächlich, kann x^{13} auch dargestellt werden, als

$x^{13} = x^{12} \cdot x$ und x^{12} weiterhin als $x^{12} = (x^6)^2$.

Ausgehend von dieser Beobachtung, ergibt sich die folgende Rekursionsformeln [18]:

$$x^k = \begin{cases} 1 & \text{falls } k = 0 \\ x^{k-1}.x & \text{falls } k \quad \textit{ungerade} \\ (x^{k/2})^2 & \text{falls } k \quad \textit{gerade} \end{cases}$$

In dieser Arbeit wurde für die Implementierung der modularen Exponentiation, die Idee der Rekursionsformeln verfolgt. In dieser rekursive Implementierung wurde die Binärdarstellung des Exponenten k nicht explizit benötigt.

Der Algorithmus startet mit Basisfällen, in dem es geprüft wird, ob die Zahlen x, k und p in den richtigen Bereichen liegen. A wird am Anfang auf 1 gesetzt (A = 1). Wenn k gleich null ist, wird A = 1 zurückgegeben. Wenn k größer null ist, wird in jedem Schritt wird geprüft, ob der Exponent k ungerade bzw. ob die aktuelle binäre Zahl eins ist. Wenn ja, wird das Ergebnis A mit x multipliziert und der Modulo berechnet; ansonsten erfolgt zuerst eine Rechtsverschiebung (k wird halbiert) und danach wird x quadriert. Danach wird den Rest bestimmt, damit $x \in [0, p-1]$ bleibt. Diese Schritte werden durchgeführt, solange der Exponent größer 0 ist. Der Algortihmus wird in der Tabelle 3.5 dargestellt. Also die Leistung dieser Methode hängt von der Leistung des Modulo-Algorithmus ab.

Algorithmus: Modulare Exponentiation
Input: p, x, $k \in \mathbf{Z}$
Output: $x^k \mod p$
1. A = 1;
2. If $(k = 0)$ then return A;
3. If $(k < 0)$ then $k = \lvert k \rvert$;
x = $(x^{-1} \mod p) \mod p$;
4. x = x $\mod$ p;
5. while (k > 0)
5.1 If $(k \,\&\, 1) = 1$ then A = A $\cdot$ x *mod* p;
5.2 k $\gg 1$;
5.3 x = x $\cdot$ x $\mod$ p;
6. Return A.

Tabelle 3.5: Modulare Exponentiation aus der Klasse *BasicTheoreticMethods.java*

In den meisten wissenschaftlichen Artikeln, liegt $k \in [0, p-1]$, aber in dieser Arbeit es wurde angenommen, dass k $\in \mathbf{Z}$ liegt. Betrachtet wurde also die Fälle, wo der Exponent k auch negative Werte haben kann, um soweit wie möglichst Sicherheitslücken auszuschließen.

In diesem Fall wird für die Berechnung der modularen Exponentiation, den Absolutwert des Exponenten genommen, aber vorher nimmt x den resultierenden Wert aus der Berechnung der modularen multiplikative Inverse von x und p.

3.1.6 Modulare multiplikative Inverse

Die modulare Inverse ist eine weitere wichtige Methode der Kryptographie. Wenn a eine ganze Zahl$\in \mathbf{Z_p}$ und p eine Primzahl ist, ist die modulare Inverse a^{-1} eine ganze Zahl, die die Beziehung

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

erfüllt.

Die modulare Inverse wird mit Hilfe des extended Euklidean Algorithmus (EEA) bestimmt. Dort sind die ganzen Zahlen x und y zu finden, für die die folgende Gleichung erfüllt ist: $ax + py = 1$.
Also muss der $ggT(a, p) = 1$ sein. Wenn dies den Fall lässt sich feststellen, dass a und p **Koprime** sind.

Extended Euklidischer Algortihmus

Der extended Euklidische Algorithmus basiert auf den euklidischen Algorithmus, der der ggT von zwei ganzen Zahlen durch wiederholte Anwendung des Divisionsalgorithmus ermittelt. Aber bei der Berechnung des ggT verfolgt wird auch den Wert von x.

Der erste Schritt des euklidischen Algorithmus besteht darin, die größere Ganzzahl durch die kleinere zu teilen. Dann wird Der Divisor wiederholt durch den Rest geteilt, bis der Rest 0 ist. Der ggT ist dann der letzte Rest ungleich Null in diesem Algorithmus.
Allerdings die Voraussetzung für die Bestimmung der Inverse ist, dass $ggT(a, p) = 1$ sein muss. Wenn dies den Fall ist, können durch Umkehren der Schritte im euklidischen Algorithmus die ganzen Zahlen x und y gefunden werden.

Die ganze Idee ist, mit der ggT zu beginnen und rekursiv rückwärts zu arbeiten. Dies kann erreicht werden, indem die Zahlen als Variablen behandelt werden, bis den Ausdruck $1 = x \cdot a + y \cdot p$ erhalten wird, der eine lineare Kombination unserer Anfangszahlen ist.
Daraus es folgt $x \cdot a \equiv 1 \mod p$. Mit Hilfe der modularen multiplikative Inverse lässt sich leicht die modulare Division zwischen 2 Zahlen bestimmen.

3.1.7 Modulare Division

Die modulare Division lässt sich mathematisch definieren als:

$$x/y \mod p \quad = \quad x \cdot y^{-1} \mod p$$

Bei der Bestimmung der modularen Division wird zuerst y^{-1}, die modulare Inverse von y über p berechnet. Danach wird die modulare Multiplikation von $x \cdot y^{-1}$ durchgeführt.

3.2 Chinesischer Restsatz

Ein weitere Methode, die eine wichtige Rolle in der Entschlüsselung bzw.in der Generierung von Schlüsseln für die Kryptographie spielt, ist der chinesische Restsatz. Der chinesische Restsatz (engl. Chinese Remainder Theorem: CRT) besagt, dass wenn 2 Zahlen p und q Koprime sind (also,wenn ggT(p,q) = 1), dann hat das Gleichungssystem:

$$x \equiv a \mod p$$
$$x \equiv b \mod q$$

eine eindeutige Lösung für $x \mod p \cdot q$.

Also wenn p und q Koprime sind, es existiert 2 Zahlen m_1 und m_2, so dass $m_1 \cdot$ p + $m_2 \cdot$ q = 1 gilt. Um m_1 und m_2 zu bestimmen, wird der EEA verwendet und daraus ergibt sich, dass

$$x = a \cdot m_2 \cdot p + b \cdot m_1 \cdot q, \quad \text{für 2 Gleichungen ist.}$$

Für beliege Gleichungen:

$$x = \sum_{i=1}^{k} a_i \cdot N_i \cdot R_i \mod N, \quad \text{mit k, die Anzahl der Gleichungen;} \tag{3.1}$$

$$N_i = N/n_i \quad \text{und} \quad R_i = N_i^{-1} \mod N.$$

In der Tat ermöglicht dieser Satz, eine Nachricht zu entschlüsseln, in dem am Anfangs eine geheime Nachricht M in beliebige Hälften (hier in 2 Hälften m_1, m_2) aufgeteilt wird und dann das Modulo jede diese Nachricht berechnet, wenn die Faktorisierung des öffentlichen Schlüssels $N = p \cdot q$ gekannt ist. Danach werden diese Nachrichten in einer Zahl x wieder kombiniert sein und ausschließlich wird der private Schlüssel berechnet.

Gegeben sind zum Beispiel 2 Zahlen p = 5 und q = 7. Gesucht ist die Zahl x für die es gilt:

$$x \equiv 2 \mod 5$$
$$x \equiv 3 \mod 7$$

Berechnung des ggT (5,7) mittels des EEA ergibt 1 = 3 · 5 - 2 · 7, also es existiert 2 Zahlen m_1, m_2, mit $m_1 = 3$ und $m_2 = -2$.
Dann ist $x = 2 \cdot -2 \cdot 7 + 3 \cdot 3 \cdot 5 = -28 + 45 \equiv 17 \mod 35$.

Für diese Arbeit wurde für die Implementierung die Gleichung 3.1 betrachtet. Zwei Array-Listen wurden genutzt, um alle Reste a_i und alle Modulo-Werten N_i zu speichern. K wird als die Größe der Liste genommen und der Algortihmus wird nur laufen, für Liste gleicher Länge. Die Tabelle 3.6 stellt der entsprechenden Algorithmus dar.

Die Gleichung aus 3.1 wird als Gauß-Algorithmus bezeichnet. Die Berechnungen können in $\mathcal{O}(\log n^2)$ Bitoperationen durchgeführt werden [3].

3.3 Polynomarithmetik

Algorithmus: Chinesischer Restsatz

Input:$a = \{a_1, ..., a_k\}, n = \{n_1, ..., n_k\}$, wobei alle n_i co-prime sind
Output: $x = \sum_{i=1}^{k} a_i \cdot N_i \cdot R_i \mod N$

1. x = 0;
2. k = n.size(); 2. If $(k > a.size())$ then k = a.size();
3. For $i = 0$ bis k-1 compute product of all moduli N =$n_1 * ... * n_k$;
4. For $i = 0$ bis k-1
   4.1 compute $N_i = N/n_i$;
   4.2 compute Inverse $R_i = N_i^{-1} \mod n_i$;
   4.3 x = x + $a_i \cdot R_i \cdot N_i$
5. Return $x \mod N$.

Tabelle 3.6: Chinesischer Restsatz aus der Klasse *BasicTheoreticMethods.java*

4 Endliche Körper

4.1 Primzahl-Generator

In der Kryptographie arbeitet man mit Primzahlen, jedoch nicht mit kleinen Zahlen. Jedoch bei höheren Primzahlen wird es schwieriger die Zahlen zu überprüfen, ob diese eine Primzahl ist. Eiune davon ist der Fermat-Test.

4.1.1 Fermat-Test

Der Fermat-Test ist einer der möglichen Algorithmen um Primzahlen herauszufinden.

Algorithmus: Fermat-Test
Input: Eine Zufällige Zahl
Output: true = Pseudoprimzahl, false = Keine Primzahl
1. Wähle eine Zahl $a \in 2, 3, 4, ..., n-2$
2. ggT berechnen. Überprüfe ob die Zahlen Teilerfremd sind.
-> Nicht teilerfremd => gemeinsamer Teiler => return false
3. Falls die Zahlen teilerfremd sind, so führe folgende Formel aus.
$a^{n-1} \equiv 1 \; mod \; n$
4. Wiederhole für eine gewisse Anzahl an Versuchen.
Merke Anzahl bestandenen versuchen.
5. F-Zeuger > F-Lügner? -> true: return true
-> false: return false

Tabelle 4.1: Test

Aber es gibt Zahlen, welche F-Zeugen sind, jedoch ist die untersuchte Zahl keine Primzahl. Anhand folgendem Beispiel können wir erkennen, wieso das ist:
Testen wir die Zahl 15. So kommt bei der zufälligen Zahl 4 in der Rechnung:
$4^{15-1} \; mod \; 15 = 1$
Es wird angedeutet, das die Zahl 15 eine Primzahl ist. Welches es jedoch nicht ist. Da man durch $3 \cdot 5 = 15$ kommt.
Aus dem Grund ist es wichtig den Test mehrmals zu durchführen. Dadurch verringert sich die Fehlerwahrscheinlichkeit. Es erhöht aber die Laufzeit dadurch.

Algorithmus im Programm

Es wird eine Zufällige Zahl generiert. Bei unseren Kryptographie-Beispielen wird mit einer 8192 Bit-Zahl erwartet. Daher geben wir auch die Länge der Bits mit hinzu.
Beim Fermat-Test wird eine 2.te Zahl (welcher kleiner ist) generiert und es wird nach dem

ggt von den 2 zufälligen Zahlen ermittelt. Ist der ggT von den beiden größer als 1,
so ist die zu überprüfende Zahl keine Primzahl. Ist es 1, so geht es weiter mit dem Test.
Wir nutzen die Formel oben und überprüfen, ob das Ergebnis 1 rausbringt. Ist dieser nicht 1, so ist es keine Primzahl. Ergibt es 1, so ist die 2.te generierte Zahl ein F-Zeuge und inkrementiert den Counter
Diesen Test für die 2.te Zahl generieren, wird beliebig mal ausgeführt und es wird überprüft, ob mehr F-Zeugen gibt als F-Lügner. Gibt es mehr F-Zeugen als F-Lügner, so ist unsere 1. generierte Zahl wahrscheinlich eine Primzahl.

4.2 Was ist ein endlicher Körper (Galois-Feld)?

Ein Endlicher Körper oder Galois-Körper ist ein algebraisches System, welches eine endliche Menge mit definierten Addition und Multiplikation.
Der Endliche Körper hat folgende Eigenschaften:

Die Endliche Menge ist eine abelsche Gruppe, in der Die Rechenoperation Addition berechnet werden kann. Endliche Menge ohne dem Nullelement ist eine abelsche Gruppe, in der die Multiplikation stattfinden kann. Endliche Körper sind auch zyklische Gruppen mit der Schreibweise: $\mathbb{Z}_p$. Der Unterschied zwischen x-Beliebige Zyklische Gruppen und einem Endlichen Körper ist, das der Endliche Körper durch einer Primpotenz p^n beschrieben werden kann.

Die Anzahl an Elementen in einem endlichen Körper nennt man Ordnung des Endlichen Körper. Die Ordnung des Endlichen Körper kann nur als Primpotenz dargestellt werden.

Allgemein: Die endlichen Körper $\mathbb{F}_p$ beinhalten p-Elemente. Diese werden durch die Ganzzahlen $0, 1, 2, ..., p-1$ repräsentiert.

Im Zusammenhang mit den Grundrechenoperationen wird es folgendermaßen definiert:

Addition: Wenn $a, b \in \mathbb{Z}_p$, dann gilt $a + b = r$ in $\mathbb{Z}_p$, wobei $r \in [0, p-1]$ ist. Dies Zahl wird durch modulo p errechnet, nach der Addition. Multiplikation: Wenn $a, b \in \mathbb{Z}_p$, dann gilt $a \cdot b = r$ in $\mathbb{Z}_p$, wobei $r \in [0, p-1]$ ist. Dies Zahl wird durch modulo p errechnet, nach der Multiplikatioon.

Subtraktion und Division sind inverse Operationen. Diese Rechenarten sind jedoch nicht anders, wie man normal die Zahlen in $\mathbb{R}$, $\mathbb{Q}$.

Diese sind wie folgt definiert:

Additive Inverse (Subtraktion): Wenn $a \in \mathbb{Z}_p$, dann ist (-a) die additive Inverse von a in $\mathbb{Z}_p$. Dies ist auch die einzige Lösung zur Gleichung : $a + x \equiv 0$ (mod p) Multiplikative Inverse (Division): Wenn $a \in \mathbb{Z}_p, a \neq 0$, dann ist a^{-1} die multiplikative Inverse von a in $\mathbb{Z}_p$. Dies ist auch die einzige Lösung zur Gleichung : $a \cdot x \equiv 1$ (mod p)

Nehmen wir als Beispiel die Primzahl 2.

$\mathbb{Z}_2 = 0, 1$

Hier hat man lediglich nur 2 Elemente. Rechnet man beispielsweise bei Addition alle Möglichkeiten, sieht es so aus: $0 + 0$; $0 + 1 = 1$; $1 + 1 = 0$ Um es einfacher und schöner zu gestalten erstellen wir eine Tabelle für Addition und Multiplikation.

Tabelle 4.2: Addition und Multiplikation in $\mathbb{Z}_2 = 0, 1$

+	0	1
0	0	1
1	1	2

+	0	1
0	0	0
1	0	1

Tabelle 4.3: Addition und Multiplikation in $\mathbb{Z}_2 = 0, 1$

+	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

·	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

Algorithmus im Programm

Das Programm verläuft folgendermaßen. Zuerst erstellt man ein Objekt mit der gewählten Primzahl. Das Objekt besitzt alle möglichen Rechenoperationen in dem Umfeld. Führt man eine Rechnung, wird überprüft ob die Zahl im jeweiligen Menge ist. Nach der Rechnung wird das Ergebnis mit modulo Prim ausgerechnet und zurückgeschickt.

Die Rechenoperationen werden von der Modulare Arithmetik bereitgestellt und im Grunde werden die Werte weitergegeben.

4.3 Primzahlpotenz

Irreduzible Polynom

Bevor man die Verbindung zu endlichen Körper macht, muss man die Irreduzible Polynome verstehen.
Irreduzible Polynome sind Polynome, welche man nicht mehr faktorisieren kann. Somit haben diese Polynome keine Nullstellen im $\mathbb{F}_p$

Beispiele für Irreduzible Polynome sind:
$x^2 + x + 1$
$x^3 + x^2 + x + 1$
$x^4 + x^3 + x^2 + x + 1$
...

4.3.1 Zusammenhang mit Endlichen Körper

Der Körper K wird folgendermaßen aufgeschrieben : $K := \mathbb{F}_{p^m}$. p für Primzahl und m für den Gradienten. Der Gradient sagt aus, welches Polynom genommen wird. Beispielsweise nimmt man für den Gradient 2 das Polynom $x^2 + x + 1$
Alle Berechnungen werden mit Polynomen ausgeführt und die X-Werte werden nicht ersetzt.
Die Elemente die in $K := \mathbb{F}_{p^m}$ repräsentieren sind:

Allgemein: $\mathbb{F}_{p^m} = \{p_{m-1} \cdot x^{m-1} + p_{m-2} \cdot x^{m-2} + ... + p_1 \cdot x + p_0; p_i \in \{0, 1, ..., p-1\}\}$
Man kann $\mathbb{F}_{p^m}$ auch so schreiben : $\mathbb{F}_{p^m}[x] = \frac{\mathbb{F}_p[x]}{Polynom}$

Wie sehen dann die Polynom-Addition, und Multiplikation aus? Um es einfacher zu verstehen, nutzen wir einen Beispiel.
Wir nehmen den Körper := $\mathbb{F}_{2^2}$
Hierbei wird unser Polynom $x^2 + x + 1$ benutzt. Da der Gradient = 2 ist. Wenn ein Polynom mit einem gleichen oder höheren Gradienten gibt, wird dieser durch Polynomdivision verkleinert und den Rest davon genommen.
bei $x^2 + x + 1$ können nur 4 Möglichkeiten geben. := $\frac{\mathbb{F}_2[x]}{x^2+x+1} = \{0, 1, x, x+1\}$

Macht man eine Tabelle für diese Elemente, kommt folgendes dabei raus:

Tabelle 4.4: TODO CAPTION

+	0	1	x	x+1
0	0	1	x	x+1
1	1	0	x+1	x
x	x	x+1	0	1
x+1	x+1	x	1	0

·	0	1	x	x+1
0	0	0	0	0
1	0	1	x	x+1
x	0	x	x+1	1
x+1	0	x+1	1	x

Algorithmus im Programm

Im Prinzip arbeitet das Programm, wie normal bei den Primzahl. Jedoch wird es mit Polynomen arbeiten, welche in ArrayList<BigInteger> abgespeichert wird.

Zuerst wird der Gradient bestimmt und es wird aus einer Liste paar Beispiel-Irreduzible Polynome genommen. Wenn ein ArrayList übergeben wird, bestimmt die Länge, den Höchsten Gradienten an.
Angenommen ist die Size der Liste q, so ist der Gradient q-1. Zuerst werden die Polynome mit den Grundrechenoperationen berechnet und anschließend über mod Irreduzibles Polynom abgerechnet. Anschließend wird der Rest ausgegeben.

5 Elliptische Kurven

5.1 Darstellungen von Punkten

Um Punkte auf einer elliptischen Kurve darzustellen oder anzugeben gibt es mehrere Möglichkeiten, die Unterschiedliche Vor- und Nachteile haben. Jede Darstellung wird daher auch anders miteinander verrechnet.

5.1.1 Affine Darstellung

Die affine Darstellung ist die gebräuchlichste Art und Weise einen Punkt und eine elliptische Kurve darzustellen, da es nur die x und y Koordinaten gibt.
In diesem Fall gibt es keine einheitliche Darstellung der Unendlichkeit und man muss prüfen, ob die Koordinaten auf der Kurve liegen.

5.1.2 Projektive Darstellung

Bei der projektiven Darstellung kommt die z - Koordinate hinzu, sodass aus der zwei dimensionalen eine drei dimensionale Kurve entsteht. Die Umrechnung eines affinen Punktes in einen projektiven Punkt ist dabei einfach durch das Hinzufügen von z = 1 durchzuführen. Bei der umgekehrten Richtung muss man die x und y Koordinaten durch z teilen, woraus folgt, dass alle Punkte mit z = 0 unendlich sind, da sie nicht auf der Kurve liegen können. Zur Erinnerung, die Formel der Kurve ändert sich von $y^2 = x^3 + Ax + B$ zu $y^2z = x^3 + Axz^2 + Bz^3$
Eine Punktaddition benötigt zwölf Multiplikationen und zwei Quadrierungen, während eine Punktverdopplung nur sieben Multiplikationen und fünf Quadrierungen braucht.

5.1.3 Jacobi Darstellung

Um einen noch größeren Speedup zu erreichen, kann man Jobische Koordinaten verwenden, da diese besonders effizient beim Verdoppeln sind. Hier ist die Darstellung (x,y,z) zu affin $(x/z^2, y/z^3)$ gewählt bei der Kurvenform

$$y^2 = x^3 + Axz^4 + Bz^6$$

Der Punkt unendlich ist definiert mit (1,1,0).

5.1.4 k-fache Punktmultiplikation

Wenn ein Punkt mit k multipliziert wird, unterscheidet man, dass bei $k > 0$

$$P + P + ... + P = kP$$

und bei $k < 0$

$$(-P) + (-P) + ... + (-P) = kP$$

berechnet wird. Durch wiederholtes Verdoppeln erreicht man bei großen k die beste Performance und da die elliptischen Kurven auf endlichen Feldern basieren, muss man sich auch keine Gedanken um immer größere Koordinaten machen, da diese immer $mod p$ gerechnet werden.

5.1.5 Negation

Die Negation wird durch einen Vorzeichenwechsel durchgeführt bei der y-Koordinate. Da die Zahl dann negativ ist, muss noch das positive Gegenstück im endlichen Zahlenraum gefunden werden, was durch die einfache Addition der Zahl p geht. Vorausgesetzt ist hierbei eine affine Darstellung. Kommt man von einer anderen Darstellung muss man zuerst zur affinen Darstellung, dann negieren und abschließend in die ursprüngliche Darstellung umrechnen.

5.2 Punktaddition

5.2.1 affine

Die Addition zweier Punkte bedeutet, dass man eine Gerade durch die Punkte zieht und diese Gerade schneidet die Kurve in einem dritten Punkt, dieser ist das Ergebnis.
Bei zwei unterschiedlichen Punkten, wovon einer nicht im unendlichen Bereich liegt, wird die Steigung durch den Differenzquotienten berechnet:

$$m = x_1 - x_2 / y_1 - y_2$$

Nun braucht man nur noch einzusetzen

$$x = m^2 - x1 - x2$$

$$y = m(x - x_1) - y_1$$

und erhält den neuen Punkt.
Ansonsten gilt $P_1 + \infty = P_1$, wobei ∞ ein Punkt außerhalb der Kurve ist.
Diese Addition ist für den Rechner bei großen Zahlen sehr aufwendig, da man dividieren muss und das die anspruchsvollste Rechenoperation ist. Man benötigt zwei Multiplikationen, eine Quadrierung und eine Division.

5.2.2 projektive

Bei $P1 \neq P2$ sieht die Addition wie folgt aus:

$$u = y_2 z_1 - y_1 z_2$$

$$v = x_2 z_1 - x_1 z_2$$

$$w = u^2 z_1 z_2 - v^3 - 2v^2 x_1 z_2$$

$$x_3 = 2uw$$

$$y_3 = t(4v - w) - 8y_1^2 u^2$$

$$z_3 = 8u^3$$

5.2.3 jakobische

Bei zwei unterschiedlichen Punkten wird wie folgt addiert:

$$r = x1z_2^2$$
$$s = x_2z_1^2$$
$$t = y1z_2^3$$
$$v = s - r$$
$$w = u - t$$
$$x_3 = -v^3 - 2rv^2 + 2^2$$
$$y_3 = -tv^3 + (rv^2 - x_3) * w_1$$
$$z_3 = vz_1z_2$$

Dadurch ergeben sich zwölf Multiplikationen und vier Quadrierungen.
Ein weiterer Vorteil dieser Darstellung ist die Kompatibilität zu affinen Punkten. Hier benötigt man für die Addition von einem Jacobi Punkt und einem affinen Punkt nur acht Multiplikationen und drei Quadrierungen.

5.3 Punktverdopplung

5.3.1 affin

Wenn man den gleichen Punkt mit sich selbst addiert, dann muss wie immer bei der Addition die Steigung berechnet werden.

$$m = 3x^2 + A/2y$$

woraus sich wiederum die x Koordinate

$$m^2 - 2x$$

und y Koordinate berechnet wird

$$y = m(x - x_0) + y$$

5.3.2 projektiv

Wenn man zwei gleiche Punkte addiert, ändert sich die gerade genannte Formel zu:

$$t = Az_1^2 + 3x_1^2$$
$$u = y_1z_1$$
$$v = ux_1y_1$$
$$w = t^2 - 8v$$
$$x_3 = 2uw$$
$$y_3 = t(4v - w) - 8y_1^2u^2$$
$$z_3 = 8u^3$$

mit dem Sonderfall von $P1 = -P2$ folgt $P1 + P2 = \infty$.

5.3.3 jakobi

Wenn man nun aber gleiche Punkte addiert, erkennt man direkt im Vergleich die Einfachheit der Formel:

$$v = 4x_1y_1^2$$

$$w = 3x_1^2 + Az_1^4$$

$$x_3 = -2v + w^2$$

$$y_3 = -8y_1^4 + (v - x_3)w_1$$

$$z_3 = 2y_1z_1$$

. Wodurch man die Operationen auf drei Multiplikationen und sechs Quadrierungen reduziert. Das ist besonders schnell berechenbar für einen Computer.

6 ECDH – Elliptic Curve Diffie Hellman

6.1 Diffie Hellman Schlüsselaustausch

Das Diffie-Hellman-Protokoll wird benutzt um auf einem unsicheren Kanal einen gemeinsamen geheimen Schlüssel zu vereinbaren um damit eine verschlüsselte sichere Kommunikation zu betreiben.
Dafür einigen sich beide Teilnehmer über den öffentlichen Kanal auf eine ausreichend große Primzahl p und eine natürliche Zahl g, die zwischen 1 und $p-1$ liegt und ein Erzeuger von Z_P ist.
Nun muss jeder Teilnehmer eine geheime Zufallszahl generieren, die kleiner p ist. Danach berechnet jeder seinen öffentlichen Schlüssel

$$A = g^a mod p$$

und schickt diesen an den anderen Teilnehmer.
Wenn jeder mit seinem privaten Schlüssel und dem öffentlichen Schlüssel des Gegenübers den gemeinsamen Schlüssel

$$K_1 = B^a mod p$$

$$K_2 = A^b mod p$$

$$K_1 = K_2$$

berechnen.
Um die Kommunikation abzuhören müsste ein Angreifer das diskrete Logarithmusproblem lösen, was bei großen Zahlen nicht effizient möglich ist.

6.2 Schlüsselaustausch bei Elliptischen Kurven

Wie beim klassischen Diffie Hellman Schlüssel Austausch wird bei der Variante mit elliptischer Kurve ein Schlüsselaustausch vorangetrieben. Dabei braucht jeder Teilnehmer einen privaten Schlüssel d, eine zufällige Zahl die zwischen 1 und $n-1$ gewählt wird und einen öffentlichen Schlüssel Punkt Q. Q entsteht durch die folgende Formel

$$Q = d * G$$

mit G als Erzeuger der Kurve.
Mit dem eigenen d und dem G vom Gegenüber kann man den Punkt

$$K = d_a * Q_b$$

$$K = d_b * Q_a$$

berechnen. Von K benötigt man in den meisten Verfahren nur die x-Koordinate.
Sollte ein Teilnehmer einen ungültigen Punkt, der nicht auf der Kurve liegt, wählen und der Andere validiert diesen Punkt nicht, ist es möglich den geheimen Schlüssel des echten Punktes herauszufinden.

7 Fazit

Literatur

[1] .

[2] *Advanced Encryption Standard*.

[3] Menezes Alfred, Oorschot Paul und Vanstone Scott. *Handbuch of Applied Cryptography*. CRC-Press, 1997.

[4] Nada Ali und Esaa Kawther. „Security Improvement in Elliptic Curve Cryptography“. In: *International Journal of Advanced Computer Science and Applications* 9 (Jan. 2018), S. 122–133.

[5] Werner Annette. *Elliptische Kurven in der Kryptographie*. Hrsg. von Springer. Springer-Verlag Berlin Heidelberg GmbH, 2002.

[6] Khan Arif. *Comparative Analysis of Elliptic Curve Cryptography*. Jan. 2017. ISBN: 978-3-330-01788-7.

[7] Hankerson Darrel, López Julio und Menezes Alfred. „Software Implementation of Elliptic Curve Cryptography over Binary Fields“. In: Jan. 2000, S. 1–24. ISBN: 978-3-540-41455-1. DOI: 10.1007/3-540-44499-8_1.

[8] Hankerson Darrel, Vanstone Scott und Menezes Alfred. „Guide to Elliptic Curve Cryptography“. In: *Springer Professional Computing*. 2004.

[9] Michel Douguet und Vincent Dupaquis. „Modulare Reduktion unter Verwendung einer speziellen Form des Modulo“. Deutsch. Pat. DE112009000152T5. 2008. URL: https://patents.google.com/patent/DE112009000152T5/de.

[10] Kossi D. Edoh. „Elliptic curve cryptography: Java implementation“. In: *Information Security Curriculum Development Conference, InfoSecCD 2004*. 2004, S. 88–93.

[11] *Elliptic Curve Crytography*. Technical Guideline. Bonn, Germany: Federal Office for Information Security, Juni 2018.

[12] Samta Gajbhiye, Monisha Sharma und Samir Dashputre. „A Survey Report On Elliptic Curve Cryptography“. In: *International Journal of Electrical and Computer Engineering (IJECE)* 1 (Okt. 2011). DOI: 10.11591/ijece.v1i2.86.

[13] Rahman Hazifur, Azad Saiful und Khan Pathan Al-Sakib. *Practical Cryptography Algorithms and Implementations using C++*. CRC Press, 2015. ISBN: 13: 978-1-4822-2890-8.

[14] *Information Technology Security Techniques- Digital Signatures., with Appendix- part 3: Certificatebased mechanism*.

[15] Steffen Daniel Jensen, Brian Melin Iversen, Rasmus Feldthaus und Markus Neubrand. *Cryptography: Fast Modular Arithmetic*. Aarhus University, 2009.

[16] Merkle Johannes und Lochter Manfred. „Ein neuer Standard für Elliptische Kurven“. In: Mai 2009.

[17] Sheetal Kalra und Sandeep Sood. „Elliptic curve cryptography: Survey and its security applications“. In: *Proceedings of the International Conference on Advances in Computing and Artificial Intelligence, ACAI 2011* (Jan. 2011), S. 102 –106. DOI: 10.1145/2007052.2007073.

[18] Hans Werner Lang. *Montgomery Multiplikation*. Hochschule Flensburg. 2014. URL: https://www.inf.hs-flensburg.de/lang/algorithmen/arithmetik/montgomery-multiplikation.htm.

[19] Anoop MS. „Elliptic curve cryptography-an implementation guide“. In: (2007).

[20] Mihailescu Marius und Nita Stefania. „Elliptic-Curve Cryptography“. In: Jan. 2021, S. 189–223. ISBN: 978-1-4842-6585-7. DOI: 10.1007/978-1-4842-6586-4_9.

[21] Mohamad Afendee Mohamed. „A survey on elliptic curve cryptography“. In: *Applied mathematical sciences* 8 (2014), S. 7665–7691.

[22] Abdalbasit Mohammed und Nurhayat Varol. „A Review Paper on Cryptography“. In: Juni 2019, S. 1–6. DOI: 10.1109/ISDFS.2019.8757514.

[23] Nayak und Rajput. „Cryptography Algorithms – The Science of Information Security: Review Paper“. In: 2017.

[24] Koblitz Neal. „Elliptic Curve Cryptosystems“. In: Bd. 48. 177. 1987. Kap. Mathematics of Computation, S. 203–209.

[25] Barett Paul. „Implementing the Rivest Shamir and Adleman Public Key Encryption Algorithm on a Standard Digital Signal Processor“. In: 1986. Kap. Advances in Cryptology — CRYPTO' 86, S. 311 –323. ISBN: ISBN 978-3-540-18047-0. DOI: doi:10.1007/3-540-47721-7_24.

[26] *Public Key Cryptography for the financial Services Industry: The Elliptic curve Digital SignatureAlgorithm (ECDSA)*. ANSI X9.62, 1999.

[27] Hossain Rownak und Hossain Selim. „Efficient FPGA Implementation of Modular Arithmetic for Elliptic Curve Cryptography“. In: *International Conference on Electrical, Computer and Communication Engineering (ECCE)* (2019), S. 1–6.

[28] Markan Ruchika und Kaur. „Literature Survey on Elliptic Curve Encryption Techniques“. In: 2013.

[29] *STANDARDS FOR EFFICIENT CRYPTOGRAPHY SEC 2: Recommended Elliptic Curve Domain Parameters*. Jan. 2010. URL: https://www.secg.org/sec2-v2.pdf.

[30] Sushanta Sahu und Manoranjan Pradhan. „Implementation of Modular Multiplication for RSA Algorithm“. In: Juli 2011, S. 112 –114. DOI: 10.1109/CSNT.2011.30.

[31] Contini Scott, Çetin Kaya Koç und Colin Walter. „Modular Arithmetic“. In: *Encyclopedia of Cryptography and Security*. Hrsg. von Henk C. A. van Tilborg und Sushil Jajodia. Boston, MA: Springer US, 2011, S. 795–798. ISBN: 978-1-4419-5906-5. DOI: 10.1007/978-1-4419-5906-5_49. URL: https://doi.org/10.1007/978-1-4419-5906-5_49.

[32] Ankita Soni und Nisheeth Saxena. „Elliptic Curve Cryptography: An Efficient Approach for Encryption and Decryption of a Data Sequence“. In: *International Journal of Science and Research (IJSR)* 2 (2013), S. 203–208.

[33] *Standard Specifications for Public Key Cryptography*.

[34] Stolbikova Veronika. „Can Elliptic Curve Cryptography Be Trusted? A Brief Analysis of the Security of a Popular Cryptosystem“. In: *ISACA Journal* 3 (Mai 2016), S. 1–5.

[35] Miller Victor. „Use of elliptic curves in cryptography“. In: LNCS 218, Springer Verlag, 1986. Kap. Advancesin Cryptography-Crypto '85, S. 417 –426.

[36] Hasenplaugh William, Gaubatz Gunnar und Gopal Vinodh. „Fast Modular Reduction“. In: 2007. DOI: doi:10.1109/ARITH.2007.18..

[37] Dietmar Wätjen. *Kryptographie Grundlagen, Algorithmen, Protokolle*. Bd. 3. Springer-Verlag, 2018. DOI: https://doi.org/10.1007/978-3-658-22474-5.

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Abkürzungsverzeichnis