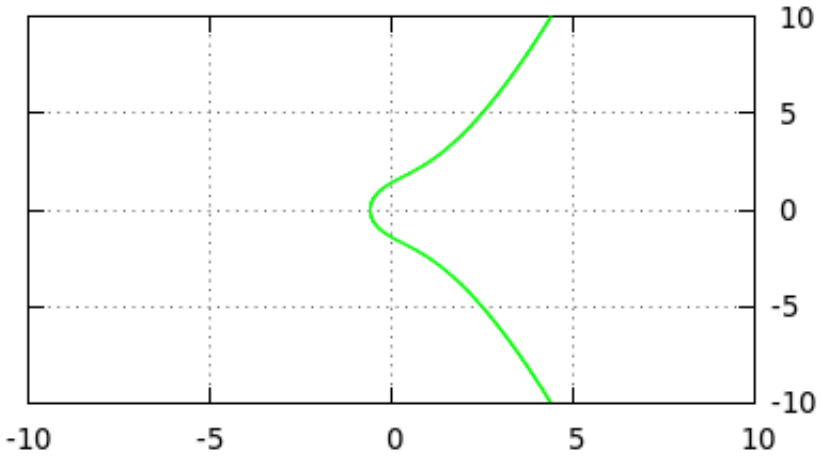
University of Applied Sciences Wedel

Studiengang: Master Informatik

Seminar: Kryptografie mittels elliptischer Kurven



Bearbeitet von: Maurice Tollmien – maurice.tollmien@gmail.com

Eingereicht am: 11. Januar 2016

FH-Wedel Betreuer: Prof. Dr. Michael Anders

Inhaltsverzeichnis

1 Einleitung

Die folgende Ausarbeitung dient dem Ziel, einen allgemeinen Überblick zu schaffen über die Möglichkeit und Vorgehensweise des asymmetrischen Schlüsselaustauschs und Signatur mittels elliptischer Kurven.

Weiterhin wird die Fragestellung aufgegriffen, ob sich Kryptografie mittels elliptischer Kurven aus Sicht von Effizienz- und Sicherheitsgründen besser eignet, als andere asymmetrische, kryptografische Verfahren und wie sich bei gleicher Anzahl an Sicherheitsbits die Schlüssellänge der verschiedenen Algorithmen verhält.

Die Motivation für eine weitere Forschung neuer oder unterschiedlicher kryptografischer Methoden, abseits der bekannten und verwendeten, ist besonders heutzutage wichtig, da es sehr schwer abzusehen ist, wie lange bekannte Verfahren noch sicher verwendet werden können.

In der folgenden Arbeit werden keine neuen Erkenntnisse auf dem Gebiet der Kryptografie mittels elliptischer Kurven gewonnen. Vielmehr geht es darum, das Konzept der Kryptografie mittels elliptischer Kurven vorzustellen und dieses anderen Verfahren gegenüber zu stellen.

2 Zur Erinnerung

In den folgenden Abschnitten werden einige grundlegende Algorithmen und Probleme erläutert, welche im Kontext der kommenden Kapitel genutzt werden und essentiell sind für die Funktionalität und Effektivität vieler kryptografischer Verfahren, wie RSA und Methoden auf Basis elliptischer Kurven.

Es wird ein Grundverständnis der diskreten Mathematik und der algebraischen Gruppentheorie für die folgenden Kapitel vorausgesetzt.

2.1 Prime Restklassengruppen

Bei der Erzeugung asynchroner Verschlüsselungen, zu denen auch die Verfahren elliptischer Kurven gehören, erfolgen die Berechnungen im Kontext algebraischer Gruppen. Eine Gruppe ist eine endliche Menge an Elementen, kombiniert mit einem Operator. Also sei $(G, \cdot)$ eine Gruppe. Die Ordnung der Gruppe G entspricht der Anzahl der Elemente in G. Die Ordnung eines Elementes $\alpha \in G$ ist die niedrigste Zahl n für die gilt: $\alpha^n = e \; mod \; n$ mit $e =$ dem neutralen Element bezüglich des Operators $\cdot$ in G.

Jede (zyklische) Gruppe besitzt einen Generator α, für den gilt, dass $\forall a \in G \; \exists i \in \mathbb{N}$: $\alpha^i = \alpha \cdot \alpha \cdot \cdots \alpha = a$. Sofern G eine zyklische endliche Gruppe ist und α ein Generator dieser, so entspricht die Ordnung von α der von G.

Ein Beispiel einer primen Restklassengruppe ist $(\mathbb{Z}_p, \cdot)$, mit der Voraussetzung, dass der Generator p eine Primzahl ist. Die Ordnung dieser Gruppe ist $p - 1$.

Alle Operationen asynchroner Verschlüsselungen finden im Kontext meist primer Restklassengruppen statt. Hier muss keine weitere Überprüfung auf Eindeutigkeit und Zyklen erfolgen, da der Generator, eine Primzahl, die gewünschten Eigenschaften der Gruppe bereits vorgibt.

2.2 Diskretes Logarithmus Problem (DLP)

Sei G eine zyklische Gruppe und $g \in G$ ein erzeugendes Element. Gegeben sei ein Element h in der durch g generierten Teilgruppe. Das diskrete Logarithmus Problem für G besteht nun darin, ein Element m zu finden, welches die Gleichung $h = g^m$ erfüllt. Der kleinste Wert m, welches $h = g^m$ erfüllt, ist der sogenannte Logarithmus von h im Bezug auf g. Also: $m = log_g(h)$.

Das diskrete Logarithmus Problem wird in der Kryptografie häufig als das zu Grunde liegende Problem für asynchronen Schlüsseltausch, digitale Signaturen oder Hash-Funktionen genutzt. Es eignet sich zur Nutzung in kryptografischen Systemen durch den Umstand, dass sich, im Sinne der Komplexitätstheorie, der diskrete Logarithmus nur sehr ineffizient berechnen lässt, während die Umkehrfunktion (auch im Sinne der Komplexitätstheorie) einfach berechenbar ist. Diese Art von Funktion nennt sich auch Einwegfunktion/Falltürfunktion und bildet die mathematische Grundlage aller asynchronen Verfahren der Kryptografie.

3 Elliptische Kurven

3.1 Anforderungen an Elliptische Kurven

3.1.1 Diskretes Logarithmus Problem für elliptische Kurven (ECDLP)

In der definierten Gruppe einer elliptischen Kurve mit einer multiplikativen Notation ist das diskrete Logarithmus Problem wie folgt definiert.

Gegeben seien P und Q einer Gruppe definiert über einer elliptischen Kurve. Folgende Gleichung muss gelöst werden: $P * k = Q$. k wird der diskrete Logarithmus von Q auf Basis von P genannt.

3.1.2 Anforderungen an asynchrone kryptografische Verfahren

Asynchrone Verfahren der Kryptografie basieren auf dem Prinzip, dass zwei Kommunikationspartner verschlüsselt miteinander kommunizieren können, ohne ein gemeinsames Geheimnis oder Schlüssel abgesprochen zu haben. Zwei entscheidende Anforderungen dabei sind, dass ein beliebiger Angreifer weder in der Lage sein soll die Kommunikation mitzulesen, noch sie zu semantisch korrekt zu manipulieren. Beide Anforderungen werden mit Kryptografie elliptischer Kurven abgedeckt.

3.2 Allgemeine Funktionsweise

In klassischen Methoden asymmetrischer Verschlüsselungen wie Diffie-Hellman und ElGamal basiert die Verschlüsselung auf endlichen, arithmetischen Feldern modulo einer großen Primzahl p oder p^n. Dabei werden die Vorteile und die Falltür-Funktionalität des diskreten Logarithmus-Problem genutzt. (siehe Kapitel: 2.2 Diskretes Logarithmus Problem (DLP))

Im Kontext der Verschlüsselung mittels elliptischer Kurven wird eine algebraische Gruppe über Punkte auf einer elliptischen Kurve definiert. Die elliptische Kurve erfüllt die folgende Kurvengleichung:

$$y^2 = x^3 + a * x + b$$

Dabei sind die Koeffizienten a und b fest definiert. Eine Kurve mit den Koeffizienten $a = -1$ und $b = 0$ ergibt folgende elliptische Kurve:

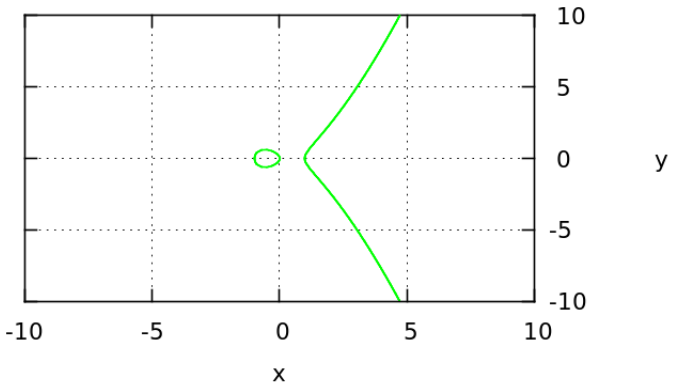


Abbildung 3.1: $y^2 = x^3 - x$

Eine Kurve mit $a = 1$ und $b = 1$ ergibt folgende elliptische Kurve:

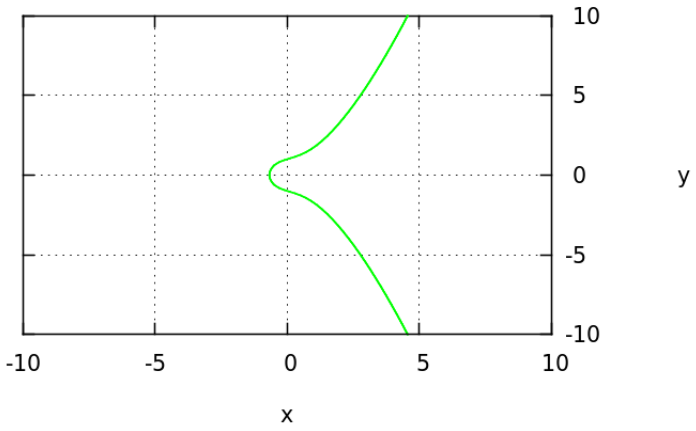


Abbildung 3.2: $y^2 = x^3 + x + 1$

Eine elliptische Kurve ist demnach definiert über die zwei kurven-lokale Konstanten a und b. Die Schreibweise zur Definition einer elliptischen Kurve ist $E(a; b)$. Punkte auf der elliptischen Kurve sind definiert über ihre ganzzahligen x und y-Koordinaten. Also: $p = (x; y)$.

Um eine algebraische Gruppe über Punkten einer elliptischen Kurve definieren zu können, muss die additive Operation auf Elementen der Gruppe definiert werden und ein neutrales Element existieren.

Für jede elliptische Kurve gilt, dass jede Gerade, definiert durch zwei Punkte der Kurve, immer genau einen dritten Schnittpunkt mit der elliptischen Kurve besitzt.

Seien nun P und Q Punkte auf einer elliptischen Kurve, so existiert ein Schnittpunkt der Geraden durch P und Q, mit einem weiteren Punkt R der Kurve. Wenn $P = Q$, ist die Gerade definiert als die Tangente am Punkt P. Da es exakt drei Schnittpunkte einer so definierten Gerade mit der elliptischen Kurve gibt, ist die Existenz und Einzigartigkeit von R garantiert.

Der Punkt O sei ein weiterer Punkt, welcher die elliptische Kurvengleichung erfüllt. Er gilt als das neutrale Element und ist per Konvention ein Punkt im Unendlichen.

Betrachtet man nun eine Gerade zwischen R und O, so schneidet diese einen dritten Punkt der Kurve. Dieser ist definiert als $P + Q$.

Sei OO der Schnittpunkt der Tangente an O mit der Kurve, wird der Schnittpunkt der Geraden zwischen OO und P mit der elliptischen Kurve nun $-P$ genannt[8].

Die folgenden Gleichungen belegen die genannten Berechnungen:

$$P + Q = Q + P$$
$$P + O = P$$
$$(-P) + P = O$$

Zusammengefasst gelten folgende Definitionen für Operationen auf der Gruppe definiert durch Punkte auf der elliptischen Kurve:

Negation:

Die Negation eines Punktes auf einer Kurve ist definiert über die Negation der y-Komponente des Punktes. Also: $P = (x; y)$ und: $-P = (x; -y)$.

Addition:

Eine Addition zweier Punkte P und Q auf einer Kurve ist definiert, indem eine Gerade zwischen P und Q gezogen wird. Der dritte Schnittpunkt der Geraden mit der Kurve (erster Schnittpunkt auf P, zweiter Schnittpunkt auf Q) wird nun negiert. Der nun errechnete Punkte bildet das Ergebnis der Addition.

Punkt verdoppeln:

Ein Punkt P auf der Kurve wird verdoppelt, indem die Tangente an P mit der Kurve geschnitten wird. Der Schnittpunkt ist nun $-(P + P)$.

Multiplikation:

Mit den aktuell definierten Operationen auf elliptischen Kurven entsteht eine Gruppe. Eine multiplikative Operation ist nicht definiert und auch nicht trivial zu konstruieren. Im weiteren Verlauf wird jedoch eine Integer-Multiplikation benötigt um einen asynchronen Schlüsselaustausch zu ermöglichen.

Da ausschließlich multiplikative Operationen eines Punktes mit einem Integer erforderlich sind, wird eine Multiplikation im Folgenden durch eine wiederholte Addition eines

Punktes simuliert. Eine Multiplikation zweier Punkte ist weiterhin nicht definiert.

Finite elliptische Kurven können über $\mathbb{Z}_p$ oder $GF(p^n)$ mit p als Primzahl definiert werden. Im Kontext der Kryptografie werden jedoch meist nur Kurven über $\mathbb{Z}_p$ und $GF(2^n)$ genutzt. Der Einfachbarkeitshalber sind alle Beispiele über $\mathbb{Z}_p$ definiert.

3.3 Asynchroner Schlüsseltausch

Asynchroner Schlüsseltausch wird gemeinhin genutzt, um einen Schlüssel für eine weitere synchrone Verschlüsselung über einen unsicheren Kanal zu transportieren. Die bekannten Algorithmen und Schemata, wie Diffie-Hellman, stützen direkt auf das diskrete Logarithmus-Problem. Die gleichen Prinzipien können jedoch auch mit Hilfe elliptischer Kurven erreicht werden. Dabei handelt es sich um analoge Verfahren zu ebendiesen und nicht um die exakten.

Im Weiteren werden beide Teilnehmer, zwischen denen ein Schlüsselaustausch stattfinden soll, mit *Alice* und *Bob* referenziert.

Die folgenden Attribute gelten für alle vorzustellenden Algorithmen:

Als E wird die elliptische Kurve bezeichnet, aus $GF(q)$, wobei $q = p^n$ eine bevorzugt große Zahl darstellt. Die Kurvenparameter der genutzten Kurve sind öffentlich. Weiterhin muss eine eindeutige, öffentlich zugängliche Funktion existieren, welche eine Nachricht m auf einen Punkt P_m der elliptischen Kurve abbildet: $f : m \to P_m$.

3.3.1 Diffie-Hellman

Ein analoger Algorithmus zu Diffie-Hellman, basierend auf elliptischen Kurven, geht folgenderweise vor.

Öffentlich bekannt ist ein gemeinsamer Punkt G (erzeugendes Element der Gruppe) auf der elliptischen Kurve E. Beide Teilnehmer des Schlüsselaustausches erstellen ihr jeweiliges Schlüsselpaar, bestehend aus einem privaten Schlüssel d, mit $0 < d < N$ und $ggt^1(d, N) = 1$ mit $N = |E|$ und einem öffentlichen Schlüssel Q, mit $Q = d * G$.

Alice' Schlüsselpaar: (d_A, Q_A)

Bobs Schlüsselpaar: (d_B, Q_B)

Der öffentliche Part des Schlüsselpaares ist jeweils beiden Kommunikationsteilnehmern bekannt. Alice berechnet nun $d_A * Q_B = (x_k, y_k)$. Bob berechnet $d_B * Q_A = (x_k, y_k)$. Die X-Koordinate x_k des Ergebnispunktes auf der elliptischen Kurve ist nun das gemeinsame Geheimnis, welches ausschließlich Alice und Bob bekannt ist.

Die Gültigkeit, dass beide Teilnehmer das gleiche Ergebnis berechnen zeigt sich durch folgende Gleichungsauflösung: $d_A * Q_B = d_A * d_B * G = d_B * d_A * G = d_B * Q_A$.

Um aus dem gemeinsamen Geheimnis einen Schlüssel zu erzeugen, wird die X-Koordinate x_k des Ergebnispunktes meist kryptografisch gehashed und als symmetrischer Schlüssel verwendet.

3.3.2 Massey-Omura

Das Schema von Massey-Omura ist ein von Diffie-Hellman motivierter Algorithmus für den asynchronen Schlüsseltausch.

Die Vorgehensweise ist wie folgt. Alice generiert sich einen geheimen Wert c mit $0 < c < N$ und $ggt(c, N) = 1$. Alice überträgt nun die Nachricht $c * P_m$ an Bob. Bob generiert sich einen geheimen Wert d mit $0 < d < N$ und $ggt(d, N) = 1$ und überträgt die Nachricht $d * (c * P_m)$ an Alice. Alice antwortet Bob mit der Nachricht $c' * (d * c * P_m) = d * P_m$ wobei gilt, dass $(c * c') = 1 \ mod \ N$. Bob berechnet nun $d' * d * P_m = P_m$ und erhält die ursprüngliche Nachricht P_m. Auch hier gilt, dass $(d * d') = 1 \ mod \ N$.

[1] Größter gemeinsamer Teiler (auch gcd)

3.3.3 ElGamal

Auch das Schema von ElGamal lässt sich als Variation des Originalen mit elliptischen Kurven als Grundlage nutzen. Öffentlich bekannt ist, wie bei Diffie-Hellman, ein gemeinsamer Punkt G (erzeugendes Element der Gruppe) auf der elliptischen Kurve E.

Bob generiert sich einen geheimen Wert d und veröffentlicht den Punkt auf der Kurve $d*G$. Alice generiert sich einen Wert c und sendet Bob das Tupel $(c*G, P_m + c*(d*G))$. Bob kann nun den ersten Teil des übertragenen Tupels mit seinem geheimen Wert d multiplizieren und vom zweiten Teil des Tupels abziehen. Das Ergebnis der Subtraktion ist die eigentliche geheime Nachricht P_m.

Die Auflösung lässt sich zeigen durch: $(P_m + c * (d * G)) - (d * (c * G)) = P_m$.

3.4 Signieren mit elliptischen Kurven (ECDSA[2])

Wie auch mit anderen asynchronen Verfahren zur Verschlüsselung, ist es oft notwendig, Nachrichten zu signieren und zuverlässig auf Authentizität prüfen zu können. Der Signier-Algorithmus mit elliptischen Kurven ist dabei eine Abwandlung des normalen DSA[3].

Die Ausgangssituation ist, dass Alice Bob eine Nachricht schickt, welche Bob auf Authentizität prüfen möchte. Beide Kommunikations-Teilnehmer einigen sich im Vorhinein auf eine elliptische Kurve C, ein generierendes Element G aus C (Punkt auf der Kurve) und eine prime, multiplikative Ordnung n des Generators G. Dabei gilt $n * G = O$.

Alice generiert oder besitzt ein Schlüsselpaar mit einem privaten Schlüssel d_A (zufälliger Integer aus dem Interval $[1, n-1]$) und einen öffentlichen Schlüssel (Punkt auf der Kurve C): $Q_A = d_A * G$.

[2]Elliptic Curve Digital Signature Algorithm
[3]Digital Signature Algorithm

3.4.1 Signieren einer Nachricht

Das Erzeugen einer Signatur erfolgt in folgenden Schritten.

(1) $e = HASH(m)$
Alice erzeugt einen kryptografisch sicheren Hash aus der Nachricht m.
(2) z seien die n linkesten Bits aus e, mit n gleich der Bitlänge der Gruppenordnung.
(3) Alice berechnet einen kryptografisch sicheren Zufallswert k aus dem Interval $1, n-1$.
(4) $(x_1, y_1) = k * G$
Alice berechnet den Kurvenpunkt aus $k * G$.
(5) $r = x_1 \ mod \ n$
$s = k^{-1}(z + r * d_A) \ mod \ n$

Die Signatur besteht nun aus dem Tupel (r, s).

3.4.2 Prüfen einer Signatur

Vor dem eigentlichen Prüfen einer Signatur auf Authentizität erfolgt üblicherweise eine Prüfung, ob es sich um eine potentiell valide Signatur handelt. Dazu gehören Checks, ob es sich bei Q_A um einen Punkt auf der Kurve handelt (mit korrekten Koordinaten) und dieser nicht dem neutralen Element O entspricht. Zusätzlich kann geprüft werden, ob $n * Q_A = O$ gilt.

Das Prüfen, ob eine empfangene Signatur korrekt ist, erfolgt auf folgende Weise.

(1) Verifizieren, dass es sich bei den Signaturwerten (r, s) um korrekte Integer-Werte im Bereich $[1, n-1]$ handelt.
(2) $e = HASH(m)$
Bob erzeugt den gleichen kryptografisch sicheren Hash aus der Nachricht m.
(3) z seien die L_n linkesten Bits aus e, mit L_n gleich der Bitlänge der Gruppenordnung n.
(4) $w = s^{-1} \ mod \ n$
$u_1 = z * w \ mod \ n$
$u_2 = r * w \ mod \ n$
(5) $(x_1, y_1) = u_1 * G + u_2 * Q_A$
Bob berechnet nun einen Punkt auf der Kurve mit den Koordinaten (x_1, y_1).

Die Signatur ist valide, wenn gilt, dass: $r \equiv x_1 \ mod \ n$

3.4.3 Korrektheit des Algorithmus'

Die Korrektheit der Berechnung lässt sich durch folgende Umstellungen der ursprünglichen und zu überprüfenden Formeln zeigen.

Der Punkt D sei der Punkt auf der Kurve, welcher in der Berechnung der Prüfung der Signatur in Schritt 5 berechnet wurde (siehe Kapitel: 3.4.2 Prüfen einer Signatur):

$D = u_1 * G + u_2 * Q_A$

Aus der Definition des öffentlichen Schlüssels von Alice, lässt sich $Q_A = d_A * G$ einsetzen:

$D = u_1 * G + u_2 * d_A * G$

Die Formel lässt sich zu folgender zusammenfassen:

$D = (u_1 + u_2 * d_A) * G$

Setzt man nun die Definitionen für u_1 und u_2 (siehe Schritt 4 in 3.4.2 Prüfen einer Signatur) und formt die Formel entsprechend um, bekommt man:

$D = (z * w + r * w * d_A) * G$

$D = (z * s^{-1} + r * s^{-1} * d_A) * G$

$D = (z + r * d_A) * s^{-1} * G$

Durch ein Einsetzen der Werte für s (Siehe Schritt 5 in Kapitel 3.4.1 Signieren einer Nachricht) wird folgende Formel erreicht:

$D = (z + r * d_A) * (k^{-1}(z + r * d_A))^{-1} * G$

$D = (z + r * d_A) * k^{-1^{-1}} * (z + r * d_A)^{-1} * G$

Da sich $k^{-1^{-1}}$ zu k reduziert und das Produkt einer Inversen und dem Original das neutrale Element ergeben, ergibt sich folgende Formel:

$D = k * G$

Welche genau dem Kurvenpunkt entspricht, der als r im Signaturtupel mitgegeben wird (Siehe Schritt 4 in Kapitel 3.4.1 Signieren einer Nachricht).

3.5 Komplexität und Berechenbarkeit von ECC

Wie in den vorherigen Kapiteln (siehe Kapitel: 3.2 Allgemeine Funktionsweise) erwähnt und erläutert, basiert die Komplexität von Kryptografie über elliptische Kurven auf dem diskreten Logarithmus-Problem für ebendiese (ECDLP[4]). Zum aktuellen Zeitpunkt sind noch keine Algorithmen bekannt, welche dieses Problem effizient oder in subexponentieller Zeit lösen. Die effizientesten derzeit bekannten Algorithmen basieren auf den Pollard-ρ und Pollard-λ-Methoden[7].

Die Pollard-ρ-Methode benötigt etwa $\sqrt{\pi * n/2}$ Schritte, die Pollard-λ-Methode etwa $2 * \sqrt{n}$ Schritte. Ein Schritt entspricht grob einer eigenständigen Gruppenoperation auf der elliptischen Kurve. Beide Methoden eignen sich sehr gut zum Parallelisieren. Verschiedene Forschungen haben ergeben, dass beide Pollard-Methoden um kleinere Faktoren schneller umgesetzt werden können[7].

Die nachfolgende Tabelle stellt die Komplexität des Lösens des ECDLP für elliptische Kurven als benötigte Zeit zum Berechnen des diskreten Logarithmus-Problem dar, abhängig der jeweiligen Schlüssellänge.

Schlüssellänge in bits	Pollard-ρ	MIPS years[5]
160	2^{80}	$8.5 * 10^{11}$
192	2^{96}	$5.6 * 10^{16}$
224	2^{112}	$3.7 * 10^{21}$
256	2^{128}	$2.4 * 10^{26}$
384	2^{192}	$4.4 * 10^{45}$
521	2^{260}	$1.3 * 10^{66}$

[4] elliptic-curve discrete logarithm problem
[5] 1 Jahr mit 1.000.000 Instruktionen/Sekunde

3.6 Unsichere Kurven

Die Forschung, welche elliptischen Kurven möglicherweise Unsicherheiten beinhalten oder besonders anfällig für bestimmte Angriffe sind, ist hochaktuell. Jedoch steht fest, dass es Unterschiede bezüglich der Sicherheit von elliptischen Kurven gibt, abhängig ihrer Kurvenparameter. Drei mögliche Klassen von Kurvenparametern/Kurven, welche zu schwächeren Kurven führt werden im Folgenden behandelt und erläutert.

In die erste Klasse an angreifbaren elliptischen Kurven fallen Kurven E über $\mathbb{F}_q$ mit einem n, welches $q^B - 1$ teilt. Für kleine B sind die Kurven angreifbar, wie von Menezes, Okamoto und Vanstone beschrieben[1]. Der Angriff reduziert das ECDLP auf das klassische, traditionelle diskrete Logarithmus-Problem in eine Teilgruppe von $\mathbb{F}_q$.

Eine zweite angreifbare Klasse elliptischer Kurven sind Kurven E über $\mathbb{F}_q$ mit $\#E(\mathbb{F}_q) = q$. Semaev[11], Smart[10], Satoh and Araki[10] beschreiben einen Angriff auf Kurven dieser Art. Dabei kann die Gruppe der elliptischen Kurve effizient auf eine additive Gruppe von $\mathbb{F}_q$ abgebildet werden.

Die dritte Klasse beschreibt Kurven definiert über $\mathbb{F}_q$ mit $q = 2^m$ und einem zusammengesetzten m. Die Angriffe erfolgen über sogenanntes "Weil-Descent"[12]. Sie sind noch immer Gegenstand aktueller Forschungen.

Generell sind Kurven für die praktische oder theoretische Angriffsmöglichkeiten existieren, zu meiden.

3.7 Gegenüberstellung von Schlüssellängen

Im Folgenden werden die Schlüssellängen verschiedener synchroner und asynchroner kryptografischer Algorithmen bezüglich ihrer Sicherheitsbits miteinander verglichen.[4] Wie klar zu erkennen ist, benötigt eine Verschlüsselung/Schlüsselübergabe für eine gleiche Anzahl an Sicherheitsbits deutlich kürzere Schlüssel bei elliptischen Kurven. Damit einher gehend kann eine Schlüssel-Generierung mit elliptischen Kurven deutlich schneller vonstatten gehen, als beispielsweise eine Schlüssel-Generierung bei RSA. Dies lässt sich jedoch leider nicht auf den Schlüsselaustausch verallgemeinern. Hier ist RSA noch schneller, als elliptische Kurven.

Sicherheits-bits	Symmetrische Algorithmen	FFC[6] zB. DSA, D-H	IFC[7] zB. RSA	ECC[8] zB. ECDSA
80	2TDEA	$L = 1024$ $N = 160$	$k = 1024$	$f = 160 - 223$
112	3TDEA	$L = 2048$ $N = 224$	$k = 2048$	$f = 224 - 255$
128	AES-128	$L = 3072$ $N = 256$	$k = 3072$	$f = 256 - 283$
192	AES-192	$L = 7680$ $N = 384$	$k = 7680$	$f = 284 - 511$
256	AES-256	$L = 15360$ $N = 512$	$k = 15360$	$f = 512+$

[6]Finite field cryptography
[7]Integer-factorization cryptography
[8]Elliptic curve cryptography

Literaturverzeichnis

[1] T. Okamoto A. J. Menezes and S. A. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. Technical report, EEE Transactions on Information Theory, 39:1639–1646, 1993.

[2] Rene Algesheimer. *Elliptische Kurven als alternatives Public Key-Verfahren im Homebanking-Standard HBCI*. `www.diplom.de`, 2000.

[3] Daniel R. L. Brown Certicom Research. Standards for Efficient Cryptography; SEC 1: Elliptic Curve Cryptography. `http://www.secg.org/sec1-v2.pdf`, 2009.

[4] William Burr William Polk Elaine Barker, William Barker and Miles Smid. COMPUTER SECURITY -Recommendation for Key Management – Part 1: General. `http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57_part1_rev3_general.pdf`, 2012.

[5] Neal Koblitz. Elliptic Curve Cryptosystems. `http://www.ams.org/journals/mcom/1987-48-177/S0025-5718-1987-0866109-5/S0025-5718-1987-0866109-5.pdf`, 1987.

[6] Uwe Krieger. Elliptische Kurven – Basis für ein alternatives Public Key Kryptosystem. `http://www.ecc-brainpool.org/art_it.pdf`, —.

[7] Uwe Krieger. Elliptische Kurven – Basis für ein alternatives Public Key Kryptosystem. `http://www.secg.org/sec1-v2.pdf`, O.J.

[8] Ben Lynn. Explicit Addition Formulae. `https://crypto.stanford.edu/pbc/notes/elliptic/explicit.html`, 1999.

[9] NSA. Suite B Implementer's Guide to NIST SP 800-56A. `https://www.nsa.gov/ia/_files/SuiteB_Implementer_G-113808.pdf`, 2009.

[10] T. Satoh and K. Araki. Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves. Technical report, Commentarii Mathematici Universitatis Sancti Pauli, 47:81–92, 1998.

[11] I. A. Semaev. Evaluation of discrete logarithms in a group of p-torsion points of an elliptic curve in characteristic p. Technical report, Mathematics of Computation, 67:353–356, 1998.

[12] Nigel Smart. Weil Descent Program. `http://www.cs.bris.ac.uk/~nigel/weil_descent.html`, O.J.

[13] tylo. ElGamal with elliptic curves. `http://crypto.stackexchange.com/questions/9987/elgamal-with-elliptic-curves`, 2013.