ELLIPTISCHE KURVEN IN DER KRYPTOLOGIE

26.06.2002

Stefan Vigerske[1]

Inhaltsverzeichnis

[1]vigerske@informatik.hu-berlin.de http://www.informatik.hu-berlin.de/~vigerske

Teil 1. Elliptische Kurven: Grundlagen

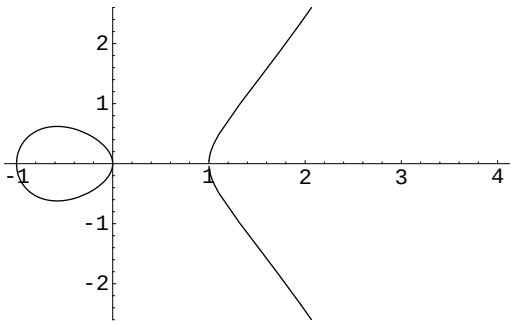
1. Elliptische Kurven über $\mathbb{R}$

Um die Bedeutung elliptischer Kurven für die Kryptographie zu verstehen, betrachten wir zunächst elliptische Kurven über dem Körper der reellen Zahlen, da dieser eine sehr gute Anschauung bietet. In späteren Abschnitten können wir dann unsere Erkenntnisse auf einen beliebigen, vorzugsweise endlichen, Körper verallgemeinern.

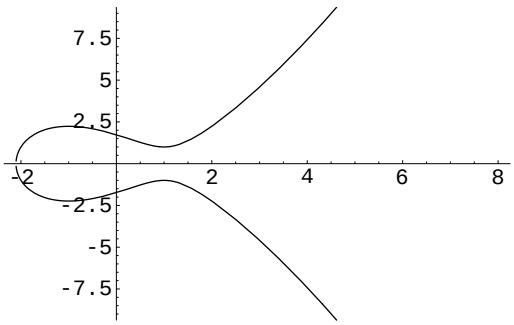
Definition 1.1. Eine elliptische Kurve E über $\mathbb{R}$ (oder allgemeiner über einem Körper mit Charakterstik[2] $\neq 2, 3$) ist eine Menge von Punkten $(x, y) \in \mathbb{R}^2$, die die folgende Gleichung erfüllen

$$y^2 = x^3 + ax + b \quad a, b \in \mathbb{R}, \tag{1.1}$$

zusammen mit einem „unendlich fernen Punkt", der mit $\mathcal{O}$ bezeichnet wird. Wir fordern zusätzlich, daß das Polynom $x^3 + ax + b$ keine mehrfachen Nullstellen haben darf.



Die elliptische Kurve $y^2 = x^3 - x$



Die elliptische Kurve $y^2 = x^3 - 3x + 3$

Das entscheidene an dieser Ansammlung von Punkten in der affinen Ebene ist die Tatsache, daß wir auf diesen Punkten eine Gruppenoperation definieren können. Es war Jacobi (1835) der als erster in „De usu Integralium Ellipticorum et Integralium Abelianorum in Analysi Diophantea" eine solche vorschlug.

Schauen wir uns zunächst eine Gerade g :

$$y = \alpha x + \beta$$

an, die zwei verschiedene Punkte $P = (x_1, y_1)$ und $Q = (x_2, y_2)$ der Ellipse schneidet.

Dann ist

$$\begin{aligned} \alpha &= \frac{y_2 - y_1}{x_2 - x_1} \\ \beta &= y_1 - \alpha x_1 \end{aligned}$$

Ein Punkt von g, d.h. $(x, \alpha x + \beta)$, liegt nun genau dann auf der elliptischen Kurve E (1.1), wenn

$$(\alpha x + \beta)^2 = x^3 + ax + b.$$

Die Punkte von $g \cap E$ sind also die Nullstellen des kubischen Polynoms

$$x^3 - (\alpha x + \beta)^2 + ax + b = 0.$$

Zwei dieser Nullstellen kennen wir bereits, das sind die x-Koordinaten von $P = (x_1, \alpha x_1 + \beta)$ und $Q = (x_2, \alpha x_2 + \beta)$. Ist x_3 die dritte Nullstellen, dann läßt sich das Polynom in Linearfaktoren zerlegen und es gilt

$$x^3 - (\alpha x + \beta)^2 + ax + b = (x - x_1)(x - x_2)(x - x_3)$$

[2]Die Charakteristik eines Körpers mit Einselement e ist die kleinste Zahl $n \in \mathbb{N}$, $n > 0$, für die $n \cdot e = 0$ ist, bzw. 0, falls keine solche Zahl existiert.

Durch Koeffizientenvergleich (für den Koeffizienten vor x^2) erhalten wir dann

$$-x_1 - x_2 - x_3 = -\alpha^2,$$

also $x_3 = \alpha^2 - x_1 - x_2$. Damit sind die Koordinaten eines dritten Punktes $(x_3, y_3) \in g \cap E$ gegeben durch

$$x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2 \tag{1.2}$$

$$y_3 = \alpha x_3 + \beta = \alpha x_3 + y_1 - \alpha x_1 = y_1 + \frac{y_2 - y_1}{x_2 - x_1} \cdot (x_3 - x_1) \tag{1.3}$$

Im Falle $x_1 = x_2$, also falls die Gerade parallel zur y-Achse verläuft, bezeichnen wir (x_3, y_3) mit $\mathcal{O}$.

Diese Herleitung zeigt, daß zu zwei gegebenen Punkten P und Q auf E, ein dritter Schnittpunkt von E mit der Geraden durch P und Q existiert. Dies liefert uns aber leider noch keine Gruppenoperation, wir müssen den erhaltenen dritten Schnittpunkt noch an der x-Achse spiegeln, damit die Gruppenaxiome erfüllt werden.

Die Begriffe Gerade und Tangente lassen sich auch rein algebraisch formulieren, so daß die oben hergeleitete Schnitteigenschaft auch über beliebigen Körpern erhalten bleibt. Dann bleibt auch die folgende definierte Operation im allgemeinen Fall (mit leichten Modifikationen) erhalten.

Definition 1.2. Sei E eine elliptische Kurve über einem Körper $\mathbb{K}$ (mit Charakteristik $\neq 2, 3$). Seien P und Q beliebige Punkte auf E, $\mathcal{O}$ der unendlich ferne Punkt. Dann sei

(1) $\mathcal{O} + P := P$ und $P + \mathcal{O} := P$ ($\mathcal{O}$ dient als neutrales Element.)
(2) $-\mathcal{O} := \mathcal{O}$
(3) Ist $P = (x_1, y_1) \neq \mathcal{O}$, dann setze $-P := (x_1, -y_1)$.
(4) Ist $P = -Q$, so setze $P + Q := \mathcal{O}$.
(5) Sind $P, Q \neq \mathcal{O}$, $P \neq -Q$ und R der dritte Schnittpunkt der Geraden durch P und Q falls $P \neq Q$ bzw. der Tangenten an die Kurve im Punkt P falls $P = Q$, mit der Kurve. Dann setze $P + Q := -R$.

Theorem 1.3. *$(E, +)$ ist eine abelsche Gruppe mit neutralem Element $\mathcal{O}$.*

2. Elliptische Kurven über beliebigen Körpern $\mathbb{K}$

Nach der anschaulichen Einführung im vorigen Abschnitt wollen wir die elliptischen Kurven etwas formaler vorstellen. Dazu benötigen wir den Begriff der projektiven Ebene:

Definition 2.1. Die projektive Ebene $\mathbb{P}^2(\mathbb{K})$ über einem Körper $\mathbb{K}$ ist die Menge von Punkten $(x, y, z) \in \mathbb{K}^3 \setminus \{(0, 0, 0)\}$ zusammen mit der Äquivalenzrelation $\sim$, wobei $(x_1, y_1, z_1) \sim (x_2, y_2, z_2)$ genau dann, wenn ein $\lambda \in \mathbb{K} \setminus \{0\}$ existiert mit $x_1 = \lambda x_2$, $y_1 = \lambda y_2$, $z_1 = \lambda z_2$. Die Äquivalenzklasse von (x, y, z) wird mit $(x : y : z)$ bezeichnet.

Beispiel. Man kann sich $\mathbb{P}^2(\mathbb{R})$ als die Menge aller Geraden im $\mathbb{R}^3$ vorstellen, die durch den Nullpunkt gehen. Beachte dabei, daß $(0, 0, 0) \notin \mathbb{P}^2(\mathbb{R})$. Durch Normierung der dritten Koordinate auf 1, erhalten wir die Repräsentanten $(x_0 : y_0 : 1)$, zusammen mit $(x_0 : y_0 : 0)$ (wobei $(x_0, y_0) \neq (0, 0)$) den unendlich fernen Punkten. Man kann dann die projektiven Punkte $(x_0 : y_0 : 1)$ mit der affinen Ebene $\mathbb{R}^2$ identifizieren.

Eine ebene (projektive) algebraische Kurve C_F über $\mathbb{K}$ ist dann die Nullstellenmenge eines Polynoms $F \in \mathbb{K}[X, Y, Z]$, für welches $F(\lambda X, \lambda Y, \lambda Z) = \lambda^n F(X, Y, Z), \lambda \in \mathbb{K}$, gilt, falls n der Gesamtgrad von F ist. Nun sieht man leicht, daß der Begriff Nullstelle wohldefiniert auf den Punkten von $\mathbb{P}^2(\mathbb{K})$ ist, d.h.

$$C_F = \left\{ (x_0 : y_0 : z_0) \in \mathbb{P}^2(\mathbb{K}) \,\middle|\, F(x_0, y_0, z_0) = 0 \right\}$$

Für unsere Zwecke ist es ausreichend, die Nullstellen eines einzigen solchen Polynoms zu betrachten:

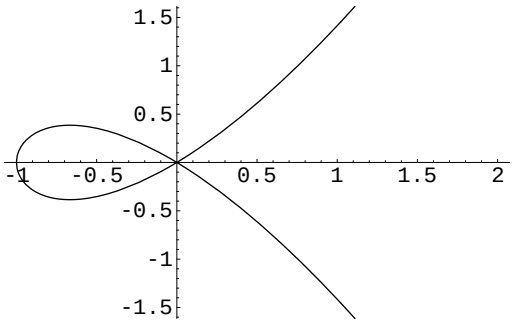
$$F(X,Y,Z) := Y^2Z + a_1XYZ + a_3YZ^2 - X^3 - a_2X^2Z - a_4XZ^2 - a_6Z^3$$

Wir bezeichnen

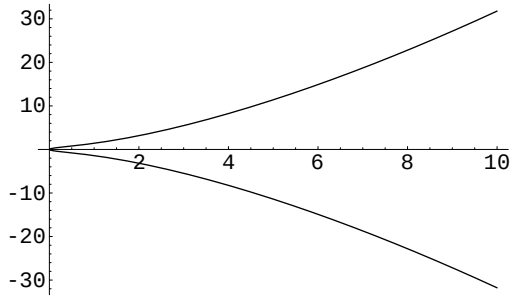
$$Y^2Z + a_1XYZ + a_3YZ^2 \quad = \quad X^3 + a_2X^2Z - a_4XZ^2 + a_6Z^3$$

auch als Weierstraß Gleichung, und sprechen in diesem Zusammenhang von den Lösungen der Weierstraß Gleichung.

Die Weierstraß Gleichung heißt nichtsingulär, wenn für alle projektiven Punkte $P = (x_0 : y_0 : z_0) \in \mathbb{P}^2(\mathbb{K})$, die $F(x_0, y_0, z_0) = 0$ erfüllen, mindestens eine der (formalen) partiellen Ableitungen $\frac{\partial F}{\partial X}$, $\frac{\partial F}{\partial Y}$, $\frac{\partial F}{\partial Z}$ in P von Null verschieden ist.



Singuläre elliptische Kurve $y^2 = x^3 + x^2$



Neillsche Parabel $y^2 = x^3$ über $\mathbb{R}$

Definition 2.2. Eine elliptische Kurve E über einem Körper $\mathbb{K}$ ist die Menge aller Lösungen einer nichtsingulären Weierstraß-Gleichung in $\mathbb{P}^2(\mathbb{K})$. Es existiert genau ein Punkt auf E, bei dem die z-Koordinate gleich Null ist, nämlich $\mathcal{O} := (0 : 1 : 0)$. $\mathcal{O}$ heißt der unendlich ferne Punkt.

Sei nun

$$E(\mathbb{K}) \quad := \quad \left\{ (x_0 : y_0 : 1) \in \mathbb{P}^2(\mathbb{K}) \,\middle|\, F(x_0, y_0, 1) = 0 \right\} \cup \{(0 : 1 : 0)\}$$

Wenn wir nun die Punkte $(x_0, y_0) \in \mathbb{K}^2$ mit den Punkten $(x_0 : y_0 : 1) \in \mathbb{P}^2(\mathbb{K})$ identifizieren, können wir die elliptische Kurve $E(\mathbb{K})$ als Nullstellenmenge von

$$f(x,y) := F(x,y,1) = y^2 + a_1xy + a_3y - x^3 - a_2x^2 - a_4x - a_6 \in \mathbb{K}[x,y]$$

in $\mathbb{K}^2$, zusammen mit dem unendlich fernen Punkt $\mathcal{O}$, auffassen.

Diese neue Auffassung der elliptischen Kurve erinnert uns stark an Definition 1. Der einzige Unterschied besteht darin, daß die affine Weierstraß Gleichung

$$y^2 + a_1xy + a_3y \quad = \quad x^3 + a_2x^2 + a_4x + a_6 \tag{2.1}$$

noch etwas komplizierter aussieht, also die Gleichung 1.1 aus dem vorigen Kapitel. Dies liegt daran, daß die Gleichung 2.1 für Körper mit beliebiger Charakteristik gilt. Wir können aber Gleichung 2.1 mittels Variablentransformation in Gleichung 1.1 überführen.

Diese Transformation vereinfacht die elliptische Kurve E aus Gleichung 2.1 zu einer isomorphen Kurve E' mit der Gleichung

$$\begin{array}{rcll} y^2 & = & x^3 + ax + b & \text{char}(\mathbb{K}) \neq 2,3 \\ y^2 & = & x^3 + ax^2 + bx + c & \text{char}(\mathbb{K}) = 3 \\ y^2 + xy & = & x^3 + ax + c & \text{char}(\mathbb{K}) = 2,\ a_1 \neq 0 \\ y^2 + cy & = & x^3 + ax + b & \text{char}(\mathbb{K}) = 2,\ a_1 = 0 \end{array}$$

mit a, b und $c \in \mathbb{K}$.

2.1. **Addition in $E(\mathbb{K})$.** Die Vorschrift für die Addition in $E(\mathbb{K})$ bleibt im Falle $\mathrm{char}(\mathbb{K}) \neq 2,3$ die aus Definition 1.2. Die etwas verallgemeinerte Formulierung der Addition, die sich auf die Weierstraß Gleichung 2.1 bezieht, setzt $-P := (x_1, -y_1 - a_1x_1 - a_3)$, falls $P = (x_1, y_1) \neq \mathcal{O}$. Sonst entspricht sie der Definition 1.2, dabei wird die Tangente an die Kurve $f(x,y) = 0$ im Punkt $P = (a,b)$ als die Gerade $\frac{\partial f}{\partial x}(a,b) \cdot (x-a) + \frac{\partial f}{\partial y}(a,b) \cdot (y-b) = 0$ definiert.

Da der Fall eines Körpers der Charakteristik 2 für die Kryptologie besonders interessant ist, geben wir hier die expliziten Formeln für diesen Fall an:

Fall $a_1 \neq 0$ in 2.1: Sei $P = (x_1, y_1) \in E(\mathbb{K})$. Dann ist

$$-P = (x_1, y_1 + x_1).$$

Für $Q = (x_2, y_2) \in E(\mathbb{K})$ und $Q \neq -P$ gilt $P + Q = (x_3, y_3)$ mit

$$x_3 = \begin{cases} \left(\frac{y_1+y_2}{x_1+x_2}\right)^2 + \frac{y_1+y_2}{x_1+x_2} + x_1 + x_2 + a & \text{falls } P \neq Q \\ x_1^2 + \frac{c}{x_1^2} & \text{falls } P = Q \end{cases}$$

$$y_3 = \begin{cases} \frac{y_1+y_2}{x_1+x_2} \cdot (x_1 + x_3) + x_3 + y_1 & \text{falls } P \neq Q \\ x_1^2 + \left(x_1 + \frac{y_1}{x_1}\right) x_3 + x_3 & \text{falls } P = Q \end{cases}$$

Fall $a_1 = 0$ in 2.1: Sei $P = (x_1, y_1) \in E(\mathbb{K})$. Dann ist

$$-P = (x_1, y_1 + c).$$

Für $Q = (x_2, y_2) \in E(\mathbb{K})$ und $Q \neq -P$ gilt $P + Q = (x_3, y_3)$ mit

$$x_3 = \begin{cases} \left(\frac{y_1+y_2}{x_1+x_2}\right)^2 + \frac{y_1+y_2}{x_1+x_2} + x_1 + x_2 & \text{falls } P \neq Q \\ \frac{x_1^4+a^2}{c^2} & \text{falls } P = Q \end{cases}$$

$$y_3 = \begin{cases} \frac{y_1+y_2}{x_1+x_2} \cdot (x_1 + x_3) + y_1 + c & \text{falls } P \neq Q \\ \frac{x_1^2+a}{c} \cdot (x_1 + x_3) + y_1 + c & \text{falls } P = Q \end{cases}$$

2.2. **Eigenschaften von $E(\mathbb{F}_q)$.** Sei E eine elliptische Kurve über $\mathbb{F}_q$ mit $q = p^m$, wobei p (eine Primzahl) die Charakteristik von $\mathbb{F}_q$ ist. Wir bezeichnen die Zahl der Punkte auf $E(\mathbb{F}_q)$ mit $\#E(\mathbb{F}_q)$. Wenn wir die Weierstraß Gleichung 2.1 betrachten, so sehen wir, daß für jedes feste $x \in \mathbb{F}_q$ die Gleichung höchstens zwei Lösungen hat (es bleibt ein quadratischen Polynom übrig), also muß die Abschätzung $\#E(\mathbb{F}_q) \leq 2q + 1$ gelten (q Möglichkeiten x zu wählen und $\mathcal{O}$). Heuristisch gesehen würde man erwarten, daß bei nur der Hälfte der betrachteten x eine Lösung in $\mathbb{F}_q$ existiert, also folgt daraus $\#E(\mathbb{F}_q) \approx q$. Der nachfolgende Satz bestätigt diese Vermutung.

Theorem 2.3. *(Hasse) Es sei $\#E(\mathbb{F}_q) = q + 1 - t$. Dann ist $|t| \leq 2\sqrt{q}$.*

Wir sehen also, daß ein zufällig gewähltes Element $x_0 \in \mathbb{F}_q$ mit der Wahrscheinlichkeit

$$\frac{\frac{1}{2}(\#E(\mathbb{F}_q) - 1)}{\|\mathbb{F}_q\|} \geq \frac{q - 2\sqrt{q}}{2q} = \frac{1}{2} - \frac{1}{\sqrt{q}}$$

die x-Koordinate eines Punktes von $E(\mathbb{F}_q)$ darstellt. Ist x_0 die x-Koordinate eines Punktes $P \in E(\mathbb{F}_q)$, dann können wir die zugehörige y-Koordinate durch Wurzelziehen in $\mathbb{F}_q$ finden, dazu existieren probabilistische Algorithmen. Wir bekommen automatisch einen zweiten Punkt, nämlich $-P$. Insgesamt haben wir einen probabilistischen Algorithmus zum Auffinden von Punkten auf $E(\mathbb{F}_q)$, der in polynomieller Zeit arbeitet.

Auch von Interesse ist oft die tatsächliche Anzahl der Punkte auf einer elliptischen Kurve $E(\mathbb{F}_q)$. 1985 hat Schoof einen polynomiellen Algorithmus für die Berechnung von $\#E(\mathbb{F}_q)$ angegeben. Eine Modifikation dieses Algorithmus haben Buchmann und Muller 1991 vorgestellt, um Ordnungen elliptischer Kurven über $\mathbb{F}_p$ zu berechnen. Eine Implementierung dieser Methode hat z.B. Ordnungen von Kurven über $\mathbb{F}_p$, wo $p \approx 10^{27}$ prim in ca. 4.5 Stunden auf einer SUN-1 SPARC-station berechnet.

3. Elliptische Kurven über $\mathbb{Z}_n$

Der Begriff einer elliptischen Kurve über einem Ring $\mathbb{Z}_n$ erweist sich als sehr nützlich. Doch auf den ersten Blick mag es verwunderlich erscheinen, elliptische Kurven über einem Ring zu betrachten, wo wir doch spätestens bei der Addition von Punkten sehr stark auf die Körperstruktur angewiesen waren. Deswegen sehen wir von der Gruppenstruktur ab, und betrachten eine elliptische Kurve über dem Ring $\mathbb{Z}_n$ erstmal als eine Menge von Punkten, die einer bestimmten Gleichung genügen.

Definition 3.1. Sei n eine positive ganze Zahl mit $\mathrm{ggT}(n, 6) = 1$. Eine elliptische Kurve über $\mathbb{Z}_n$ ist gegeben durch die Gleichung

$$E_{a,b}: \quad y^2 = x^3 + ax + b, \tag{3.1}$$

wobei $a, b \in \mathbb{Z}$ und $\mathrm{ggT}\left(4a^2 + 27b^2, n\right) = 1$. Die Menge der Punkte auf $E_{a,b}$ wird als $E_{a,b}(\mathbb{Z}_n)$ bezeichnet. Es sind die Lösungen der Gleichung 3.1 in $\mathbb{Z}_n \times \mathbb{Z}_n$ zusammen mit dem unendlich fernen Punkt $\mathcal{O}_n$.

Es sei nun p ein Primteiler von n, und bezeichne $\bar{a}$ den Rest von a modulo p. $E_{\bar{a},\bar{b}}$ ist dann die Gleichung einer elliptischen Kurve über dem Körper $\mathbb{F}_p$. Zu einem Punkt $P = (x, y) \in E_{a,b}(\mathbb{Z}_n)$ definieren wir $P_p := (\bar{x}, \bar{y})$. Zu $\mathcal{O}_n$ definieren wir $(\mathcal{O}_n)_p$ als den unendlich fernen Punkt $\mathcal{O}_p$ in $E_{\bar{a},\bar{b}}(\mathbb{F}_p)$. Offenbar gilt $P_p \in E_{\bar{a},\bar{b}}(\mathbb{F}_p)$.

Wie am Anfang des Abschnittes angemerkt, wird es uns kaum gelingen eine Gruppenstruktur auf $E_{a,b}(\mathbb{Z}_n)$ zu bekommen. Trotz allem definieren wir eine Pseudo-Addition wie in der Definition 1.2. Wir werden versuchen Formel 1.2 anzuwenden, solange wir keinen undefinierten Ausdruck erhalten, d.h. solange der Fall $\mathrm{ggT}(x_2 - x_1, n) \neq 1$ nicht eintritt. In diesem Fall ist unsere Pseudo-Addition nicht definiert.

Die folgenden Eigenschaften der Pseudo-Addition können nachgewiesen werden:

(1) Für $P, Q \in E_{a,b}(\mathbb{Z}_n)$ und $P + Q$ nicht definiert, ergibt sich während der Berechnung ein nicht trivialer Teiler von n.
(2) Ist $P, Q \in E_{a,b}(\mathbb{Z}_n)$ und $P + Q$ definiert durch die Pseudo-Addition, dann gilt
$$(P + Q)_p = P_p + Q_p$$
für alle Primteiler p von n. (Beachte, daß die rechte Seite der Gleichung immer definiert ist.)
(3) Insbesondere ist $P \in E_{a,b}(\mathbb{Z}_n)$, $k \in \mathbb{Z}$ und $k \cdot P := \underbrace{P + \ldots + P}_{k \text{ mal}}$ definiert durch wiederholte Anwendung der Pseudo-Addition, dann gilt
$$(kP)_p = kP_p$$
für alle Primteiler p von n.

Angenommen n ist das Produkt von zwei Primzahlen p, q. Setze

$$\tilde{E}_{a,b}(\mathbb{Z}_n) := E_{a,b}(\mathbb{F}_p) \times E_{a,b}(\mathbb{F}_q)$$

Im Gegensatz zu $E_{a,b}(\mathbb{Z}_n)$ ist $\tilde{E}_{a,b}(\mathbb{Z}_n)$ eine Gruppe, da sie direktes Produkt von zwei Gruppen ist. Jeder Punkt P auf $E_{a,b}(\mathbb{Z}_n)$ korrespondiert zu einem eindeutigen Punkt auf $\tilde{E}_{a,b}(\mathbb{Z}_n)$, nämlich zu (P_p, P_q). Punkte auf $\tilde{E}_{a,b}(\mathbb{Z}_n)$ von der Form $(\mathcal{O}_p, Q)$ und $(P, \mathcal{O}_q)$ haben keine korrespondierenden Punkte auf $E_{a,b}(\mathbb{Z}_n)$. Wegen der obigen Eigenschaft (2) sehen wir, falls die Pseudo-Addition in $E_{a,b}(\mathbb{Z}_n)$ definiert ist, so ist diese identisch mit der Addition in $\tilde{E}_{a,b}(\mathbb{Z}_n)$, d.h. für $P, Q \in E_{a,b}(\mathbb{Z}_n)$ und $P + Q$ definiert, gilt:

$$\left(\left(P \underbrace{+}_{in\, E_{a,b}(\mathbb{Z}_n)} Q\right)_p, (P + Q)_q\right) = (P_p, P_q) \underbrace{+}_{in\, \tilde{E}_{a,b}(\mathbb{Z}_n)} (Q_p, Q_q)$$

Das hat weitreichende Konsequenzen, wir können nämlich in der Gruppe $\tilde{E}_{a,b}(\mathbb{Z}_n)$ rechnen, ohne die Primfaktoren p und q tatsächlich zu kennen. Sollte die Anwendung der Gruppenoperationen fehlschlagen, so erhalten wir einen nicht trivalen Faktor von n. Sind p und q in der Größenordnung von 100 Dezimalstellen, so kann das Fehlschlagen der Pseudo-Addition als sehr unwahrscheinlich angenommen werden, da das Faktorisieren von solch großen Zahlen als sehr hart angesehen wird.

Teil 2. Public-Key Kryptosysteme über elliptischen Kurven

Viele Public-Key Kryptosysteme lassen sich mit Hilfe einer Gruppe implementieren, in der das diskrete Logarithmus Problem als sehr schwierig angesehen werden kann. Die elliptischen Kurven geben uns ein großes Arsenal an endlichen Gruppen in die Hand, in denen der diskrete Logarithmus schwierig zu berechnen ist. Die gängigen Verfahren wie ElGamal, Diffie-Hellman Schlüsseltausch usw. bieten über den elliptischen Kurven gleich hohe Sicherheit, wie über endlichen Körpern bei gleichzeitig kürzeren Schlüssellängen. Dies kann entscheidend für z.B. die Anwendung auf Chipkarten sein, wo der Speichplatz knapp bemessen ist.

Die Erfahrungen zeigen, daß ein komplettes Kryptosystem für die Verwendung elliptischer Kurven weniger als 4% der Chipfläche ausmachen würde.

Da viele elliptische Kurven über dem gleichen Körper implementiert werden können, kann jeder Benutzer eine andere elliptische Kurve wählen (die Körperoperationen können z.B. mit dem gleichen spezialisierten Chip realisiert werden, folglich braucht man nur eine Hardware für alle). Aus Sicherheitsgründen kann die elliptische Kurve periodisch geändert werden.

Es gibt auch andere Kryptoverfahren, die speziell die Struktur von elliptischen Kurven ausnützen, darunter das Public-Key Verfahren mit elliptischen Kurven über dem Ring $\mathbb{Z}_n$.

In diesem Kapitel sei G eine Gruppe der Ordnung n ($|G| = n$) und $h \in G$ ein Element der Primordnung r ($h^r = 1$, r prim). Dann ist r ein Primteiler von n und $\langle h\rangle = \{1, h, h+h, h+h+h, \ldots\}$ eine zyklische Untergruppe von G. Da r eine Primzahl ist, haben alle Elemente aus $\alpha \in \langle h\rangle$ außer das neutrale Element die Ordnung r. Da bedeutet, daß $\langle h\rangle$ außer $\langle 1\rangle$ keine Untergruppe besitzt.

4. Das diskrete Logarithmus-Problem

Da elliptische Kurven additiv beschrieben werden, kann das diskrete Logarithmus Problem wie folgt definiert werden:

Definition 4.1. Sei $(G, +)$ eine additiv geschriebene Gruppe, H eine von einem Element erzeugte Untergruppe von G, d.h. $H = \langle h\rangle$. Für ein beliebiges Element $a \in G$ besteht das diskrete Logarithmus Problem (**DLP**) darin, zu entscheiden, ob $a \in H$, und falls ja, ein $m \in \mathbb{Z}$ zu berechnen, so daß $m \cdot h := \underbrace{h + \ldots + h}_{m \text{ mal}} = a$ ist. Dieses m wird als $\log_h a$, der Logarithmus von a, bezeichnet.

Wie beim Faktorisieren stellt sich bei der Berechnung von diskreten Logarithmen heraus, daß i.a. keine effizienten, d.h. in Polynomialzeit durchführbaren Algorithmen bekannt sind. Während aber für die Faktorisierung einer beliebigen Zahl grundsätzlich subexponentielle Algorithmen zur Verfügung stehen, sind subexponentielle Algorithmen zur Berechnung diskreter Logarithmen i.a. nicht bekannt. Es gibt aber Familien von Gruppen, für die aber doch subexponentielle Algorithmen bekannt sind; die multiplikativen Gruppen der endlichen Körper $\mathbb{F}_q$ gehören zu dieser Familie.

Die Familie der Gruppen, die durch bestimmte elliptische Kurven über endlichen Körpern gebildet werden, gehören vermutlich nicht dazu; diese bestimmten elliptischen Kurven zeichnen sich dadurch aus, gewisse Eigenschaften nicht zu haben.

5. Schlüssel-Austausch-Verfahren

Die Verfahren, die in diesem Abschnitt vorgestellt werden, werden dazu verwendet, um zwischen zwei oder mehreren Parteien über nicht gesichterte Kanäle einen gemeinsamen, geheimen Wert zu berechnen; aus diesem geheimen Wert kann z.B. mittels einer Hash-Funktion ein gemeinsamer Sitzungs-Schlüssel (Session Key) erzeugt werden. Wie betrachten hier ausschließlich symmetrische Zweiweg-Protokolle.

Das simpelste Verfahren ist zugleich auch das älteste Public-Key-Verfahren, das öffentlich bekannt war:

5.1. **Diffie-Hellman.**

Algorithmus 5.1. *(Diffie-Hellman)*

(1) *Erzeugung des Schlüsselpaares*
  A *wählt eine Zufallszahl* s_A, $1 < s_A < r$ *und berechnet* $\sigma_A \leftarrow s_A \cdot h$;
  B *wählt eine Zufallszahl* s_B, $1 < s_B < r$ *und berechnet* $\sigma_B \leftarrow s_B \cdot h$.

(2) *Austausch*
  A *sendet* σ_A *an* **B**;
  B *sendet* σ_B *an* **A**.

(3) *Berechnen des gemeinsamen Wertes*
  A *berechnet* $\kappa_A \leftarrow s_A \cdot \sigma_B$
  B *berechnet* $\kappa_B \leftarrow s_B \cdot \sigma_A$

$\kappa = \kappa_\mathrm{A} = \kappa_B$ *ist der gemeinsame, geheime Wert.*

Protokoll 5.1 basiert auf dem DLP: s_A und s_B sind die geheimen Schlüssel, σ_A und σ_B sind die öffentlichen Schlüssel. κ ist der geimeinsame Wert, denn offensichtlich ist

$$\kappa_A = s_A \cdot \sigma_B = s_A \cdot (s_B \cdot h) = s_B \cdot (s_A \cdot h) = s_B \cdot \sigma_A = \kappa_B$$

Außerdem ist κ nur von **A** und **B** bekannt, denn andernfalls müßte s_A und s_B rekonstruiert werden, indem $s_A = \log_h \sigma_A$ und $s_B = \log_h \sigma_B$ berechnet werden. Die Berechnung von κ aus σ_A, σ_B und h wird auch als Diffie-Hellman-Problem bezeichnet.

Diese Konstruktion ist für praktische Anwendungen etwas problematisch, denn es ist nicht ganz klar, wann Schritt 1 ausgeführt wird, d.h. wann die Schlüsselpaare erzeugt werden. Wenn die Schlüsselpaare im voraus erzeugt werden, dann werden **A** und **B** bei der Durchführung von 5.1 immer denselben gemeinsamen Wert berechnen; wenn die Schlüsselpaare jedesmal neu erzeugt werden, tauschen **A** und **B** unzertifizierte Schlüssel aus.

Das Problem wird durch die Verwendung von zwei Schlüsselpaaren (für jede beteiligte Partei) gelöst, einem statischen (langlebigen) Schlüsselpaar und einem flüchtigen (kurzlebigen) Schlüsselpaar; das statische Schlüsselpaar wird dabei grundsätzlich von einer Zertifizierungs-Einrichtung (CA) zertifiziert, das flüchtige Schlüsselpaar wird für jeden Protokoll-Durchgang neu erzeugt und nur ein einziges mal verwendet. Im folgenden wird hier das statische und das flüchtige Schlüsselpaar mit (s, σ) und (t, τ) bezeichnet, wobei $s, t \in \mathbb{Z}_r$ die geheimen und $\sigma, \tau \in H$ die öffentlichen Schlüssel sind.

5.2. **MTI.** Wir stellen hier nun einige Erweiterungen von 5.1 vor, in denen statische und flüchtige Schlüsselpaare zum Einsatz kommen. Wir stellen zunächst zwei Vertreter aus den MTI-Familien vor (benannt nach T. Matsumoto, Y. Takashima, H. Imai):

Algorithmus 5.2. *(MTI/A0)*

(1) *Erzeugung des statischen Schlüsselpaares*
  A *wählt eine Zufallszahl* s_A, $1 < s_A < r$ *und berechnet* $\sigma_A \leftarrow s_A \cdot h$;
  B *wählt eine Zufallszahl* s_B, $1 < s_B < r$ *und berechnet* $\sigma_B \leftarrow s_B \cdot h$.
(2) *Erzeugung des flüchtigen Schlüsselpaares*
  A *wählt eine Zufallszahl* t_A, $1 < t_A < r$ *und berechnet* $\tau_A \leftarrow t_A \cdot h$;
  B *wählt eine Zufallszahl* t_B, $1 < t_B < r$ *und berechnet* $\tau_B \leftarrow t_B \cdot h$;
(3) *Austausch*
  A *sendet* τ_A *an* ***B***;
  B *sendet* τ_B *an* ***A***.
(4) *Berechnen des gemeinsamen Wertes*
  A *besorgt* ***B****s authentischen öffentlichen Schlüssel* σ_B;
  B *besorgt* ***A****s authentischen öffentlichen Schlüssel* σ_A;
  A *berechnet* $\kappa \leftarrow s_A \cdot \tau_B + t_A \cdot \sigma_B = (s_A t_B + s_B t_A) \cdot h$;
  B *berechnet* $\kappa \leftarrow s_B \cdot \tau_A + t_B \cdot \sigma_A = (s_A t_B + s_B t_A) \cdot h$.

Algorithmus 5.3. *(MTI/C0)*

(1) *Erzeugung des statischen Schlüsselpaares*
  A *wählt eine Zufallszahl* s_A, $1 < s_A < r$ *und berechnet* $\sigma_A \leftarrow s_A \cdot h$;
  B *wählt eine Zufallszahl* s_B, $1 < s_B < r$ *und berechnet* $\sigma_B \leftarrow s_B \cdot h$.
(2) *Erzeugung des flüchtigen Schlüsselpaares*
  A *besorgt* ***B****s authentischen öffentlichen Schlüssel* σ_B;
  B *besorgt* ***A****s authentischen öffentlichen Schlüssel* σ_A;
  A *wählt eine Zufallszahl* t_A, $1 < t_A < r$ *und berechnet* $\tau_A \leftarrow t_A \cdot \sigma_B$;
  B *wählt eine Zufallszahl* t_B, $1 < t_B < r$ *und berechnet* $\tau_B \leftarrow t_B \cdot \sigma_A$;
(3) *Austausch*
  A *sendet* τ_A *an* ***B***;
  B *sendet* τ_B *an* ***A***.
(4) *Berechnen des gemeinsamen Wertes*
  A *berechnet* $\kappa \leftarrow s_A^{-1} t_A \cdot \tau_B = (t_A t_B) \cdot h$;
  B *berechnet* $\kappa \leftarrow s_B^{-1} t_B \cdot \tau_A = (t_A t_B) \cdot h$.

Man kann sich leicht davon überzeugen daß die Protokolle 5.2 und 5.3 vom DLP in H abhängt. Die Protokolle der MTI-Familie weisen eine Reihe von Schwächen auf; zur Illustration demonstrieren wir einen Angriff auf MTI/C0, wenn r keine Primzahl ist.

Angenommen, es sei $r = pu$, p eine Primzahl und $u > 1$ ein Teiler von r.

(1) **E** fängt **A**s Nachricht τ_A ab und ersetzt sie durch $p \cdot \tau_A$;
(2) **E** fängt **B**s Nachricht τ_B ab und ersetzt sie durch $p \cdot \tau_B$;
(3) **A** berechnet $\kappa \leftarrow s_A^{-1} t_A \cdot (p \cdot \tau_B)$;
(4) **B** berechnet $\kappa \leftarrow s_B^{-1} t_B \cdot (p \cdot \tau_A)$.

κ ist nun in der Untergruppe mit u Elementen enthalten, die von $p \cdot h$ erzeugt wird; wenn u klein genug ist, dann kann κ effizient z.B. durch erschöpfende Suche gefunden werden. Dieser Angriff wird als „Angriff mit kleinen Untergruppen" (small subgroup attack) bezeichnet. Es gibt jedoch auch Angriffe gegen MTI-Protokolle, die nicht einfach nur durch gute Wahl von h verhindert werden können.

6. Verschlüsselungs-Verfahren

6.1. Ein RSA-Äquivalent für elliptischen Kurven über $\mathbb{Z}_n$.

Algorithmus 6.1. *(KMOV)*

(1) *Schlüsselpaar-Erzeugung:*
A *führt die folgenden Schritte durch:*
- *wähle zwei Primzahlen p und q mit $p \equiv 2 \mod 3$ und $q \equiv 2 \mod 3$*
- *berechne $n \leftarrow pq$*
- *wähle eine zufällige Zahl e, so daß $ggT(e, (p+1)(q+1)) = 1$*
- *berechne ein d, so daß $ed \equiv 1 \mod (p+1)(q+1)$.*

A*s öffentlicher Schlüssel ist (n, e).* ***A****s geheimer Schlüssel ist d.*

(2) *Verschlüsselung:*
B *führt die folgenden Schritte durch, um eine Nachricht $M = (x, y) \in \mathbb{Z}_n \times \mathbb{Z}_n$ an* ***A*** *zu übertragen:*
- *besorge* ***A****s authentischen, öffentlichen Schlüssel (n, e)*
- *berechne $(c_1, c_2) \leftarrow e \cdot (x, y)$ in der Gruppe $\tilde{E}_{0,b}(\mathbb{Z}_n) = E_{0,b}(\mathbb{F}_p) \times E_{0,b}(\mathbb{F}_q)$, wobei $b = y^2 - x^3 \mod n$.*

$C = (c_1, c_2)$ ist die Verschlüsselung von M.

(3) *Entschlüsselung:*
A *führt den folgenden Schritt durch:*
- *berechne $M = (x, y) \leftarrow d \cdot (c_1, c_2)$.*

Beachte hierbei, daß **B** im 2. Schritt die Faktoren p und q nicht kennt, aber trotzdem in $\tilde{E}_{0,b}(\mathbb{Z}_n)$ rechnen kann, indem er die Pseudo-Addition in $E_{0,b}(\mathbb{Z}_n)$ anwendet (vgl. Kapitel 3). Die Wahrscheinlichkeit, daß das Rechnen in $E_{0,b}(\mathbb{Z}_n)$ fehlschlägt ist sehr gering.

Um die Korrektheit des Verfahrens zu verifizieren, stellen wir zuerst fest:

Theorem 6.2. *Es sei q eine Potenz einer ungeraden Primzahl, so daß $q \equiv 2 \mod 3$ gilt, und sei weiter $b \in \mathbb{F}_q, b \neq 0$. Betrachte die elliptische Kurve $E : y^2 = x^3 + b$ über $\mathbb{F}_q$. Dann gilt*

$$\#E(\mathbb{F}_q) = q + 1.$$

Als Folgerung des Satzes erhalten wir

$$\#E_{0,b}(\mathbb{F}_p) = p + 1 \qquad \#E_{0,b}(\mathbb{F}_q) = q + 1$$

und damit

$$\#\tilde{E}_{0,b}(\mathbb{Z}_n) = (p+1)(q+1)$$

Wegen $de \equiv 1 \mod \#\tilde{E}_{0,b}(\mathbb{Z}_n)$ erhalten wir

$$d \cdot (c_1, c_2) = de \cdot (x, y) \equiv (x, y) \mod \#\tilde{E}_{0,b}(\mathbb{Z}_n).$$

Dieses Verfahren hat die Eigenschaft, daß die Kurve, mit der wir rechnen, von der zu verschlüsselnden Nachricht abhängt. Dieses Kryptosystem ist allerdings nicht so effizient wie das RSA-Verfahren, dafür soll es resistenter gegen bestimmte bekannte RSA-Attacken sein.

Es sind aber auch folgende Attacke bekannt:

Theorem 6.3. *(Angriff bei teilweise bekanntem Klartext)*

Sei (n, e) ein öffentlicher Schlüssel für KMOV und $C = (c_1, c_2)$ die verschlüsselte Nachricht $M = (x, y)$. Ist nun n, e, C und x oder y gegeben, so kann der Klartext M effizient berechnet werden.

Der Beweis dieses Satzes benutzt die Redundanz, die durch $b \equiv c_2^2 - c_1^3 \mod n$ gegeben ist.

Wenn dieselbe Nachricht mehrmals mit verschiedenen Public-Keys verschlüsselt wird, ist folgender Satz anwendbar:

Theorem 6.4. *Sei $t \geq 1$ und seien (e_1, n_1), (e_2, n_2), (e_3, n_3) verschiedene öffentliche Schlüssel. Sei $M = (x, y) \in \{0, \min_i \{n_i\} - 1\}^2$ eine unbekannte Nachricht. Wenn 3 Cryptotexte existieren, die mit den 3 Schlüsseln verschlüsselt wurden, dann kann M in Zeit $O\left(t^2 \log(n)^3\right)$ mit Wahrscheinlichkeit $1 - \frac{1}{t}$ gefunden werden, wobei $n = \max_i \{n_i\}$.*

6.2. **ElGamal.** Das bekannteste Public-Key-Verfahren, welches auf dem DLP basiert, ist das Verschlüsselungsverfahren nach T. ElGamal:

Algorithmus 6.5. *(ElGamal-Verschlüsselung)*

(1) *Schlüsselpaar-Erzeugung*
   ***A** führt die folgenden Schritte durch:*
   - *wähle eine Gruppe G der Ordnung n und ein Element $h \in G$ mit Primordnung r;*
   - *wähle eine Zufallszahl a mit $1 < a < r$;*
   - *berechne $\alpha \leftarrow a \cdot h$*

   ***A**s öffentlicher Schlüssel ist (G, h, α), **A**s geheimer Schlüssel ist a.*

(2) *Verschlüsselung*
   ***B** führt die folgenden Schritte durch:*
   - *besorge **A**s authentischen öffentlichen Schlüssel;*
   - *kodiere die Nachricht M als Element μ von G;*
   - *wähle eine Zufallszahl k mit $1 < k < r$;*
   - *berechne $\rho \leftarrow k \cdot h$;*
   - *berechne $\beta \leftarrow \mu + k \cdot \alpha$.*

   $C = (\rho, \beta)$ ist die Verschlüsselung von M.

(3) *Entschlüsselung*
   ***A** führt die folgenden Schritte durch:*
   - *berechne mit dem geheimen Schlüssel a die Nachricht $\mu = \beta - a \cdot \rho$;*
   - *stelle M aus μ wieder her.*

Die ElGamal-Verschlüsselung basiert tatsächlich auf dem DLP, denn um μ aus ρ und β zu rekonstruieren, muß $a = \log_h \alpha$ bekannt sein.

Ein Nachteil der ElGamal-Verschlüsselung besteht in der „Ciphertext-Expansion", der Kryptotext $C = (\rho, \beta)$ ist nämlich doppelt so lang, wie der Klartext μ. Da Public-Key-Verfahren üblicherweise nur auf kleinen Mengen von Klartext angewendet werden (z.B. auf einen Session-Key für symmetrische Verschlüsselung), ist dies meistens nicht kritisch. Andererseits ist die Text-Expansion die Folge einer anderen, vorteilhaften Eigenschaft: Da bei jeder Verschlüsselung für k eine (andere) Zufallszahl gewählt wird, werden bei verschiedenen Verschlüsselungen mit ansonsten gleichen Parametern gleiche Klartexte auf unterschiedliche Kryptotexte abgebildet. Diese Eigenschaft erschwert den „Wörterbuch-Angriff", bei dem große Mengen an Klartext und korrespondierendem Kryptotext vorausberechnet werden. Mithin ist es kritisch, daß für jede Verschlüsselung ein anderes k gewählt wird. Um das zu sehen, beachte man, daß $\beta_1 - \beta_2 = \mu_1 - \mu_2$ ist, wenn k konstant gewählt wird; wenn einem Angreifer bereits z.B. μ_1 bekannt ist, kann er daraus μ_2 berechnen.

6.3. **Einbettung von Text.** Bei Verschlüsselungen und auch bei Signaturen mit Message-Recovery müssem wir den Text M bei den bisher gezeigten Verfahren als Gruppenelement $\mu \in G$ darstellen; dies wird als Einbettung von M in G bezeichnet. Die Einbettung soll effizient sein, und außerdem wollen wir M aus μ eindeutig (und ebenfalls effizient) wiederherstellen können. Um M in G einbetten zu können, muß M i.a. zunächst in Blöcke „passender" Größe unterteilt werden, denn bei festem G können höchstens $|G|$ unterschiedliche Texte M eingebettet werden. Wir gehen im weiteren davon aus, daß M binärcodiert auf offensichtliche Weise als natürliche Zahl interpretiert werden kann, und daß M bereits eine „passende" Größe hat. Wenn z.B. $G = \mathbb{Z}_p^*$ ist, dann ergibt sich eine natürliche Möglichkeit, M auf μ abzubilden, denn hier ist μ identisch mit M aufgefaßt als positive ganze Zahl.

Bei anderen Gruppen kann die Einbettung komplizierter werden. Wenn wir einen Text als Punkt einer elliptischen Kurve E über $\mathbb{F}_q$ kodieren wollen, dann müssen wir eine Repräsentation von M als Paar (x, y) finden. Auf den ersten Blick hat man keine andere Wahl, als M z.B. auf die x-Koordinate eines Punktes abzubilden, weil die x- und die y-Koordinaten eines Punktes nicht unabhängig voneinander sind. Gibt man nämlich die x-Koordinate vor, so kann man maximal nur noch das Vorzeichen der y-Koordinate auswählen.

An dieser Stelle wird aber noch eine ganz andere Schwierigkeit offenkundig, denn nicht zu jeder x-Koordinate existiert ein Punkt auf der Kurve. In diesem Fall müßten wir die Möglichkeit haben, M ersatzweise auf eine andere x-Koordinate abzubilden, und zwar so, daß M aus x eindeutig rekonstruiert werden kann. Wenn M als Element von $\mathbb{F}_q$ interpretiert wird, z.B. durch $M = \sum_{i=0}^{m-1} a_i p^i$ mit $a_i \in \mathbb{F}_p$ und $q = p^m$, dann kann für die Einbettung das folgende Schema angewendet werden:

$$M \mapsto mk + i$$

für das kleinste $i, 0 \leq i \leq k$, so daß $mk + i$ die x-Koordinate eines Punktes aus $E(\mathbb{F}_q) \setminus \{\mathcal{O}\}$ ist. Da $x \in \mathbb{F}_q$ ist und die Abbildung in gewisser Weise eindeutig sein soll, muß $M < \frac{q}{k}$ sein, und daher können höchstens etwas $\log_2\left(\frac{q}{k}\right) = \log_2 q - \log_2 k$ Bits auf diese Weise eingebettet werden. Mit anderen Worten, es müssen für jede Einbettung $\log k$ Bits aufgewendet werden, um die Wahrscheinlichkeit eines Fehlschlags entsprechend zu verkleinern, d.h. mit dieser Methode kann immer noch nicht garantiert werden, daß es für M eine Darstellung als Punkt der Kurve gibt.

In Abschnitt 2.2 hatten wir dazu bereits festgestellt, daß man bei zufälliger Wahl einer x-Koordinate mit einer Wahrscheinlichkeit von etwa $\frac{1}{2}$ einen Punkt auf der Kurve erhält, d.h. man erhält mit einer Wahrscheinlichkeit von etwas $\frac{1}{2}$ keinen Punkt. Wenn wir für ein M also k Versuche für die Einbettung haben, dann kann M mit einer Wahrscheinlichkeit von 2^{-k} nicht auf die x-Koordinate eines Punktes aus $E(\mathbb{F}_q) \setminus \{\mathcal{O}\}$ abgebildet werden. Dies geht natürlich mit einem Verlust einher, denn je größer K ist, desto weniger Bits können für M selber verwendet werden, d.h. die Einbettung wird ineffizienter. Wenn wir z.B. $k = 256$ wählen, dann müssen wir dafür 8 Bits (für jedes M) aufwenden, allerdings ist die Wahrscheinlichkeit eines Fehlschlages mit 2^{-256} verschwindend gering.

Bei anderen Gruppen kann das Einbettungs-Problem weitaus komplizierter sein. Um dieses Problem bei der Verschlüsselung zu vermeiden, kann man die folgende Variante der ElGamal-Verschlüsselung anwenden:

Algorithmus 6.6.

(1) *Schlüsselpaar-Erzeugung*

A *führt die folgenden Schritte durch:*

- *wähle eine elliptische Kurve E über $\mathbb{F}_q$ der Ordnung n und ein Element $h \in E(\mathbb{F}_q)$ mit Primordnung r;*
- *wähle eine Zufallszahl a mit $1 < a < r$;*
- *berechne $\alpha \leftarrow a \cdot h$*

A*s öffentlicher Schlüssel ist $(E(\mathbb{F}_q), h, \alpha)$,* ***A****s geheimer Schlüssel ist a.*

(2) *Verschlüsselung*

B *führt die folgenden Schritte durch:*

- *besorge* ***A****s authentischen öffentlichen Schlüssel;*
- *wähle eine Zufallszahl k mit $1 < k < r$;*
- *berechne $\rho \leftarrow k \cdot h$;*
- *berechne $\beta \leftarrow k \cdot \alpha$;*
- *für eine Hash-Funktion $i : E(\mathbb{F}_q) \to \mathbb{F}_q$ berechne $c \leftarrow i(\beta) \oplus M$*

$C = (\rho, c)$ ist die Verschlüsselung von M.

(3) *Entschlüsselung*

A *führt die folgenden Schritte durch:*

- *berechne $\beta \leftarrow a \cdot \rho$*
- *berechne $M \leftarrow i(\beta) \oplus c$*

Der Trick ist, daß die Multiplikation von β mit einer Einbettung von M in $E(\mathbb{F}_q)$ durch ein simples bitweises XOR von $i(\beta)$ und M ersetzt wird.

Die Vorteile liegen auf der Hand: M wird direkt und deterministisch in die Verschlüsselung eingebracht, ohne die u. U. probabilistische, mühselige Suche nach einem Gruppenelement, in welches M ansonsten eingebettet werden muß.

6.4. **MOV.** Menez und Vanstone schlugen ein auf ElGamal basierendes Kryptosystem für Nachrichten $M = (x, y) \in \mathbb{F}_q \times \mathbb{F}_q$ vor, bei dem auch das Problem der Kodierung einer Nachricht als Element von $E(\mathbb{F}_q)$ nicht auftritt.

Algorithmus 6.7. *(MOV - Menezes, Okamoto, Vanstone)*

(1) *Schlüsselpaar-Erzeugung*

***A** führt die folgenden Schritte durch:*

- *wähle eine elliptische Kurve E über $\mathbb{F}_q$ der Ordnung n und ein Element $h \in E(\mathbb{F}_q)$ mit Primordnung r;*
- *wähle eine Zufallszahl a mit $1 < a < r$;*
- *berechne $\alpha \leftarrow a \cdot h$*

***A**s öffentlicher Schlüssel ist $(E(\mathbb{F}_q), h, \alpha)$, **A**s geheimer Schlüssel ist a.*

(2) *Verschlüsselung*

***B** führt die folgenden Schritte durch:*

- *besorge **A**s authentischen öffentlichen Schlüssel;*
- *wähle eine Zufallszahl k mit $1 < k < r$;*
- *berechne $\rho \leftarrow k \cdot h$;*
- *berechne $c_1 \leftarrow x(k \cdot \alpha)_1$ und $c_2 \leftarrow y(k \cdot \alpha)_2$, wobei γ_i die i-te Koordinate von $\gamma \in E(\mathbb{F}_q)$ sein soll $(i = 1, 2)$;*

$C = (\rho, c_1, c_2)$ *ist die Verschlüsselung von M.*

(3) *Entschlüsselung*

***A** führt die folgenden Schritte durch:*

- *berechne $\beta \leftarrow a \cdot \rho$*
- *berechne $x \leftarrow c_1 \frac{1}{\beta_1} = x(k \cdot \alpha)_1 \frac{1}{(a \cdot \rho)_1} = x(ka \cdot h)_1 \frac{1}{(ka \cdot h)_1} = x \in \mathbb{F}_q$*
- *berechne $y \leftarrow c_2 \frac{1}{\beta_2} = y(k \cdot \alpha)_2 \frac{1}{(a \cdot \rho)_2} = y(ka \cdot h)_2 \frac{1}{(ka \cdot h)_2} = y \in \mathbb{F}_q$*

Wenn ein Angreifer x (oder y) kennt, ist es ihm einfach möglich, y (oder x) zu berechnen. Um die Sicherheit zu erhöhen, kann man auch nur $(\rho, c_1) \in E(\mathbb{F}_q) \times \mathbb{F}_q \subseteq \mathbb{F}_q^3$ als eine Verschlüsselung von x senden. Damit wäre der Kryptotext dann aber dreimal so lang wie der Klartext.

Wir können die „Cyphertext-Expansion" verringern, indem wir die y-Koordinate von $\rho \in E(\mathbb{F}_q)$ mit nur einem Bit kodieren, da es zu einer festen x-Koordinate von ρ ja nur maximal zwei y-Koordinaten geben kann, für die $\rho \in E(\mathbb{F}_q)$ gilt.

Damit vergrößern sich die Nachrichten nur noch um den Faktor $\approx \frac{3}{2}$.

Angenommen, das DLP ist für $q \sim 2^{160}$ nicht mehr effizient lösbar, dann erhalten wir bei gleicher „Sicherheit" von RSA, ElGamal über $\mathbb{F}_q^*$ und MOV durch Komprimierung der y-Koordinate von ρ und durch Senden von nur (ρ, c_1):

System	Kryptotextgröße in Bits
RSA	1024
ElGamal	2048
MOV	321

Vergleich der Kryptotextgrößen einer 100 Bit großen Nachricht

7. Signatur-Verfahren

Für eine Gruppe G der Ordnung n sei $g : G \to \mathbb{Z}_n$ eine einfache (d.h. effizient berechenbare) bijektive Funktion, die ein $\alpha \in G$ als eindeutiges Element von $\mathbb{Z}_n$ repräsentiert.

7.1. ElGamal.

Algorithmus 7.1. *(ElGamal-Signatur)*

(1) *Schlüsselpaar-Erzeugung*
   ***A** führt die folgenden Schritte durch:*
   - *wähle eine Gruppe G der Ordnung n und ein Element $h \in G$ mit Primordnung r;*
   - *wähle eine Zufallszahl a mit $1 < a < r$*
   - *berechne $\alpha \leftarrow a \cdot h$*

   ***A**s öffentlicher Schlüssel ist (G, h, α). **A**s geheimer Schlüssel ist a.*

(2) *Signatur*
   ***A** führt die folgenden Schritte durch:*
   - *wähle eine Zufallszahl k mit $1 < k < r$;*
   - *berechne $\rho \leftarrow k \cdot h$*
   - *berechne den Hash-Wert $i(M)$ der Nachricht M;*
   - *berechne $s \leftarrow k^{-1}(i(M) - ag(\rho)) \mod n$.*

   ***A**s Signatur ist (ρ, s).*

(3) *Verifikation*
   ***B** führt die folgenden Schritte durch:*
   - *besorge **A**s authentischen, öffentlichen Schlüssel;*
   - *berechne $v_1 \leftarrow g(\rho) \cdot \alpha + s \cdot \rho$;*
   - *berechne $v_2 \leftarrow i(M) \cdot h$;*

   ***B** akzeptiert die Signatur S genau dann, wenn $v_1 = v_2$ ist.*

Die Verifikation funktioniert folgendermaßen:

$$\begin{aligned} v_1 &= g(\rho) \cdot \alpha + s \cdot \rho \\ &= ag(\rho) \cdot h + k^{-1}(i(M) - ag(\rho)) \cdot k \cdot h \\ &= i(M) \cdot h \\ &= v_2 \end{aligned}$$

Um eine Signatur zu fälschen, muß $a = \log_h \alpha$ bekannt sein, alternativ muß $k = \log_h \rho$ aus einer Signatur rekonstruiert werden, um a zu berechnen:

$$\begin{aligned} s &\equiv k^{-1}(i(M) - ag(\rho)) \mod n \\ \to a &\equiv \frac{i(M) - ks}{g(\rho)} \mod n \end{aligned}$$

Die ElGamal-Signatur basiert daher auch auf dem DLP. Für die ElGamal-Signatur gelten dieselben Hinweise, wie für die ElGamal-Verschlüsselung: Die Signatur ist etwas doppelt so lang, wie der Text, jedoch gilt auch hier, daß üblicherweise nur kleine Mengen an Text (nämlich der Hash-Wert der Nachricht M) signiert werden, so daß dies in den meisten Fällen kein Problem ist.

Die Verwendung einer Hash-Funktion ist aber ohnehin aus Gründen der Sicherheit zwingend erforderlich: Wird bei dem Signatur-Verfahren keine Hash-Funktion auf die Nachricht M angewendet, dann wird in Schritt (2) s durch $k^{-1}(M - ag(\rho)) \mod n$ berechnet. In diesem Fall kann ein Angreifer **E** eine beliebige Signatur ohne Kenntnis des geheimen Schlüssels a erzeugen, indem **E** folgende Schritte (anstelle des 2. Schrittes) ausführt:

- wähle zwei Zufallszahlen u und v, so daß $0 < u, v < n$ und v kein Teiler von n ist;
- berechne $\rho \leftarrow u \cdot h + v \cdot \alpha = (u + av) \cdot h$;
- berechne $s \leftarrow -g(\rho) v^{-1} \mod n$

Das Paar $S = (\rho, s)$ ist eine gültige Signatur für $M \equiv su \mod n$, denn die Verifikation ergibt

$$\begin{aligned} v_1 &= g(\rho) \cdot \alpha + s \cdot \rho \\ &= ag(\rho) \cdot h + (s(u + av)) \cdot h \\ &= (ag(\rho) + su + sav) \cdot h \\ &= (ag(\rho) + su - ag(\rho)) \cdot h \\ &= (su) \cdot h \\ &= M \cdot h \\ &= v_2 \end{aligned}$$

Da u und v mehr oder weniger beliebig ausgewählt werden können, kann praktisch für jede beliebige Nachricht eine gültige Signatur erzeugt werden. Dies wird verhindert, indem anstelle von M der Hash-Wert $i(M)$ signiert wird; hier ist es offensichtlich von Bedeutung, daß das Urbild $i(M)$ nicht effizient berechnet werden kann.

Außerdem muß für jede Signatur eine neue Zufallszahl k gewählt werden, andernfalls kann ein Angreifer mit hoher Wahrscheinlichkeit k und den geheimen Schlüssel a aufdecken: Seien $S_1 = (\rho_1, s_1)$ und $S_2 = (\rho_2, s_2)$ die Signaturen von M_1 und M_2. Dann ist $\rho_1 = \rho_2 = \rho = k \cdot h$ und

$$\begin{aligned} s_1 &\equiv k^{-1}(i(M_1) - ag(\rho)) \mod n, \\ s_2 &\equiv k^{-1}(i(M_2) - ag(\rho)) \mod n. \end{aligned}$$

Es folgt $(s_1 - s_2)\, k \equiv i(M_1) - i(M_2) \mod n$, und wenn $(s_1 - s_2) \mod n$ invertierbar ist, dann ist

$$k \equiv \frac{i(M_1) - i(M_2)}{s_1 - s_2} \mod n.$$

Wenn $g(\rho) \mod n$ invertierbar ist, dann kann daraus $a \equiv \frac{i(M) - ks}{g(\rho)} \mod n$ für $M = M_1$ oder $M = M_2$ berechnet werden.

Die Wahl eines unterschiedlichen k für jede Signatur führt nebenbei auch dazu, daß bei ansonsten gleichen Parametern ein Text unterschiedliche Signaturen erzeugen kann.

7.2. **DSA.** Der größte Teil der Berechnungen beim ElGamal-Signaturverfahren wird für die Exponentiationen (d.h. skalare Multiplikationen) verwendet. Die drei Exponentiationen bei der Verifikation lassen sich durch die Methode der „simultanen Exponentiation" durch einige Verbesserungen zu einer einzigen Exponentiation zusammenfassen. Eine weitere Möglichkeit besteht darin, das Hamming-Gewicht der Exponenten (d.h. der skalaren Faktoren) zu begrenzen; genau dies wird bei DSA (Digital Signature Algorithm) gemacht:

Algorithmus 7.2. *(verallgemeinerte DSA-Signatur)*

(1) *Schlüsselpaar-Erzeugung*

***A** führt die folgenden Schritte durch:*

- *wähle eine Primzahl q, so daß $2^{159} < q < 2^{160}$ ist;*
- *wähle eine Gruppe G der Ordnung n, so daß $q|n$ gilt; bestimme ein Element $h \in G$ mit Primordnung q*
- *wähle eine Zufallszahl a mit $1 < a < q$;*
- *berechne $\alpha \leftarrow a \cdot h$*

***A**s öffentlicher Schlüssel ist (G, h, α), **A**s geheimer Schlüssel ist a.*

(2) *Signatur*

***A** führt die folgenden Schritte durch:*

- *wähle eine Zufallszahl k mit $1 < k < q$;*
- *berechne $\rho \leftarrow k \cdot h$*
- *berechne den Hash-Wert $i(M)$ der Nachricht M;*
- *berechne $s \leftarrow k^{-1}(i(M) + ag(\rho)) \mod q$*

***A**s Signatur von M ist (ρ, s).*

(3) *Verifikation*

***B** führt die folgenden Schritte durch:*

- *besorge **A**s authentischen, öffentlichen Schlüssel*
- *berechne $w \leftarrow s^{-1} \mod q$ und $i(M)$;*
- *berechne $v \leftarrow (i(M)\, w) \cdot h + (g(\rho)\, w) \cdot \alpha$;*

***B** akzeptiert die Signatur genau dann, wenn $v = \rho$ ist.*

Die Verifikation funktioniert folgendermaßen:

$$\begin{aligned} v &= (i(M)\, w) \cdot h + (g(\rho)\, w) \cdot \alpha \\ &= \left(i(M)\, s^{-1}\right) \cdot h + \left(g(\rho)\, s^{-1} a\right) \cdot h \\ &= s^{-1} (i(M) + a g(\rho)) \cdot h \\ &= k (i(M) + a g(\rho))^{-1} (i(M) + a g(\rho)) \cdot h \\ &= k \cdot h \\ &= \rho \end{aligned}$$

DSA ist mit $G = \mathbb{Z}_p^*$ Teil des DSS (Digital Signature Standard), und da DSA ein enger Verwandter der ElGamal-Signatur ist, gelten die Sicherheitshinweise für ElGamal auch für DSA.

Angenommen, das DLP ist für $q \sim 2^{160}$ nicht mehr effizient lösbar, dann können wir einen großen Vorteil des DSA über elliptischen Kurven (EC DSA) gegenüber RSA und allgemeinem DSA feststellen, weil der Schlüssel und die Signaturgrößen bei gleicher Sicherheit viel kleiner sind:

System	Systemparameter	öffentlicher Schlüssel	privater Schlüssel	Signaturgröße
RSA	-	1088	2048	1024
DSA	2208	1024	160	320
EC DSA	481	161	160	320

Vergleich der Schlüssel- und Signaturgrößen einer 2000 Bit langen Nachricht

7.3. **Chaum-van Antwerpen.** In diesem Abschnitt zeigen wir, daß auch ungewöhnliche, DLP-basierte Signatur-Verfahren auf beliebigen Gruppen und damit auf elliptischen Kurven übertragen werden können. Das folgende Signatur-Verfahren erzeugt „unabweisbare" Signaturen (Undeniable Signatures). Bei diesem Signatur-Verfahren erfordert die Verifikation die Unterstützung des Signierers:

Algorithmus 7.3. *(Chaum-van Antwerpen)*

(1) *Schlüsselpaar-Erzeugung:*

***A** führt folgende Schritte durch:*

- *wähle eine Gruppe G der Ordnung n und ein Element $h \in G$ mit Primordnung r;*
- *wähle eine Zufallszahl a mit $1 < a < r$;*
- *berechne $\alpha \leftarrow a \cdot h$.*

***A**s öffentlicher Schlüssel ist (G, h, α), **A**s geheimer Schlüssel ist a.*

(2) *Signatur*

***A** führt die folgenden Schritte durch:*

- *Bilde M (oder einen Hash-Wert von M) auf $\mu \in H = \langle h \rangle$ ab;*
- *berechne $\rho \leftarrow a \cdot \mu$;*

***A**s Signatur von M ist ρ.*

(3) *Verifikation*

***B** führt die folgenden Schritte durch:*

- *besorge **A**s authentischen, öffentlichen Schlüssel;*
- *wähle zwei Zufallszahlen $e_1, e_2 \in \mathbb{Z}_r^*$;*
- *berechne $\beta \leftarrow e_1 \cdot \rho + e_2 \cdot \alpha$;*

***A** führt die folgenden Schritte durch:*

- *berechne* $v \leftarrow \left(a^{-1} \mod r\right) \cdot \beta$;

***B** akzeptiert die Signatur genau dann, wenn* $e_1 \cdot \mu + e_2 \cdot h = v$ *ist.*

Die Verifikation folgt aus

$$\begin{aligned} v &= \left(a^{-1} \mod r\right) \cdot \beta \\ &= \left(a^{-1} \mod r\right) \cdot (e_1 \cdot \rho + e_2 \cdot \alpha) \\ &= a^{-1} e_1 a \cdot \mu + a^{-1} e_2 a \cdot h \\ &= e_1 \cdot \mu + e_2 \cdot h \end{aligned}$$

Die Kooperation von **A** ist zwingend erforderlich, ohne **A**s Mithilfe kann **B** die Signature nicht verifizieren. **A** könnte seine Kooperation nach belieben verweigern, legitim wäre dies aber nur, wenn eine gefälschte Signatur zur Überprüfung vorläge. **A** kann aber gefälschte Signaturen als solche durch das Disavowal-Protokoll als solche nachweisen:

Algorithmus 7.4. *Sei* $\rho \neq a \cdot \mu$

(1) *Disavowal*

***B** führt die folgenden Schritte durch:*
- *wähle zwei Zufallszahlen* $e_1, e_2 \in \mathbb{Z}_r^*$;
- *berechne* $\beta_e \leftarrow e_1 \cdot \rho + e_2 \cdot \alpha$;

***A** führt die folgenden Schritte durch:*
- *berechne* $v_e \leftarrow \left(a^{-1} \mod r\right) \beta_e$;

***B** führt die folgenden Schritte durch:*
- *prüfe, ob* $v_e \neq e_1 \cdot \mu + e_2 \cdot h$ *ist;*
- *wähle zwei Zufallszahlen* $f_1, f_2 \in \mathbb{Z}_r^*$;
- *berechne* $\beta_f \leftarrow f_1 \cdot \rho + f_2 \cdot \alpha$;

***A** führt die folgenden Schritte durch:*
- *berechne* $v_f \leftarrow \left(a^{-1} \mod r\right) \beta_f$;

***B** führt die folgenden Schritte durch:*
- *prüfe, ob* $v_f \neq f_1 \cdot \mu + f_2 \cdot h$ *ist;*
- *prüfe schließlich, ob* $f_1 \cdot (v_e - e_2 \cdot h) = e_1 \cdot (v_f - f_2 \cdot h)$ *ist.*

Der Schlüssel zu diesem Protokoll steckt in dem aller letzten Schritt. Damit überprüft **B** nämlich, ob **A**s Berechnungen dem Protokoll entsprechen, denn es gilt

$$\begin{aligned} f_1 \cdot (v_e - e_2 \cdot h) &= f_1 \cdot \left(a^{-1}\beta_e - e_2 \cdot h\right) \\ &= f_1 \cdot \left(a^{-1} \cdot (e_1 \cdot \rho + e_2 \cdot \alpha) - e_2 \cdot h\right) \\ &= f_1 a^{-1} e_1 \cdot \rho \\ e_1 \cdot (v_f - f_2 \cdot h) &= e_1 \cdot \left(a^{-1}\beta_f - f_2 \cdot h\right) \\ &= e_1 \cdot \left(a^{-1} \cdot (f_1 \cdot \rho + f_2 \cdot \alpha) - f_2 \cdot h\right) \\ &= e_1 a^{-1} f_1 \cdot \rho \end{aligned}$$

Literatur

[1] Artur Merke: Elliptische Kurven in der Kryptographie - Eine Einführung (Seminarvortrag), Uni Karlsruhe (TH), SS 1997
http://i11www.ira.uka.de/~amerke/el_kurv.ps.gz

[2] Safuat Hamdy: Anwendungen elliptischer Kurven in der Kryptologie (Diplomarbeit)
http://www.informatik.tu-darmstadt.de/TI/Mitarbeiter/hamdy/misc/aekk.ps

[3] Rainer Wilmink: Elliptic Curve Cryptosystems (Diplomarbeit), August 1999
http://www.mathematik.uni-bielefeld.de/infoboerse/examensarbeiten/rainer/rainer.html

[4] Linkliste: http://www.rzuser.uni-heidelberg.de/~hb3/ellc.html