Fakulteten för teknik- och naturvetenskap

Johan Jonsson

Elliptiska kurvor och Lenstras faktoriseringsalgoritm

Elliptic Curves and Lenstra's Factorization Algorithm

Matematik
Examensarbete 15 hp, magisternivå

Datum:	2009-06-05
Handledare:	Håkan Granath
Examinator:	Alexander Bobylev

Karlstads universitet 651 88 Karlstad
Tfn 054-700 10 00 Fax 054-700 14 60
Information@kau.se www.kau.se

Sammanfattning

En elliptisk kurva består av nollställena till ett kubisk polynom i två variabler, sådant att det existerar åtminstone en punkt på kurvan och kurvan är icke-singulär. Punkterna på en sådan kurva bildar en abelsk grupp och olika egenskaper hos dessa grupper beskrivs i den här uppsatsen. Bland annat presenteras Mordell-Weils sats som säger att en elliptisk kurva över en talkropp är en ändligt genererad grupp. Nagell-Lutz sats ger nödvändiga villkor för att en punkt på en rationell elliptisk kurva ska ha ändlig ordning. Resultatet att en elliptisk kurva över de komplexa talen är isomorf med en torus presenteras också. Tillämpningen heltalsfaktorisering presenteras genom en beskrivning av Lenstras algoritm. En implementation av denna algoritm i form av ett datorprogram görs och denna implementation jämförs med den triviala algoritmen för heltalsfaktorisering.

Abstract

An elliptic curve consists of the zeros of a cubic polynomial i two variables, such that there exists at least one point on the curve and the curve is non-singular. The points on such a curve form an abelian group and various properties of these groups are described in this thesis. Among other things the Mordell-Weil theorem, which states that an elliptic curve over a number field is a finitely generated group, is presented. The Nagell-Lutz theorem gives necessary conditions for a point on a rational elliptic curve to have finite order. Another presented result is that an elliptic curve over the complex numbers is a torus. The application of elliptic curves in integer factorization is presented by describing Lenstra's algorithm. A computer program implementing this algorithm is made and this implementation is compared to the trivial algorithm for integer factorization.

Innehåll

Inledning

Trots vad namnet ”elliptisk kurva” kanske antyder så är en elliptisk kurva *inte* samma sak som en ellips. En elliptiska kurva är istället nollställena till en speciell form av kubiskt polynom i två variabler. Namnet kommer från det faktum att dessa kurvor har en koppling till beräkningen av en ellips omkrets.

En intressant aspekt med ämnet elliptiska kurvor är att det finns kopplingar till så många av matematikens olika grenar och områden. Komplex analys, topologi, geometri och abstrakt algebra är bara några exempel på områden som finns representerade på olika sätt.

Några tillämpningar av elliptiska kurvor är i algoritmer för kryptering, heltalsfaktorisering och bevis av att ett heltal är ett primtal. Tillämpningen heltalsfaktorisering kommer att undersökas närmare i ett avsnitt av den här uppsatsen.

Uppsatsen är till största delen baserad på materialet i boken ”Rational Points on Elliptic Curves” av Silverman och Tate (se [1] i referenslistan). De flesta av de givna exemplen och många av de bevis som ges i uppsatsen består av lösningar till övningsuppgifter i den använda litteraturen. Vissa bevis av satser och lemman har gjorts för att förtydliga detaljer som har saknats. För satser som är bevisade i den använda litteraturen så har bevisen i regel inte återgetts utan hänvisningar till respektive källa har istället gjorts.

Jag vill också passa på att rikta ett stort tack till min handledare Håkan Granath för att ha introducerat ämnet elliptiska kurvor och handlett mig genom denna uppsats.

1 Kubiska kurvor

För enkelhetens skull så kommer alla kroppar i den här framställningen, om inget annat anges, antas ha karakteristik skild från två och tre så "kropp" kommer att betyda "kropp med karakteristik skild från två och tre".

För att studera kubiska kurvor underlätter det att använda projektiv geometri och därför kommer en kort introduktion av detta nu att ges. Det *projektiva planet* över en kropp $\mathbb{F}$ kommer att betecknas med $\mathbb{P}^2(\mathbb{F})$ och detta består av ekvivalensklasser av punkter $(X, Y, Z) \in \mathbb{F} \times \mathbb{F} \times \mathbb{F}$, $(X, Y, Z) \neq (0_{\mathbb{F}}, 0_{\mathbb{F}}, 0_{\mathbb{F}})$. En punkt $(X, Y, Z) \in \mathbb{F} \times \mathbb{F} \times \mathbb{F}$ är *ekvivalent* med en annan punkt $(X', Y', Z') \in \mathbb{F} \times \mathbb{F} \times \mathbb{F}$ om det finns ett $t \in \mathbb{F}$, $t \neq 0_{\mathbb{F}}$, så att $X = tX'$, $Y = tY'$ och $Z = tZ'$. Detta kommer att skrivas $(X, Y, Z) \sim (X', Y', Z')$ och $\sim$ är en ekvivalensrelation på mängden $\mathbb{F} \times \mathbb{F} \times \mathbb{F} \backslash \{(0, 0, 0)\}$. Det projektiva planet $\mathbb{P}^2(\mathbb{F})$ definieras alltså som mängden av alla dessa ekvivalensklasser. Ekvivalensklassen som innehåller (X, Y, Z) kommer att skrivas $[X, Y, Z]$. Det "vanliga" affina planet över en kropp $\mathbb{F}$ kommer att betecknas med $\mathbb{A}^2(\mathbb{F})$.

Nollställena till ett homogent polynom, av grad tre, på formen

$$\begin{aligned} &F(X, Y, Z) = \\ &= aX^3 + bX^2Y + cXY^2 + dY^3 + eX^2Z + fXYZ + gY^2Z + hXZ^2 + iYZ^2 + jZ^3 \end{aligned} \quad (1)$$

över någon kropp $\mathbb{F}$, kallas för en *projektiv kubisk kurva* eller enbart en *kubisk kurva*. Om kurvan betecknas med C så kommer mängden av alla nollställen att betecknas med

$$C(\mathbb{F}) = \{[X, Y, Z] \in \mathbb{P}^2(\mathbb{F}) : F(X, Y, Z) = 0\}$$

Nollställena till ett polynom, av grad tre, på formen

$$f(x, y) = ax^3 + bx^2y + cxy^2 + dy^3 + ex^2 + fxy + gy^2 + hx + iy + j \quad (2)$$

över någon kropp $\mathbb{F}$, kallas för en *affin kubisk kurva*.

Ett polynom $f(x, y) \in \mathbb{F}[x, y]$ av grad d kan *homogeniseras* till ett homogent polynom $F(X, Y, Z) \in \mathbb{F}[X, Y, Z]$ genom att sätta $F(X, Y, Z) = Z^d f(X/Z, Y/Z)$. Ett homogent polynom $F(X, Y, Z) \in \mathbb{F}[X, Y, Z]$ *dehomogeniseras*, med avseende på Z, till ett polynom $f(x, y) \in \mathbb{F}[x, y]$ genom att sätta $f(x, y) = F(x, y, 1)$. Dehomogenisering med avseende på X och Y definieras analogt.

Homogenisering av (2) ger (1) och dehomogenisering av (1) med avseende på Z ger (2) så de två kurvorna är relaterade till varandra. Skillnaden är att den projektiva varianten har punkter i oändligheten, det vill säga punkter $[X, Y, Z]$ där $Z = 0_{\mathbb{F}}$. I fortsättningen kommer för enkelhetens skull ofta en affin kubisk kurva att anges även om det är motsvarande projektiva kubiska kurva som avses. En kurva över $\mathbb{Q}$ kommer att kallas för en *rationell* kurva.

En punkt $P = (x_0, y_0) \in \mathbb{F} \times \mathbb{F}$, på en affin kurva $C : f(x, y) = 0$ över kroppen $\mathbb{F}$, kallas för en *singulär punkt* på C om

$$\frac{\partial f}{\partial x}(x_0, y_0) = \frac{\partial f}{\partial y}(x_0, y_0) = 0$$

En punkt $P = [X_0, Y_0, Z_0] \in \mathbb{P}^2(\mathbb{F})$, på en projektiv kurva $C : F(X, Y, Z) = 0$ över kroppen $\mathbb{F}$, kallas för en *singulär punkt* på C om

$$\frac{\partial F}{\partial X}(X_0, Y_0, Z_0) = \frac{\partial F}{\partial Y}(X_0, Y_0, Z_0) = \frac{\partial F}{\partial Z}(X_0, Y_0, Z_0) = 0$$

En kurva kallas för en *icke-singulär kurva* om den saknar singulära punkter.

Definition 1.1. *Låt C vara en icke-singulär kubisk kurva över någon kropp $\mathbb{F}$ och låt $P, Q \in \mathbb{P}^2(\mathbb{F})$ vara två punkter på C. Då definieras $P * Q$ som den tredje skärningspunkten mellan C och linjen genom P och Q.*

Notera följande. För att kompositionsregeln $*$ ska bli väldefinierad så måste "linjen genom" tolkas på rätt sätt. Om $P = Q$ så måste linjen genom P och Q tolkas som tangenten till C i punkten P. Om linjen genom P och Q är tangent till C i P så måste den tredje skärningspunkten tolkas som P eftersom skärningspunkten P då har multiplicitet två. En *inflektionspunkt* på en kurva definieras som en icke-singulär punkt på kurvan där skärningspunkten mellan tangenten i punkten och kurvan har multiplicitet tre. Om $P = Q$ är en inflektionspunkt så måste $P * P$ tolkas som P.

Lemma 1.1. *Kompositionsregeln $*$ är kommutativ, dvs $P * Q = Q * P$ för alla $P, Q \in C(\mathbb{F})$, och $P * (P * Q) = Q$ för alla $P, Q \in C(\mathbb{F})$.*

Bevis. Eftersom linjen genom P och Q är samma linje som linjen genom Q och P så är $*$ kommutativ. $P * (P * Q) = Q$ eftersom P, Q, $P * Q$ och $P * (P * Q)$ tillhör samma linje. □

Däremot är $*$ inte associativ enligt följande exempel så punkterna på C tillsammans med $*$ bildar ingen grupp.

Exempel 1.1. Betrakta den icke-singulära rationella kubiska kurvan

$$C : y^2 = x^3 + 17$$

Den har de rationella punkterna $P = (-2, 3)$, $Q = (-1, 4)$ och $R = (2, 5)$. För dessa punkter så gäller att $Q * R = (-\frac{8}{9}, \frac{109}{27})$ och $P * Q = (4, 9)$. Vidare gäller att $P * (Q * R) = (\frac{94}{25}, \frac{1047}{125})$ och $(P * Q) * R = (-2, -3)$ så det är tydligt att $*$ inte är associativ.

Det är möjligt att definiera en gruppstruktur på mängden $C(\mathbb{F})$ med hjälp av kompositionsregeln i följande definition.

Definition 1.2. *Låt C vara en icke-singulär kubisk kurva över någon kropp $\mathbb{F}$ med en given punkt $\mathcal{O} \in C(\mathbb{F})$ och låt $P, Q \in C(\mathbb{F})$ vara två punkter på C. Då definieras $P + Q$ som*

$$P + Q = \mathcal{O} * (P * Q)$$

Som valet av +-tecken i föregående definition antyder så kommer additiv notation att användas för denna gruppstruktur.

Lemma 1.2. *Låt C vara en icke-singulär kubisk kurva över någon kropp $\mathbb{F}$ med en given punkt $\mathcal{O} \in C(\mathbb{F})$ och låt $+$ vara definierad med hjälp av $\mathcal{O}$ enligt definition 1.2. Då är $+$ kommutativ och $\mathcal{O}$ är neutralt element med avseende på $+$. Om $P \in C(\mathbb{F})$ så är $-P = P * (\mathcal{O} * \mathcal{O})$ invers till P med avseende på $+$.*

Bevis. Eftersom $*$ är kommutativ så gäller för alla $P, Q \in C(\mathbb{F})$ att

$$P + Q = \mathcal{O} * (P * Q) = \mathcal{O} * (Q * P) = Q + P$$

så $+$ är också kommutativ. $\mathcal{O}$ är neutralt element med avseende på $+$ eftersom, enligt lemma 1.1

$$\mathcal{O} + P = P + \mathcal{O} = \mathcal{O} * (P * \mathcal{O}) = \mathcal{O} * (\mathcal{O} * P) = P$$

för alla $P \in C(\mathbb{F})$. Låt $P \in C(\mathbb{F})$ och definiera $-P = P * (\mathcal{O} * \mathcal{O})$. Då är

$$P + (-P) = P + (P * (\mathcal{O} * \mathcal{O})) = \mathcal{O} * (P * (P * (\mathcal{O} * \mathcal{O}))) = \mathcal{O} * (\mathcal{O} * \mathcal{O}) = \mathcal{O}$$

så $-P$ är invers till P med avseende på $+$. □

Lemma 1.3. *Låt C vara en icke-singulär kubisk kurva över någon kropp $\mathbb{F}$ med en given neutral punkt $\mathcal{O} \in C(\mathbb{F})$. Då är $+$ associativ om och endast om*

$$R * (\mathcal{O} * (P * Q)) = P * (\mathcal{O} * (Q * R))$$

för alla $P, Q, R \in C(\mathbb{F})$.

Bevis.

$$\begin{gathered} P + (Q + R) = (P + Q) + R \iff \\ P + (\mathcal{O} * (Q * R)) = R + (\mathcal{O} * (P * Q)) \iff \\ \mathcal{O} * (P * (\mathcal{O} * (Q * R))) = \mathcal{O} * (R * (\mathcal{O} * (P * Q))) \iff \\ P * (\mathcal{O} * (Q * R)) = R * (\mathcal{O} * (P * Q)) \end{gathered}$$

□

Det som återstår för att visa att $(C(\mathbb{F}), +)$ bildar en grupp är alltså att $R * (\mathcal{O} * (P * Q)) = P * (\mathcal{O} * (Q * R))$ för alla $P, Q, R \in C(\mathbb{F})$. Detta kommer inte att presenteras här utan lämnas till nästa avsnitt där Weierstrass normalform introduceras vilket underlätter beviset.

2 Elliptiska kurvor

Nu kommer begreppet elliptisk kurva att definieras. Innan definitionen kan göras så måste Weierstrass normalform introduceras.

Definition 2.1. *En kubisk kurva, över någon kropp $\mathbb{F}$, på formen*

$$y^2 = x^3 + ax^2 + bx + c$$

sägs vara på Weierstrass normalform.

Varje icke-singulär kubisk kurva C över en någon kropp $\mathbb{F}$, som har åtminstone en punkt tillhörande $C(\mathbb{F})$, är birationellt ekvivalent med en kubisk kurva på Weierstrass normalform (Silverman [1], sida 22). Det kan alltså vara intressant att studera kubiska kurvor på denna ofta enklare form.

Definition 2.2. *En icke-singulär kubisk kurva C över någon kropp $\mathbb{F}$, som har åtminstone en punkt tillhörande $C(\mathbb{F})$, kallas för en* elliptisk kurva.

Det kommer senare att visas att varje icke-singulär kubisk kurva på Weierstrass normalform har åtminstone en punkt på kurvan så varje icke-singulär kubisk kurva på Weierstrass normalform är en elliptisk kurva.

Att avgöra om en kubisk kurva på Weierstrass normalform är icke-singulär eller inte är generellt sett inte svårt genom att undersöka diskriminanten enligt följande sats.

Sats 2.1. *En kubisk kurva,*

$$y^2 = f(x) = x^3 + ax^2 + bx + c$$

över någon kropp $\mathbb{F}$ är icke-singulär om och endast om diskriminanten till f(x),

$$D = -4a^3c + a^2b^2 + 18abc - 4b^3 - 27c^2$$

är nollskild.

Bevis. Se Silverman [2], sida 50. □

Sats 2.2. *En elliptisk kurva,*

$$C : f(x, y) = y^2 - x^3 - ax^2 - bx - c = 0 \tag{3}$$

har exakt en punkt i oändligheten och denna punkt är en inflektionspunkt.

Bevis. Homogenisering av kurvans ekvation (3) ger

$$F(X, Y, Z) = Y^2Z - X^3 - aX^2Z - bXZ^2 - cZ^3 = 0 \tag{4}$$

Dehomogenisering av (4) med avseende på Y ger

$$g(x, z) = F(x, 1, z) = z - x^3 - ax^2z - bxz^2 - cz^3$$

Partiell derivering av g ger

$$\frac{\partial g}{\partial x} = -3x^2 - 2axz - bz^2,$$
$$\frac{\partial g}{\partial z} = 1 - ax^2 - 2bxz - 3cz^2$$

Betrakta punkten $[0, 1, 0]$ på C. Eftersom

$$\frac{\partial g}{\partial z}\Big(\frac{0}{1}, \frac{0}{1}\Big) = \frac{\partial g}{\partial z}(0, 0) = 1 \neq 0$$

så är $[0,1,0]$ inte en singulär punkt. Värdet på gradienten till f i punkten $(0,0)$ är

$$\nabla f(0,0) = (0,1)$$

så tangenten till C i punkten $[0,1,0]$ är $z = 0$, eller på homogen form $Z = 0$, det vill säga linjen i oändligheten.

Insättning av $Z = 0$ i (4) ger $X^3 = 0$ så linjen i oändligheten skär kurvan C endast i punkten $[0,1,0]$ och denna skärningspunkt har multiplicitet tre. Alltså är $[0,1,0]$ en inflektionspunkt på C. □

Om inget annat anges så kommer i fortsättningen punkten i oändligheten på en elliptisk kurva att betecknas med $\mathcal{O}$ på grund av följande sats.

Sats 2.3. *Låt*

$$C : y^2 = x^3 + ax^2 + bx + c$$

vara en elliptisk kurva över någon kropp $\mathbb{F}$. Då bildar mängden $C(\mathbb{F})$ av punkter på C tillsammans med kompositionsregeln $+$ definierad i definition 1.2 en abelsk grupp med $\mathcal{O}$ som neutralt element.

Bevis. Allt utom associativiteten hos $+$ har redan visats. Ett bevis av associativiteten finns i Schmitt [3], sida 12. Beviset är egentligen inte svårt men det är många olika fall som behöver behandlas så beviset kräver mycket arbete. □

Sats 2.4. *Låt*

$$C : y^2 = x^3 + ax^2 + bx + c$$

vara en elliptisk kurva över någon kropp $\mathbb{F}$ och låt $C(\mathbb{F}) \ni P = (x_0, y_0) \neq \mathcal{O}$ vara en punkt på C. Då har linjen genom P och $\mathcal{O}$ ekvationen $x = x_0$ och

$$-P = (x_0, -y_0) = P * \mathcal{O}$$

Bevis. Punkterna $P = (x_0, y_0)$ och $(x_0, -y_0)$ tillhör både $C(\mathbb{F})$ och linjen $x = x_0$. Homogenisering av linjen $x = x_0$ ger

$$F(X, Y, Z) = X - x_0 Z = 0$$

och $F(\mathcal{O}) = F([0,1,0]) = 0$ så linjen går även igenom $\mathcal{O}$. Alltså är $P * \mathcal{O} = (x_0, -y_0)$. Punkten $\mathcal{O}$ är en inflektionspunkt på C så $\mathcal{O} * \mathcal{O} = \mathcal{O}$ och då är

$$-P = P * (\mathcal{O} * \mathcal{O}) = P * \mathcal{O} = (x_0, -y_0)$$

□

Följande sats presenterar ett antal formler som underlättar addition av punkter på en elliptisk kurva.

Sats 2.5. *Låt $C : y^2 = f(x) = x^3 + ax^2 + bx + c$ vara en elliptisk kurva över någon kropp $\mathbb{F}$ och låt $\mathcal{O} \in C(\mathbb{F})$ vara punkten på C i oändligheten. Om $C(\mathbb{F}) \ni P_1 = (x_1, y_1) \neq \mathcal{O}$ och $C(\mathbb{F}) \ni P_2 = (x_2, y_2) \neq \mathcal{O}$ är två punkter på C, sådana att $x_1 \neq x_2$, så är*

$$P_1 + P_2 = (k^2 - a - x_1 - x_2, -k(k^2 - a - x_1 - x_2) - m) \tag{5}$$

där

$$k = \frac{y_2 - y_1}{x_2 - x_1} \;\; och \;\; m = y_1 - kx_1$$

Om $\mathbb{F} \ni P = (x_0, y_0) \neq \mathcal{O}$ *är en punkt på* C*, sådan att* $y_0 \neq 0$*, så är*

$$x(2P) = \lambda^2 - a - 2x_0 = \frac{x_0^4 - 2bx_0^2 - 8cx_0 + b^2 - 4ac}{4x_0^3 + 4ax_0^2 + 4bx_0 + 4c} \tag{6}$$

och

$$y(2P) = \lambda(x_0 - x(2P)) - y_0 \tag{7}$$

där

$$\lambda = \frac{3x_0^2 + 2ax_0 + b}{2y_0}$$

Bevis. Följande bevis kommer väsentligen från Silverman [1], sidorna 29-31, men en härledning av uttryck (7) har lagts till.

Linjen genom P_1 och P_2 har ekvationen $y = kx+m$. Insättning av denna ekvation i kurvans ekvation ger

$$(kx + m)^2 = k^2x^2 + 2kmx + m^2 = x^3 + ax^2 + bx + c$$

Om $P_1 * P_2 = (x_3, y_3)$ så är

$$x^3 + (a - k^2)x^2 + (b - 2km)x + c - m^2 = (x - x_1)(x - x_2)(x - x_3)$$

Genom att jämföra x^2-termer så fås att

$$x_3 = k^2 - a - x_1 - x_2$$

och då är $y_3 = kx_3 + m$. Eftersom $P_1 + P_2 = \mathcal{O} * (P_1 * P_2)$ så är enligt sats 2.4

$$P_1 + P_2 = (x_3, -y_3) = (k^2 - a - x_1 - x_2, -k(k^2 - a - x_1 - x_2) - m)$$

Implicit derivering av $y^2 = f(x) = x^3 + ax^2 + bx + c$ ger

$$\frac{dy}{dx} = \frac{f'(x)}{2y} = \frac{3x^2 + 2ax + b}{2y}$$

så tangenten till C i punkten P har ekvationen

$$y = \lambda x + \nu, \;\; \text{där} \;\; \lambda = \frac{f'(x_0)}{2y_0} \;\; \text{och} \;\; \nu = y_o - \lambda x_0$$

På samma sätt som ovan så fås nu att

$$x(2P) = \lambda^2 - a - 2x_0$$

och

$$y(2P) = -\lambda \cdot x(2P) - \nu = \lambda \cdot (-x(2P)) - y_o + \lambda x_0 = \lambda(x_0 - x(2P)) - y_0$$

Ett antal algebraiska omskrivningar av $\lambda^2 - a - 2x_0$ ger sedan uttryck (6) ovan. □

3 Punkter med ändlig ordning

Nu kommer punkter som har ändlig ordning i gruppen bildad av alla punkter på en elliptiska kurva att studeras. Det följer från grundläggande gruppteori att dessa punkter bildar en undergrupp till gruppen av alla punkter på kurvan. Det finns en viktig sats på detta område som brukar kallas Nagell-Lutz sats efter dess oberoende upptäckare Tryggve Nagell och Élisabeth Lutz. Med hjälp av denna sats så kan alla rationella punkter som har ändlig ordning på en rationell elliptisk kurva med heltalskoefficienter bestämmas i ett ändligt antal steg.

Innan Nagell-Lutz sats presenteras så kan det vara bra att känna till följande sats från Silverman [1]. Den ger möjlighet att på ett enkelt sätt avgöra om en elliptisk kurva har några punkter med ordning två och om en given punkt på en elliptisk kurva har ordning två eller ordning tre.

Sats 3.1. *Låt*

$$C : y^2 = f(x) = x^3 + ax^2 + bx + c$$

vara en elliptisk kurva över en kropp $\mathbb{F}$ *och låt* $C(\mathbb{F}) \ni P = (x_0, y_0) \neq \mathcal{O}$ *vara en punkt på* C*. Då har* P *ordning två om och endast om* $y_0 = 0$*.* P *har ordning tre om och endast om* x_0 *är ett nollställe till*

$$\psi_3(x) = 3x^4 + 4ax^3 + 6bx^2 + 12cx + 4ac - b^2$$

betraktat som ett polynom över $\mathbb{F}$*.*

Bevis. Se Silverman [1], sida 40. □

Sats 3.2 (**Nagell-Lutz**)**.** *Låt*

$$C : y^2 = f(x) = x^3 + ax^2 + bx + c$$

vara en rationell elliptisk kurva där a, b, c *är heltal och låt*

$$D = -4a^3c + a^2b^2 + 18abc - 4b^3 - 27c^2$$

vara diskriminanten till $f(x)$*. Om* $P = (x_0, y_0)$ *är en rationell punkt med ändlig ordning på* C *så är* x_0 *och* y_0 *heltal. Om* P *har ordning två så är* $y_0 = 0$*, annars så är* y_0 *en delare till* D*.*

Bevis. Se Silverman [1], sida 56. □

Nagell-Lutz sats har följande följdsats som kraftigt begränsar antalet möjligheter som behöver undersökas när alla punkter av ändlig ordning på en rationell elliptisk kurva ska bestämmas.

Följdsats 3.1. *Låt förutsättningarna vara som i Nagell-Lutz sats. Om* $y_0 \neq 0$ *så är* y_0^2 *en delare till* D*.*

Bevis. Låt $\phi(x) = x^4 - 2bx^2 - 8cx + b^2 - 4ac$, $F(x) = 3x^3 - ax^2 - 5bx + 2ab - 27c$ och $\Phi(x) = -3x^2 - 2ax + a^2 - 4b$. Då är

$$F(x)f(x) + \Phi(x)\phi(x) = D$$

Antag att $y_0 \neq 0$. Då är $2P \neq \mathcal{O}$ så enligt (6) i sats 2.5 så är $\phi(x_0) = x(2P) \cdot 4f(x_0)$ och då är

$$f(x_0)[F(x_0) + 4\Phi(x_0)x(2P)] = D$$

Alltså är y_0^2 en delare till D eftersom $y_0^2 = f(x_0)$. □

Nu följer ett antal exempel på elliptiska kurvor och vilka punkter med ändlig ordning de har. Undergruppen till $C(\mathbb{F})$ bestående av alla punkter med ändlig ordning kommer för enkelhetens skull att betecknas med Φ.

Exempel 3.1. Betrakta den rationella elliptiska kurvan

$$C : y^2 = x^3 - 2$$

Nagell-Lutz sats kan användas för att bestämma alla rationella punkter med ändlig ordning på denna kurva. $y = 0$ insatt i kurvans ekvation ger $x^3 = 2$ så rationella punkter med ordning två saknas. Diskriminanten till $x^3 - 2$ är

$$D = -2^2 \cdot 3^3$$

Om $P = (x, y)$, $y \neq 0$, är en ändlig rationell punkt på C så är x och y heltal och y^2 är en delare till D. Det inses enkelt att y^2 delar D om och endast om

$$y = \pm 1, \pm 2, \pm 3, \pm 6$$

Dessa värden på y ger ekvationerna $x^3 = 3$, $x^3 = 6$, $x^3 = 11$ respektive $x^3 = 38$ som alla saknar heltalslösningar. Alltså är $\mathcal{O}$ den enda rationella punkten med ändlig ordning på C så Φ är den triviala gruppen med ett element.

Exempel 3.2. Betrakta den rationella elliptiska kurvan

$$C : y^2 = x^3 + px$$

där $p \geq 2$ är ett primtal. Nagell-Lutz sats kan återigen användas för att bestämma alla rationella punkter med ändlig ordning på denna kurva. $y = 0$ insatt i kurvans ekvation ger

$$x(x^2 + p) = 0$$

så $(0, 0)$ är den enda rationella punkten med ordning två. Diskriminanten till $x^3 + px$ är

$$D = -4p^3$$

Om $P = (x, y)$, $y \neq 0$, är en ändlig rationell punkt på C så är x och y heltal och y^2 en delare till D. Det är enkelt att se att $y^2 \mid D$ om och endast om

$$y = \pm 1, \pm 2, \pm p, \pm 2p$$

Eftersom $y^2 \geq 0$ för alla y så måste $x \geq 0$ om $P = (x, y) \in C$.

$y = \pm 1$ ger ekvationen $x^3 + px - 1 = 0$ som saknar heltalslösningar.

$y = \pm 2$ ger ekvationen $x^3 + px - 4 = 0$ som endast har lösningen $(x, y) = (1, \pm 2)$ då $p = 3$.

$y = \pm p$ ger ekvationen $x^3 + px - p^2 = 0$ som saknar heltalslösningar.

$y = \pm 2p$ ger ekvationen $x^3 + px - 4p^2 = 0$ som endast har lösningen $(x, y) = (p, \pm 2p)$ då $p = 3$.

Kandidater till att vara rationella punkter på C med ändlig ordning ≥ 3 är alltså $(1, \pm 2)$ då $p = 3$ och $(3, \pm 6)$ då $p = 3$. Om $p = 3$ så är

$$x(2P) = \frac{x^4 - 6x^2 + 9}{4x^3 + 12x}$$

Eftersom $x(2(1, \pm 2)) = 1/4 \notin \mathbb{Z}$ så har $2(1, \pm 2)$ inte ändlig ordning och då kan $(1, \pm 2)$ inte ha ändlig ordning. På samma sätt så har $(3, \pm 6)$ inte ändlig ordning eftersom $x(2(3, \pm 6)) = 1/4 \notin \mathbb{Z}$.

Alltså är $\mathcal{O}$ och $(0, 0)$ de enda punkterna på C som har ändlig ordning och

$$\Phi = \{\mathcal{O}, (0, 0)\} \cong \mathbb{Z}_2$$

för alla primtal p.

Exempel 3.3. Betrakta den rationella kubiska kurvan

$$C : y^2 - y = x^3 - x^2$$

Genom omskrivningen

$$y^2 - y = x^3 - x^2 \iff (8y - 4)^2 = (4x)^3 - 4(4x)^2 + 16$$

så ses att C är birationellt ekvivalent med den elliptiska kurvan

$$C' : v^2 = u^3 - 4u^2 + 16$$

där $v = 8y - 4$ och $u = 4x$. På homogen form så blir sambandet mellan kurvorna

$$\begin{cases} U = 4X \\ V = 8Y - 4Z \\ W = Z \end{cases} \tag{8}$$

där $x = X/Z$, $y = Y/Z$, $u = U/W$ och $v = V/W$. Antag att $P = (u, v)$ är en rationell punkt på C'. Då är u och v heltal. $v = 0$ insatt i ekvationen för C' ger $u^3 - 4u^2 + 16 = 0$ och denna ekvation saknar heltalslösningar så C' saknar punkter med ordning 2. Diskriminanten till $u^3 - 4u^2 + 16$ är

$$D = -2^8 \cdot 11$$

så v^2 delar D om och endast om $v = \pm 1, \pm 2, \pm 4, \pm 8, \pm 16$. Dessa värden på v ger ekvationerna $u^3 - 4u^2 + 15 = 0$, $u^3 - 4u^2 + 12 = 0$, $u^3 - 4u^2 = 0$, $u^3 - 4u^2 - 48 = 0$ respektive $u^3 - 4u^2 - 240$. Den enda av dessa ekvationer som har lösning är

$$u^3 - 4u^2 = u^2(u - 4) = 0$$

som har lösningarna $u = 0$ och $u = 4$. Kandidater till att vara punkter med ändlig ordning på C' är alltså

$$(u, v) = (0, \pm 4) \text{ och } (u, v) = (4, \pm 4)$$

Enligt sats 2.5 så är $2(0, 4) = (4, -4)$, $2(0, -4) = (4, 4)$, $2(4, 4) = (0, 4)$ och $2(4, -4) = (0, -4)$. Eftersom $-(0, 4) = (0, -4)$ och $-(4, 4) = (4, -4)$ så är $(0, 4) + (0, -4) = (4, 4) + (4, -4) = \mathcal{O}'$, där $\mathcal{O}' = [0, 1, 0]$ är punkten i oändligheten på C'. Nu ger sambandet $5P = 2P + 2P + P$ att $5(0, 4) = 5(0, -4) = 5(4, 4) = 5(4, -4) = \mathcal{O}'$ så alla kandidaterna har ändlig ordning. Då måste gruppen bestående av alla rationella punkter på C' med ändlig ordning vara

$$\Phi' = \{\mathcal{O}', (0, 4), (0, -4), (4, 4), (4, -4)\} \cong \mathbb{Z}_5$$

Sambandet $v = 8y - 4$ ger $y = (v + 4)/8$ och $u = 4x$ ger $x = u/4$ så

$$\begin{aligned}
(u, v) = (0, 4) &\iff (x, y) = (0, 1) \\
(u, v) = (0, -4) &\iff (x, y) = (0, 0) \\
(u, v) = (4, 4) &\iff (x, y) = (1, 1) \\
(u, v) = (4, -4) &\iff (x, y) = (1, 0)
\end{aligned}$$

Sambandet (8) ger att punkten $\mathcal{O}' = [0, 1, 0]$ på C' motsvarar punkten $\mathcal{O} = [0, 1, 0]$ på C. Alltså är gruppen bestående av alla punkter på C med ändlig ordning

$$\Phi = \{\mathcal{O}, (0, 1), (0, 0), (1, 1), (1, 0)\} \cong \mathbb{Z}_5$$

Föregående tre exempel har behandlat kurvor över de rationella talen. Nu följer ett exempel på kurvor över några ändliga kroppar. Till skillnad från fallet med oändliga kroppar går det nu att vara säker på att alla punkter på kurvan har ändlig ordning eftersom det endast kan finnas ett ändligt antal punkter på kurvan.

Exempel 3.4. Betrakta den elliptiska kurvan

$$C : y^2 = x^3 + 1$$

över kroppen $\mathbb{Z}_p$. $a = b = 0$ och $c = 1$ så polynomet ψ_3 i sats 3.1 är

$$\psi_3(x) \equiv 3x^4 + 12x \pmod{p}$$

Låt $p = 5$, d.v.s. betrakta C över $\mathbb{Z}_5$. Genom att prova alla möjliga värden på x och y så fås att

$$C(\mathbb{Z}_5) = \{\mathcal{O}, (0, 1), (0, 4), (2, 2), (2, 3), (4, 0)\}$$

Eftersom det endast finns en abelsk grupp med ordning 6 så måste $C(\mathbb{Z}_5) \cong \mathbb{Z}_6$.

Låt nu istället $p = 7$. Då är

$$\begin{aligned}
&C(\mathbb{Z}_7) = \\
&= \{\mathcal{O}, (0, 1), (0, 6), (1, 3), (1, 4), (2, 3), (2, 4), (3, 0), (4, 3), (4, 4), (5, 0), (6, 0)\}
\end{aligned}$$

så $|C(\mathbb{Z}_7)| = 12$. Enligt sats 3.1 så är $(3,0)$, $(5,0)$ och $(6,0)$ de enda punkterna med ordning 2. Då måste $C(\mathbb{Z}_7) \cong \mathbb{Z}_3 \oplus \mathbb{Z}_2 \oplus \mathbb{Z}_2$ eftersom $\mathbb{Z}_3 \oplus \mathbb{Z}_2 \oplus \mathbb{Z}_2$ är den enda abelska gruppen med ordning 12 som har 3 element med ordning 2.

Låt nu $p = 11$. Då är

$$C(\mathbb{Z}_{11}) =$$
$$= \{\mathcal{O}, (0,1), (0,10), (2,3), (2,8), (5,4), (5,7), (7,5), (7,6), (9,2), (9,9), (10,0)\}$$

så även i detta fall är $|C(\mathbb{Z}_{11})| = 12$. Enligt sats 3.1 så är $(10,0)$ den enda punkten med ordning 2. Då måste $C(\mathbb{Z}_{11}) \cong \mathbb{Z}_{12}$ eftersom den enda gruppen av ordning 12 som har exakt 1 element av ordning 2 är den cykliska gruppen med 12 element.

Nu kommer ett alternativ till att använda Nagell-Lutz sats för att bestämma Φ att presenteras. Denna metod kommer från Silverman [1] och bygger på att en rationell elliptisk kurva kan reduceras modulo ett primtal för att få en ny kurva över en ändlig kropp.

Definition 3.1. *Låt p vara ett fixerat primtal. Då skrivs reduktion modulo p av ett heltal a som $\tilde{a}$.*

Om $[a,b,c] \in \mathbb{P}^2(\mathbb{Q})$ så kallas $[a,b,c]$ *normaliserad* om a, b och c är heltal som saknar gemensamma faktorer. Varje punkt $P \in \mathbb{P}^2(\mathbb{Q})$ är ekvivalent med en normaliserad punkt som är unik bortsett från en faktor -1 så varje punkt $P \in \mathbb{P}^2(\mathbb{Q})$ kan antas vara normaliserad.

Definition 3.2. *Låt p vara ett fixerat primtal. Om $P = [a,b,c] \in \mathbb{P}^2(\mathbb{Q})$ så definieras reduktion modulo p av P som*

$$\tilde{P} = [\tilde{a}, \tilde{b}, \tilde{c}]$$

Notera att $\tilde{P} \in \mathbb{P}^2(\mathbb{Z}_p)$ eftersom $\tilde{P} \neq [0,0,0]$ då a, b och c saknar gemensamma faktorer. Reduktion modulo p av ett $P \in \mathbb{P}^2(\mathbb{Q})$ är alltså en väldefinierad avbildning $\mathbb{P}^2(\mathbb{Q}) \to \mathbb{P}^2(\mathbb{Z}_p)$ eftersom P kan antas vara normaliserad.

Lemma 3.1. *Om $P = (a,b) = [a,b,1] \in \mathbb{A}^2(\mathbb{Q})$ så tillhör $\tilde{P}$, dess reduktion modulo ett primtal p, $\mathbb{A}^2(\mathbb{Z}_p)$ om och endast om a och b har nämnare relativt prima med p.*

Bevis. Antag att $a = \frac{c}{dp}$ och $b = \frac{e}{fp}$ för några heltal c, d, e, f sådana att $d, f \neq 0$. Då är

$$P = [a,b,1] = \left[\frac{c}{dp}, \frac{e}{fp}, 1\right] \sim [cf, ec, dfp]$$

så $\tilde{P} = [\tilde{c}\tilde{f}, \tilde{e}\tilde{c}, 0] \notin \mathbb{A}^2(\mathbb{Z}_p)$.

Antag nu istället att $a = \frac{c}{d}$ och $b = \frac{e}{f}$ för några heltal c, d, e, f sådana att $d, f \neq 0$ och $\mathrm{SGD}(d,p) = \mathrm{SGD}(f,p) = 1$. Då är

$$P = [a,b,1] = \left[\frac{c}{d}, \frac{e}{f}, 1\right] \sim [cf, de, df]$$

där $\mathrm{SGD}(df,p) = 1$ så $\tilde{P} \in \mathbb{A}^2(\mathbb{Z}_p)$. □

Reduktionen modulo p av en rationell elliptisk kurva definieras enligt följande definition.

Definition 3.3. *Låt*

$$C : y^2 = x^3 + ax^2 + bx + c$$

vara en kubisk kurva över $\mathbb{Q}$ *där* $a, b, c \in \mathbb{Z}$ *och låt* p *vara ett primtal. Då definieras reduktion modulo* p *av* C *som kurvan*

$$\tilde{C} : y^2 = x^3 + \tilde{a}x^2 + \tilde{b}x + \tilde{c}$$

över $\mathbb{Z}_p$.

Sats 3.3. *Låt*

$$C : y^2 = x^3 + ax^2 + bx + c$$

vara en icke-singulär kubisk kurva över $\mathbb{Q}$ *med heltalskoefficienter och låt* $\mathcal{O} \in C(\mathbb{Q})$ *vara det neutrala elementet i gruppoperationen på* C*. Antag att* $\tilde{C}$ *är icke-singulär och låt* $\tilde{\mathcal{O}} \in \tilde{C}(\mathbb{Q})$ *vara det neutrala elementet i gruppoperationen på* $\tilde{C}$*. Då är reduktion modulo* p *en grupphomomorfism* $\mathbb{P}^2(\mathbb{Q}) \to \mathbb{P}^2(\mathbb{Z}_p)$.

Bevis. Se Silverman [1], sida 254. □

Följdsats 3.2. *Låt förutsättningarna vara som i föregående sats. Låt* Φ *vara undergruppen till* $C(\mathbb{Q})$ *bestående av alla element med ändlig ordning. Då är reduktion modulo* p *en isomorfism från* Φ *till en undergrupp i* $C(\mathbb{Z}_p)$.

Bevis. Om $P = (a, b) \in \Phi$ så är a och b heltal och då är $\tilde{P} \neq \tilde{O}$ enligt lemma 3.1. Då består kärnan till reduktion modulo p endast av $\mathcal{O}$. Enligt första isomorfisatsen för grupper så är då reduktion modulo p en isomorfism från Φ till en undergrupp i $C(\mathbb{Z}_p)$. □

Följdsats 3.2 kan användas för att bestämma Φ till en rationell elliptiska kurva C genom att reducera C modulo några lämpliga primtal och sedan undersöka punkterna på dessa nya elliptiska kurvor. Denna metod är ofta ett kraftfullt verktyg för att bestämma Φ.

Exempel 3.5. Låt C vara den rationella kurvan

$$C : y^2 = f(x) = x^3 + bx + c$$

där $b, c \in \mathbb{Z}$ är sådana att $b \equiv 11 \pmod{15}$ och $c \equiv 4 \pmod{15}$. Diskriminanten till $f(x)$ är $D = 4b^3 + 27c^2$. Antag att $D \neq 0$ så att C är icke-singulär. Reduktion modulo 3 av C ger kurvan

$$\tilde{C}_3 : y^2 = x^3 + 2x + 1$$

över $\mathbb{Z}_3$. Eftersom $b \equiv 2 \pmod{3}$ och $c \equiv 1 \pmod{3}$ så är $D \equiv 2 \pmod{3}$ och då är $\tilde{C}_3$ icke-singulär. Gruppen med punkter på kurvan är

$$\tilde{C}_3(\mathbb{Z}_3) = \{\tilde{\mathcal{O}}, (0,1), (0,2), (1,1), (1,2), (2,1), (2,2)\}$$

så $\tilde{C}_3(\mathbb{Z}_3) \cong \mathbb{Z}_7$. Då är Φ isomorf med en undergrupp till $\mathbb{Z}_7$ så Φ har 1 eller 7 element. Reduktion modulo 5 av C ger kurvan

$$\tilde{C}_5 : y^2 = x^3 + x + 4$$

över $\mathbb{Z}_5$. Eftersom $b \equiv 1 \pmod 5$ och $c \equiv 4 \pmod 5$ så är $D \equiv 1 \pmod 5$ och då är $\tilde{C}_5$ icke-singulär. Gruppen med punkter på kurvan är

$$\tilde{C}_5(\mathbb{Z}_5) = \{\tilde{\mathcal{O}}, (0,2), (0,3), (1,1), (1,4), (2,2), (2,3), (3,2), (3,3)\}$$

$|\tilde{C}_5(\mathbb{Z}_5)| = 9$ så Φ är ismorf med en undergrupp till en grupp med ordning 9. Då måste $|\Phi| = 1$ eftersom en grupp med 9 element inte kan ha några undergrupper med 7 element. Alltså är $\Phi = \{\mathcal{O}\}$.

4 Rang av en elliptisk kurva

I föregående avsnitt så undersöktes punkterna med ändlig ordning på en elliptisk kurva. En elliptisk kurva har ofta också punkter med oändligt ordning och dessa kommer nu att undersökas. Det som gör detta möjligt är följande mycket djupa och viktiga resultat av Louis Mordell och André Weil.

Sats 4.1 (**Mordell-Weil**). *Låt C vara en elliptiska kurva över en talkropp $\mathbb{F}$. Då är gruppen $C(\mathbb{F})$ ändligt genererad.*

Bevis. Se Silverman [2], sidorna 201-220. □

Om C är en rationell elliptisk kurva så följer från Mordell-Weils sats att

$$C(\mathbb{Q}) \cong \underbrace{\mathbb{Z} \oplus \mathbb{Z} \oplus \cdots \oplus \mathbb{Z}}_{r \text{ termer}} \oplus \mathbb{Z}_{p_1^{e_1}} \oplus \mathbb{Z}_{p_2^{e_2}} \oplus \cdots \oplus \mathbb{Z}_{p_n^{e_n}}$$

för några primtal $p_1, p_2, \ldots, p_n$ och några heltal $e_1, e_2, \ldots, e_n$. Heltalet r kallas för *rangen* av $C(\mathbb{Q})$.

Tyvärr så kan inte Mordell-Weils sats användas för att bestämma värdet på rangen till en elliptiska kurva men idéerna från beviset av satsen kan användas för att konstruera en metod som ofta kan göra detta. Metoden, som nu kommer att presenteras i sammanfattat form, kommer från Silverman [1], sidorna 83 till 94.

Metoden fungerar genom att betrakta ett par av elliptiska kurvor C och $\bar{C}$ som är relaterade till varandra på följande sätt. Låt

$$C : y^2 = x^3 + ax^2 + bx$$

vara en elliptisk kurva över $\mathbb{Q}$ där a och b är heltal. Då definieras $\bar{a} = -2a$, $\bar{b} = a^2 - 4b$ och den elliptiska kurvan $\bar{C}$ definieras

$$\bar{C} : y^2 = x^3 + \bar{a}x^2 + \bar{b}x$$

Mellan kurvorna C och $\bar{C}$ finns ett nära samband och $\bar{C}$ fungerar som hjälpkurva när rangen av C ska bestämmas. $C(\mathbb{Q})$ och $\bar{C}(\mathbb{Q})$ kommer för enkelhetens skull att betecknas med Γ respektive $\bar{\Gamma}$. Notera att punkten $(0,0)$ alltid tillhör Γ och $\bar{\Gamma}$.

Låt $\mathbb{Q}^*$ beteckna den multiplikativa gruppen bestående av alla nollskilda rationella tal och definiera $\mathbb{Q}^{*2}$ som undergruppen

$$\mathbb{Q}^{*2} = \{q^2 : q \in \mathbb{Q}^*\}$$

till $\mathbb{Q}^*$. Definiera avbildningen $\alpha : \Gamma \to \mathbb{Q}^*/\mathbb{Q}^{*2}$ enligt

$$\begin{aligned} \alpha(\mathcal{O}) &= 1 \pmod{\mathbb{Q}^{*2}} \\ \alpha(0,0) &= b \pmod{\mathbb{Q}^{*2}} \\ \alpha(x,y) &= x \pmod{\mathbb{Q}^{*2}}, \text{ om } x \neq 0 \end{aligned}$$

Avbildningen $\bar{\alpha} : \bar{\Gamma} \to \mathbb{Q}^*/\mathbb{Q}^{*2}$ definieras analogt med α. Både α och $\bar{\alpha}$ är homomorfismer (se Silverman [1], sida 85).

Sats 4.2. *För rangen r av Γ gäller att*

$$2^r = \frac{|\alpha(\Gamma)| \cdot |\bar{\alpha}(\bar{\Gamma})|}{4}$$

Bevis. Se Silverman [1], sida 91. □

Föregående sats säger alltså att rangen av C kan bestämmas genom att $|\alpha(\Gamma)|$ och $|\bar{\alpha}(\bar{\Gamma})|$ bestäms och det är detta faktum som kommer att användas här.

Sats 4.3. *Om $P = (x, y)$ är en punkt på C så kan x och y skrivas på formen*

$$x = \frac{m}{e^2} \text{ och } y = \frac{n}{e^3}$$

för några heltal m, n och e sådana att $e > 0$ och $\mathrm{SGD}(m, e) = \mathrm{SGD}(n, e) = 1$.

Bevis. Se Silverman [1], sida 68. □

Låt $P = (x, y) = (m/e^2, n/e^3)$, med m,n,e som i sats 4.3, vara en punkt på C. Om $m = 0$ så är $P = (0, 0)$ och $\alpha(0, 0) = b \pmod{\mathbb{Q}^{*2}}$ så det är alltid så att $b \pmod{\mathbb{Q}^{*2}} \in \alpha(\Gamma)$.

Om $n = 0$ så är $y = 0$ och då följer att $x = 0$ eller $x^2 + ax + b = 0$. Fallet $P = (0, 0)$ är redan utrett så antag att $x^2 + ax + b = 0$. Om $a^2 - 4b$ är en kvadrat så är

$$\left(\frac{-a+d}{2}, 0\right), \left(\frac{-a-d}{2}, 0\right) \in \Gamma$$

där $d = \sqrt{a^2 - 4b}$. Då är alltså $(-a + d)/2, (-a - d)/2 \in \alpha(\Gamma)$.

Antag nu att $m \neq 0$ och $n \neq 0$ och låt $b_1 = \pm\,\mathrm{SGD}(m, b)$, där tecknet väljs så att $mb_1 > 0$. Då finns ett $m_1 > 0$ sådant att $m = b_1 m_1$ och ett b_2 sådant att $b = b_1 b_2$ och $\mathrm{SGD}(m_1, b_2) = 1$. Det följer då (se Silverman [1], sida 92) att

$$N^2 = b_1 M^4 + a M^2 e^2 + b_2 e^4 \tag{9}$$

där $M^2 = m_1$ och $N^2 = b_1 m_1^2 + a m_1 e^2 + b_2 e^4$. Då kan x skrivas på formen

$$x = \frac{b_1 M^2}{e^2}$$

så $x = b_1 \pmod{\mathbb{Q}^{*2}}$.

Ordningen av $\alpha(\Gamma)$ kan alltså bestämmas genom att faktorisera b på formen $b = b_1 b_2$ och för varje möjlig faktorisering undersöka motsvarande ekvation (9) med

M, e och N som variabler. Om ekvationen har en lösning (M, e, N), sådan att $M \neq 0$, så är $b_1 \pmod{\mathbb{Q}^{*2}} \in \alpha(\Gamma)$. Eftersom $\mathrm{SGD}(m, e) = 1$ så följer att $\mathrm{SGD}(M, e) = 1$ om ekvationen har lösning.

Notera att om $(-a+d)/2 \in \alpha(\Gamma)$ eller $(-a-d)/2 \in \alpha(\Gamma)$ så har b faktoriseringen $b = \frac{-a+d}{2} \cdot \frac{-a-d}{2}$ så detta fall behöver inte undersökas separat.

Exempel 4.1. Betrakta återigen den rationella elliptiska kurvan

$$C : y^2 = x^3 + px, \text{ där } p \text{ är ett primtal}$$

vars punkter med ändlig ordning undersöktes i exempel 3.2. Koefficienterna $a = 0$ och $b = p$ ger $\bar{a} = 0$ och $\bar{b} = -4p$ så

$$\bar{C} : y^2 = x^3 - 4px$$

Möjliga faktoriseringar $b = b_1 b_2$ är $b = (-1)(-p)$ och $b = 1 \cdot p$ så b_1 och b_2 måste ha samma tecken. Om b_1 och b_2 är negativa så saknar ekvationen $N^2 = b_1 M^4 + b_2 e^4$ lösning så b_1 och b_2 måste vara positiva. $\alpha(\mathcal{O}) = 1$ och $\alpha(0, 0) = p$ så

$$\alpha(\Gamma) = \{1, p\}$$

Möjliga faktoriseringar $\bar{b} = \bar{b}_1 \bar{b}_2$ är $b = 1 \cdot (-4p) = (-1) \cdot 4p = 4 \cdot (-p) = (-4) \cdot p = 2 \cdot (-2p) = (-2) \cdot 2p$ så $\bar{b}_1 = \pm 1, \pm 2, \pm 4, \pm p, \pm 2p, \pm 4p$. Eftersom $4 \equiv 1 \pmod{\mathbb{Q}^{*2}}$ så är

$$\bar{\alpha}(\bar{\Gamma}) \subseteq \{\pm 1, \pm 2, \pm p, \pm 2p\}$$

Då har alltså Γ rang 0, 1 eller 2 enligt sats 4.2.

Låt nu primtalet p vara sådant att $p \equiv 7 \pmod{16}$. Det är klart att $\bar{\alpha}(\bar{\Gamma}) \supseteq \{1, -p\}$ eftersom $\bar{\alpha}(\mathcal{O}) = 1$ och $\bar{\alpha}(0, 0) = -p$. Antag att ekvationen

$$N^2 = -M^4 + 4pe^4 \tag{10}$$

har en lösning. Då är $N^2 \equiv 15M^4 + 12e^4 \pmod{16}$. Kvadratiska rester modulo 16 är 0, 1, 4 och 9 så $N^2 \equiv 0, 1, 4, 9 \pmod{16}$, $M^4 \equiv 0, 1 \pmod{16}$ och $e^4 \equiv 0, 1 \pmod{16}$. Då måste $M^4 \equiv e^4 \equiv 0 \pmod{16}$. Men $\mathrm{SGD}(M, e) = 1$ så ekvation (10) saknar lösning. Alltså är $-1 \notin \bar{\alpha}(\bar{\Gamma})$. Antag att ekvationen

$$N^2 = 2M^4 - 2pe^4 \tag{11}$$

har en lösning. Då finns ett N_0 sådant att $2N_0^2 = M^4 - pe^4$ och då är $2N_0^2 \equiv M^4 + 9e^4 \pmod{16}$. De möjligheter som finns är $2N_0^2 \equiv 0, 2, 8 \pmod{16}$, $M^4 \equiv 0, 1 \pmod{16}$ och $e^4 \equiv 0, 1 \pmod{16}$. Då måste $M^4 \equiv e^4 \equiv 0 \pmod{16}$. Men $\mathrm{SGD}(M, e) = 1$ så ekvation (11) saknar lösning och då är $2 \notin \bar{\alpha}(\bar{\Gamma})$ och $-2p \notin \bar{\alpha}(\bar{\Gamma})$. Antag till sist att ekvationen

$$N^2 = -2M^4 + 2pe^4 \tag{12}$$

har en lösning. Då finns ett N_0 sådant att $2N_0^2 = -M^4 + pe^4$ och då är $2N_0^2 \equiv 15M^4 + 7e^4 \pmod{16}$. Som tidigare så är $2N_0^2 \equiv 0, 2, 8 \pmod{16}$, $M^4 \equiv 0, 1 \pmod{16}$ och $e^4 \equiv 0, 1 \pmod{16}$ de möjligheter som finns och då måste $M^4 \equiv e^4 \equiv 0 \pmod{16}$.

Som tidigare följer då att $-2 \notin \bar{\alpha}(\bar{\Gamma})$ och $2p \notin \bar{\alpha}(\bar{\Gamma})$. Eftersom $\bar{\alpha}(\bar{\Gamma})$ är en undergrupp till $\mathbb{Q}^*/\mathbb{Q}^{*2}$ och $p \cdot (-p) \equiv -1 \pmod{\mathbb{Q}^{*2}}$ så är $p \notin \bar{\alpha}(\bar{\Gamma})$. Alltså är

$$\bar{\alpha}(\bar{\Gamma}) = \{1, p\}$$

då $p \equiv 7 \pmod{16}$ och då har Γ rang 0 enligt sats 4.2. Genom att kombinera detta med resultatet från exempel 3.2 så fås att

$$C(\mathbb{Q}) = \{\mathcal{O}, (0,0)\} \cong \mathbb{Z}_2$$

då $p \equiv 7 \pmod{16}$.

5 Elliptiska kurvor över de komplexa talen

Genom att betrakta elliptisk kurvor över de komplexa talen så kommer nu en något mer analytisk aspekt av elliptiska kurvor att presenteras. Det kommer visa sig att en elliptisk kurva över $\mathbb{C}$ i själva verket är en torus.

För att kunna presentera materialet i detta avsnitt, som i huvudsak är en sammanfattning av kapitel 6 i Silverman [2], så måste först begreppet gitter introduceras och Weierstrass $\wp$-funktion samt Eisenstein serie definieras.

Definition 5.1. *Ett* gitter (i $\mathbb{C}$) *är en undergrupp till den additiva gruppen* $\mathbb{C}$ *på formen*

$$L = \mathbb{Z}\omega_1 + \mathbb{Z}\omega_2 = \{n_1\omega_1 + n_2\omega_2 : n_1, n_2 \in \mathbb{Z}\}$$

där $\omega_1 \in \mathbb{C}$ *och* $\omega_2 \in \mathbb{C}$ *är* $\mathbb{R}$*-linjärt oberoende.*

Två gitter L_1 och L_2 sägs vara lika med avseende på *skalning* om det finns ett $c \in \mathbb{C}, c \neq 0$ sådant att $L_1 = cL_2$.

Definition 5.2. *Låt* L *vara ett gitter.* Weierstrass $\wp$-funktion*, för gittret* L*, definieras som*

$$\wp(z) = \wp(z; L) = \frac{1}{z^2} + \sum_{\substack{\omega \in L \\ \omega \neq 0}} \Big[\frac{1}{(z-\omega)^2} - \frac{1}{\omega^2}\Big]$$

där $z \in \mathbb{C}$.

Weierstrass $\wp$-funktion har bland annat följande egenskaper (Silverman [2], sidorna 153-154). Det är en meromorfisk funktion som har poler endast i punkterna som tillhör gittret L och dessa poler har ordning 2. Serien i definitionen av Weierstrass $\wp$-funktion konvergerar absolut och likformigt i varje kompakt delmängd till $\mathbb{C} \setminus L$. Derivatan av $\wp(z)$ är

$$\wp'(z) = -2\sum_{\omega \in L} \frac{1}{(z-\omega)^3}$$

En viktig egenskap hos $\wp$ är att det är en så kallad *elliptisk funktion* med avseende på gittret L. Detta betyder att

$$\wp(z+\omega) = \wp(z) \quad \text{för alla } \omega \in L, z \in \mathbb{C}$$

Definition 5.3. *Låt L vara ett gitter.* Eisensteins serie *för L, med vikt $2k$, definieras som serien*

$$G_{2k} = G_{2k}(L) = \sum_{\substack{\omega \in L \\ \omega \neq 0}} \frac{1}{\omega^{2k}}$$

Eisenstein serie är absolutkonvergent för alla $k \geq 2$ (Silverman [2], sidorna 153-154) men det är endast serierna $g_2 = 60G_4$ och $g_3 = 140G_6$ som kommer att vara intressanta i detta sammanhang.

Sats 5.1. (a) *Låt L vara ett gitter och låt C vara kurvan*

$$C : y^2 = 4x^3 - g_2 x - g_3$$

över $\mathbb{C}$, där $g_2 = 60G_4$ och $g_3 = 140G_6$ enligt ovan. Då är C en elliptisk kurva och avbildningen $\theta : \mathbb{C}/L \to C(\mathbb{C})$ definierad enligt

$$z \mapsto [\wp(z; L), \wp'(z; L), 1]$$

är en gruppisomorfism.
(b) *Låt nu istället C vara en given elliptisk kurva över $\mathbb{C}$. Då existerar ett gitter L, unikt upp till skalning, och en gruppisomorfism $\theta : \mathbb{C}/L \to C(\mathbb{C})$ också definierad enligt*

$$z \mapsto [\wp(z; L), \wp'(z; L), 1]$$

Bevis. Se Silverman [2], sidorna 158-161. □

Notera att för avbildningen θ i föregående sats så gäller för alla $\omega \in L$ att $\theta(\omega) = [0, 1, 0] = \mathcal{O}$ eftersom

$$[\wp(z), \wp'(z), 1] \sim [(z - \omega)^3 \wp(z), (z - \omega)^3 \wp'(z), (z - \omega)^3]$$

för alla $\omega \in L$. Faktorgruppen $\mathbb{C}/L$ är en torus så en elliptisk kurva över $\mathbb{C}$ är faktiskt isomorf med en torus.

En *endomorfism* är en homomorfism, definierad genom rationella funktioner, från en elliptisk kurva tillbaka på samma elliptiska kurva. Eftersom punkterna på en elliptisk kurva $C(\mathbb{C})$ bildar en abelsk grupp så har $C(\mathbb{C})$ alltid endomorfismerna $\phi_n : C(\mathbb{C}) \to C(\mathbb{C})$ definierade enligt

$$\phi_n(P) = nP$$

för alla $P \in C(\mathbb{C})$ och alla heltal n. Det är naturligt att fråga sig om det existerar fler endomorfismer än dessa och på grund av detta så finns följande definition.

Definition 5.4. *Låt C vara en elliptisk kurva över de komplexa talen. C sägs ha* komplex multiplikation *om det finns en endomorfism ϕ på $C(\mathbb{C})$, sådan att ϕ inte är på formen $P \mapsto nP$ för något heltal n, där $P \in C(\mathbb{C})$. En sådan endomorfism ϕ kallas för en* komplex multiplikation *på C.*

Även om en elliptisk kurva över $\mathbb{C}$ har komplex multiplikation så kan endomorfismerna på $C(\mathbb{C})$ inte ha vilken form som helst, vilket följande två satser visar.

Sats 5.2. *Låt $\phi : C(\mathbb{C}) \to C(\mathbb{C})$ vara en endomorfism på en elliptisk kurva $C(\mathbb{C})$. Då existerar ett $\alpha \in \mathbb{C}$ sådant att ϕ ges av $f : \mathbb{C}/L \to \mathbb{C}/L$ definierad enligt*

$$f(z) = \alpha z$$

där L är ett gitter associerat med C enligt sats 5.1b.

Bevis. Se Silverman [1], sida 204. □

Sats 5.3. *Låt C vara en elliptisk kurva över $\mathbb{C}$ och låt L vara gittret associerat med C. Då är $\phi : \mathbb{C}/L \to \mathbb{C}/L$ definierad enligt $\phi(z) = \alpha z$, $z \in \mathbb{C}$, en endomorfism om och endast om*

$$\alpha L \subseteq L$$

Bevis. Se Silverman [1], sida 204. □

Det finns mer som kan sägas om endomorfismerna. Bland annat så har talet α i de två föregående satserna följande egenskaper.

Sats 5.4. *Låt C vara en elliptisk kurva över $\mathbb{C}$ med en komplex multiplikation $\phi : \mathbb{C}/L \to \mathbb{C}/L$ definierad enligt $\phi(z) = \alpha z$, $z \in \mathbb{C}$. Då finns $A, B \in \mathbb{Z}$ sådana att*

$$\alpha^2 + A\alpha + B = 0$$

och dessutom är α inte reellt så diskriminanten $A^2 - 4B < 0$.

Bevis. Låt $\{\omega_1, \omega_2\}$ vara en bas till L, det vill säga $L = \mathbb{Z}\omega_1 + \mathbb{Z}\omega_2$. Eftersom $\alpha L \subseteq L$ så måste det finnas heltal $a, b, c, d \in \mathbb{Z}$ sådana att

$$\begin{cases} \alpha\omega_1 = a\omega_1 + b\omega_2 \\ \alpha\omega_2 = c\omega_1 + d\omega_2 \end{cases}, \quad \text{eller} \quad \begin{cases} (a-\alpha)\omega_1 + b\omega_2 = 0 \\ c\omega_1 + (d-\alpha)\omega_2 = 0 \end{cases} \tag{13}$$

Om $f(z) = \alpha z$ är en komplex multiplikation så måste det finnas en icke-trivial lösning ω_1, ω_2 till (13). Då följer att determinanten

$$\begin{vmatrix} a-\alpha & b \\ c & d-\alpha \end{vmatrix} = \alpha^2 - (a+d)\alpha + ad - bc = 0$$

så det finns heltal A, B sådana att $\alpha^2 + A\alpha + B = 0$.

Antag att $\alpha \in \mathbb{R}$. Eftersom ω_1 och ω_2 är $\mathbb{R}$-linjärt oberoende så måste $\alpha - a = 0$ i 13, det vill säga $\alpha \in \mathbb{Z}$. Men detta är en motsägelse eftersom ϕ är en komplex multiplikation. Alltså kan α inte vara ett reellt tal. Då följer att diskriminanten $A^2 - 4B < 0$. □

Låt $\mathrm{End}(C)$ beteckna mängden av alla endomorfismer på den elliptiska kurvan C över $\mathbb{C}$. Genom att definiera addition av två endomorfismer ϕ_1 och ϕ_2 som

$$(\phi_1 + \phi_2)(P) = \phi_1(P) + \phi_2(P), \quad P \in C(\mathbb{C})$$

och multiplikation av ϕ_1 och ϕ_2 som

$$(\phi_1\phi_2)(P) = \phi_1(\phi_2(P)), \quad P \in C(\mathbb{C})$$

så bildar $\mathrm{End}(C)$ en ring kallad *endomorfiringen* till C.

Följande sats visar vilka endomorfiringar som är möjliga. För att kunna presentera satsen så måste först definitionen av begreppet en ordning i en talkropp göras.

Definition 5.5. *Låt K vara en talkropp med grad r, det vill säga en ändlig kroppsutvidgning av $\mathbb{Q}$ sådan att $[K : Q] = r$. En* ordning *R i K är en delring till K sådan att R är ändligt genererad som abelsk grupp och har rang r.*

Sats 5.5. *Låt C vara en elliptisk kurva över $\mathbb{C}$ och låt ω_1, ω_2 vara generatorer till gittret L associerat med C. Om C inte har komplex multiplikation så är*

$$\mathrm{End}(C) \cong \mathbb{Z}$$

annars så är

$$\mathrm{End}(C) \cong R$$

där R är en ordning i talkroppen $\mathbb{Q}(\omega_1/\omega_2)$, som i detta fall är en imaginär kvadratisk kroppsutvidgning av $\mathbb{Q}$.

Bevis. Beviset från Silverman [2], sida 164, presenteras här eftersom alla nödvändiga delar redan är behandlade.

Om C inte har komplex multiplikation så följer från definitionen av komplex multiplikation att $\mathrm{End}(C) \cong \mathbb{Z}$.

Antag nu att C har komplex multiplikation och låt $R = \{\alpha \in \mathbb{C} : \alpha L \subseteq L\}$. Från sats 5.3 följer att $R \cong \mathrm{End}(C)$ och från definitionen av komplex multiplikation följer att $R \supset \mathbb{Z}$. Enligt sats 5.4 så finns för alla $\alpha \in R$ heltal A och B sådana att

$$\alpha^2 + A\alpha + B = 0$$

så alla element i R är algebraiska heltal. Låt nu $\tau = \omega_1/\omega_2$. Då är gittren $L = \mathbb{Z}\omega_1 + \mathbb{Z}\omega_2$ och $\mathbb{Z} + \mathbb{Z}\tau$ lika med avseende på skalning eftersom

$$\frac{1}{\omega_2} L = \mathbb{Z} + \mathbb{Z}\tau$$

Antag nu att $\alpha \in R$ och $\alpha \notin \mathbb{Z}$ så att $b \neq 0$. Då finns för varje $\alpha \in R$ heltal a, b, c, d sådana att

$$\begin{cases} \alpha = a + b\tau \\ \alpha\tau = c + d\tau \end{cases}$$

Från detta ekvationssystem följer att

$$\tau^2 + \tau\Big(\frac{a-d}{b}\Big) - \frac{c}{b} = 0 \tag{14}$$

Eftersom $\tau \notin \mathbb{R}$ så är (14) minimalpolynom till τ över $\mathbb{Q}$ och då följer att $\mathbb{Q}(\tau)$ är en imaginär kvadratisk kroppsutvidgning av $\mathbb{Q}$. Ringen $R \supset \mathbb{Z}$ är en delring till $\mathbb{Q}(\tau) = \mathbb{Q}(\alpha)$ så R är ändligt genererad som abelsk grupp. Alltså är R en ordning i $\mathbb{Q}(\tau)$ när C har komplex multiplikation. □

Nu följer två exempel på gitter och dess associerade elliptiska kurvor. Dessa kurvor visar sig ha komplex multiplikation.

Exempel 5.1. Betrakta gittret

$$L = \mathbb{Z}[i] = \{a + bi : a, b \in \mathbb{Z}\}$$

det vill säga de Gaussiska heltalen. Eftersom $iL{=}L$ så är $G_6(L) = G_6(iL)$. Men då är

$$G_6(L) = \sum_{\substack{\omega \in L \\ \omega \neq 0}} \frac{1}{\omega^6} = \sum_{\substack{\omega \in iL \\ \omega \neq 0}} \frac{1}{\omega^6} = \sum_{\substack{\omega \in L \\ \omega \neq 0}} \frac{1}{(i\omega)^6} = -\sum_{\substack{\omega \in L \\ \omega \neq 0}} \frac{1}{\omega^6} = -G_6(L)$$

så det måste vara så att $g_3 = G_6(L) = 0$. Då följer att $\mathbb{C}/L$ är isomorf med den elliptiska kurvan $v^2 = 4u^3 - g_2 u$. Genom att sätta

$$u = \frac{\sqrt{g_2}}{2} x \qquad \text{och} \qquad v = \sqrt{\frac{g_2 \sqrt{g_2}}{2}} y$$

så följer att $v^2 = 4u^3 - g_2 u$ är birationellt ekvivalent med den elliptiska kurvan

$$C : y^2 = x^3 - x$$

Alltså är $\mathbb{C}/L$ isomorf med $C(\mathbb{C})$.

Från det faktum att $iL = L$ så följer också att det finns en endomorfism $\mathbb{C}/L \to \mathbb{C}/L$ definierad enligt

$$z \mapsto iz$$

så C har komplex multiplikation. Vidare följer att

$$\wp(iz) = \frac{1}{(iz)^2} + \sum_{\substack{\omega \in L \\ \omega \neq 0}} \Big[\frac{1}{(iz - \omega)^2} - \frac{1}{\omega^2} \Big] = -\frac{1}{z^2} + \sum_{\substack{\omega \in L \\ \omega \neq 0}} \Big[\frac{1}{(iz - i\omega)^2} - \frac{1}{(i\omega)^2} \Big] =$$
$$= -\frac{1}{z^2} - \sum_{\substack{\omega \in L \\ \omega \neq 0}} \Big[\frac{1}{(z - \omega)^2} - \frac{1}{\omega^2} \Big] = -\wp(z)$$

och

$$\wp'(iz) = -2 \sum_{\omega \in L} \frac{1}{(iz - \omega)^3} = -2 \sum_{\omega \in L} \frac{1}{(iz - i\omega)^3} = -2 \sum_{\omega \in L} \frac{i}{(z - \omega)^3} = i\wp'(z)$$

så det finns en endomorfism $\phi : C(\mathbb{C}) \to C(\mathbb{C})$ definierad enligt

$$[x, y, 1] \mapsto [-x, iy, 1]$$

Från endomorfismen ϕ så kan endomorfismerna ϕ^2 och ϕ^3 bildas och dessa kan skrivas $[x, y, 1] \mapsto [x, -y, 1]$ respektive $[x, y, 1] \mapsto [-x, -iy, 1]$. Endomorfismen ϕ^4 är identitetsavbildningen så ϕ har ordning 4 i endomorfiringen till C. Enligt sats 5.5 så är $\mathrm{End}(C)$ isomorf med en ordning i den kvadratiska kroppsutvidgningen $\mathbb{Q}(i)$ så

$$\mathrm{End}(C) = \mathbb{Z}[\phi] \cong \mathbb{Z}[i]$$

för denna kurva.

Exempel 5.2. Betrakta gittret

$$L = \mathbb{Z}[\rho], \qquad \text{där } \rho = e^{2\pi i/3} = \frac{-1+\sqrt{-3}}{2}$$

Samma resonemang som i föregående exempel kommer nu att användas. För L så gäller att $\rho L = L$ och eftersom

$$G_4(L) = \sum_{\substack{\omega\in L\\ \omega\neq 0}} \frac{1}{\omega^4} = \sum_{\substack{\omega\in \rho L\\ \omega\neq 0}} \frac{1}{\omega^4} = \sum_{\substack{\omega\in L\\ \omega\neq 0}} \frac{1}{(\rho\omega)^4} = \rho^2 \sum_{\substack{\omega\in L\\ \omega\neq 0}} \frac{1}{\omega^4} = \rho^2 G_4(L)$$

så är konstanten $g_2 = G_4(L) = 0$. Alltså är $\mathbb{C}/L$ isomorf med den elliptiska kurvan $v^2 = 4u^3 - g_3$. Genom att sätta

$$u = \sqrt[3]{\frac{g_3}{4}}x \qquad \text{och} \qquad v = \sqrt{g_3}y$$

så ses att denna kurva är birationellt ekvivalent med den elliptiska kurvan

$$C : y^2 = x^3 - 1$$

Eftersom $\rho L = L$ så följer att

$$z \mapsto \rho z$$

är en endomorfism på $\mathbb{C}/L$ och C har komplex multiplikation. Det följer också att

$$\wp(\rho z) = \frac{1}{(\rho z)^2} + \sum_{\substack{\omega\in L\\ \omega\neq 0}} \Big[\frac{1}{(\rho z-\omega)^2} - \frac{1}{\omega^2}\Big] = \frac{\rho}{z^2} + \sum_{\substack{\omega\in L\\ \omega\neq 0}} \Big[\frac{1}{(\rho z-\rho\omega)^2} - \frac{1}{(\rho\omega)^2}\Big] =$$
$$= \frac{\rho}{z^2} + \sum_{\substack{\omega\in L\\ \omega\neq 0}} \Big[\frac{\rho}{(z-\omega)^2} - \frac{\rho}{\omega^2}\Big] = \rho\,\wp(z)$$

och

$$\wp'(\rho z) = -2\sum_{\omega\in L} \frac{1}{(\rho z-\omega)^3} = -2\sum_{\omega\in L} \frac{1}{(\rho z-\rho\omega)^3} = -2\sum_{\omega\in L} \frac{1}{(z-\omega)^3} = \wp'(z)$$

så $\phi : C(\mathbb{C}) \to C(\mathbb{C})$ definierad enligt

$$[x, y, 1] \mapsto [\rho x, y, 1]$$

är en endomorfism på $C(\mathbb{C})$. Endomorfismen ϕ^2 blir $[x, y, 1] \mapsto [\rho^2 x, y, 1]$ och ϕ^3 blir identitetsavbildningen så ϕ har i detta fall ordning 3 i endomorfiringen till C. Enligt sats 5.5 så är $\mathrm{End}(C)$ isomorf med en ordning i den kvadratiska kroppsutvidgningen $\mathbb{Q}(\rho)$ så

$$\mathrm{End}(C) = \mathbb{Z}[\phi] \cong \mathbb{Z}[\rho]$$

för denna kurva.

6 Tillämpning av elliptiska kurvor för heltalsfaktorisering

Grupperna som genereras av punkterna på elliptiska kurvor över $\mathbb{Z}_p$ kan användas för faktorisering av positiva heltal med hjälp av en algoritm upptäckt av H. W. Lenstra, Jr. I detta avsnitt kommer denna algoritm att presenteras och även implementeras i form av ett datorprogram.

Följande sats av Helmut Hasse säger att ordningen av grupperna bestående av punkterna på en elliptisk kurva över $\mathbb{Z}_p$ befinner sig i ett speciellt intervall runt p. Detta är en viktig egenskap som används av Lenstras algoritm.

Sats 6.1 (**Hasse**). *Om C är en elliptisk kurva över en ändlig kropp $\mathbb{Z}_p$ så är*

$$|C(\mathbb{Z}_p)| = p + 1 - \epsilon_p, \quad \text{där } \epsilon_p \leq 2\sqrt{p} \tag{15}$$

Bevis. Se Silverman [2], sida 131. □

Hasses sats säger ingenting om hur ϵ_p är fördelat över intervallet $[-2\sqrt{p}, 2\sqrt{p}]$ för slumpvis valda kurvor men det finns en förmodan om detta gjord av Mikio Sato och John Tate.

Förmodan 6.1 (**Sato-Tate**). *Låt C vara en elliptisk kurva över $\mathbb{Q}$ och antag att C inte har komplex multiplikation. Då gäller för reduktion av C modulo primtal p att*

$$\lim_{N\to\infty} \frac{|\{p \leq N : a \leq \frac{\epsilon_p}{2\sqrt{p}} \leq b\}|}{|\{p \leq N\}|} = \int_a^b \frac{2}{\pi}\sqrt{1-x^2}\,dx, \qquad -1 \leq a \leq b \leq 1$$

Det är endast i undantagsfall som en elliptisk kurva har komplex multiplikation så vad som gäller för dessa fall kommer inte att beskrivas här.

För att kunna använda Lenstras algoritm på ett effektivt sätt så måste elliptiska kurvor över ringar $\mathbb{Z}_n$, där n är sammansatt, kunna behandlas och därför görs följande två definitioner.

Definition 6.1. *Låt $n \in \mathbb{Z}, n > 3$ och låt C vara kurvan*

$$C : y^2 = x^3 + ax^2 + bx + c$$

där $a, b, c \in \mathbb{Z}_n$. Då definieras mängden av alla punkter på C över ringen $\mathbb{Z}_n$ som

$$C(\mathbb{Z}_n) = \{[X, Y, Z] \in \mathbb{P}^2(\mathbb{Z}_n) : Y^2Z = X^3 + aX^2Z + bXZ^2 + cZ^3\}$$

Mängden $C(\mathbb{Z}_n)$ bildar generellt sett ingen grupp men en kompositionsregel analog med definition 1.2 definieras ändå i de fall då detta är möjligt.

Definition 6.2. *Låt $n \in \mathbb{Z}, n > 3$ och låt*

$$C : y^2 = x^3 + bx + c$$

vara en kurva över $\mathbb{Z}_n$. *Låt* $P_1 = (x_1, y_1) \in C(\mathbb{Z}_n)$ *och* $P_2 = (x_2, y_2) \in C(\mathbb{Z}_n)$ *vara två punkter på* C. *Om* $P_1 \neq P_2$ *och inversen* $(x_2 - x_1)^{-1}$ *existerar i* $\mathbb{Z}_n$ *så definieras* $P_1 + P_2 = (x_3, y_3)$ *där*

$$x_3 = \lambda^2 - x_1 - x_2,\ y_3 = \lambda(x_1 - x_3) - y_1,\ \lambda = \frac{y_2 - y_1}{x_2 - x_1} \tag{16}$$

Om $P_1 = P_2$ *och* $2y_1$ *har en invers i* $\mathbb{Z}_n$ *så definieras* $P_1 + P_2 = 2P_1 = (x_3, y_3)$ *där*

$$x_3 = \lambda^2 - x_1 - x_2,\ y_3 = \lambda(x_1 - x_3) - y_1,\ \lambda = \frac{3x_1^2 + 2ax_1 + b}{2y_1} \tag{17}$$

Om $P_1 + P_2$ inte är definierat enligt förgående definition så kommer det att sägas vara odefinierat. Följande sats förklarar varför Lenstras algoritm fungerar. Satsen och beviset är baserat på förklaringen av Lenstras algoritm som ges i Silverman [1], sidorna 133-135.

Sats 6.2. *Låt* $n \in \mathbb{Z}, n > 3$ *vara sammansatt och* $\mathrm{SGD}(6, n) = 1$. *Antag att primtalet* $p > 3$ *delar* n. *Låt* C *vara kurvan*

$$C : y^2 = x^3 + bx + c$$

över $\mathbb{Q}$, *där* $b, c \in \mathbb{Z}$ *och* $\mathrm{SGD}(4b^3 + 27c^2, n) = 1$. *Antag att* $P = (x_1, y_1)$, *där* $x_1, y_1 \in \mathbb{Z}$, *är en punkt på* C. *Om* $\bar{C}$ *betecknar reduktion modulo* n *av* C, $\bar{P}$ *betecknar reduktion modulo* n *av* P *och* $k \in \mathbb{Z}$ *är sådant att* $|\tilde{C}(\mathbb{Z}_p)|$ *delar* k *så är* $k\bar{P}$ *odefinierat i* $\bar{C}(\mathbb{Z}_n)$.

Bevis. Notera först att $\tilde{C}$ är icke-singulär eftersom $\mathrm{SGD}(4b^3 + 27c^2, n) = 1$ medför att diskriminanten $4b^3 + 27c^2 \neq 0 \pmod{p}$. Om $|\tilde{C}(\mathbb{Z}_p)|$ delar k så har alla element i $\tilde{C}(\mathbb{Z}_p)$ en ordning som delar k. Då har $\tilde{P} \in \tilde{C}(\mathbb{Z}_p)$ en ordning som delar k så $\widetilde{kP} = k\tilde{P} = \tilde{\mathcal{O}}$ eftersom reduktion modulo p är en grupphomomorfism enligt sats 3.3. Antag att $k\bar{P}$ är definierat i $\bar{C}(\mathbb{Z}_n)$. Då är $k\bar{P} \neq \bar{\mathcal{O}}$, där $\bar{\mathcal{O}} = [0, 1, 0]$ är punkten i oändligheten på $\bar{C}$. Då följer att $k\tilde{P} \neq \tilde{\mathcal{O}}$, vilket är en motsägelse. Alltså är $k\bar{P}$ inte definierat i $\bar{C}(\mathbb{Z}_n)$. □

Notera att om $k\bar{P}$ i sats 6.2 inte är definierat så kommer ett försök till att beräkna $k\bar{P}$ med hjälp av formlerna (16) och (17) att misslyckas genom att $x_2 - x_1$ eller $2y_1$ saknar invers. Då är $\mathrm{SGD}(x_2 - x_1, n) > 1$ respektive $\mathrm{SGD}(2y_1, n) > 1$ så en faktor > 1 till n har hittats. Notera dock att det kan inträffa att $\mathrm{SGD}(x_2 - x_1, n) = n$ eller $\mathrm{SGD}(2y_1, n) = n$ så faktorn kan vara trivial. Risken att detta ska inträffa visar sig i praktiken vara liten.

För att kunna använda sats 6.2 för faktorisering så måste rätt kurva C och rätt k väljas, annars kommer antagligen ingen faktor att hittas. Eftersom p, naturligtvis, inte är känt så måste på något sätt gissningar av C och k göras till dess att $|\tilde{C}(\mathbb{Z}_p)|$ råkar dela k. Lenstras algoritm är alltså en så kallad probabilistisk algoritm. För att öka chansen att en faktor hittas så bör heltalet k väljas så att det är en produkt av små primtal. Om Sato-Tates förmodan antas vara sann så är det rimligt att anta att en kurva C, sådan att $|\tilde{C}(\mathbb{Z}_p)|$ också är en produkt av små primtal, hittas ganska fort om C väljs slumpmässigt. Lenstra [4] föreslår att k väljs som

$$k = \prod_{r=2}^{w} r^{e(r)} \tag{18}$$

där $e(r)$, för varje heltal $r \geq 2$, är det största heltal m sådant att

$$r^m \leq v + 2\sqrt{v} + 1$$

för några heltal v och w. Talet v kan ses som en övre begränsning för den faktor till n som söks och w påverkar den tid det kommer att ta att undersöka en kurva. Min handledare, Håkan Granath, föreslår att endast heltal r som är primtal tas med i (18). Skulle alla heltal r tas med i (18) så kommer k att innehålla onödigt många primtalsfaktorer av de flesta primtal $< w$ och då kommer beräkningen av $k\bar{P}$ att ta onödigt lång tid.

En kurva C med en punkt P kan slumpas fram genom att först slumpvis välja koefficienten b och koordinaterna x_0 och y_0 och sedan sätta $c = y_0^2 - x_0^3 - bx$.

En beskrivning av Lenstras algoritm i form av pseudokod finns algoritm 1. Fortfarande återstår en del detaljer som måste utredas för att kunna implementera algoritmen.

Algoritm 1 (Lenstra)

Förvillkor: Heltalet $n > 3$ som ska faktoriseras är sammansatt och sådant att $\text{SGD}(n, 6) = 1$ och sådant att n inte är på formen a^m för några heltal a och m, $m \geq 2$.

Eftervillkor: En icke-trivial faktor till n har hittats.

Välj slumpvis ett heltal b mellan 1 och n.
Välj slumpvis en punkt $P = (x_1, y_1)$, där x_1 och y_1 är heltal mellan 1 och n.
loop
  $c := y_1^2 - x_1^3 - bx_1$
  if $\text{SGD}(4b^3 + 27c^2, n) = n$ **then**
    $b := b + 1$
  else if $\text{SGD}(4b^3 + 27c^2, n) > 1$ **then**
    Returnera faktorn $\text{SGD}(4b^3 + 27c^2, n)$.
  else // $\text{SGD}(4b^3 + 27c^2, n) = 1$
    Bestäm ett lämpligt heltal k och ett lämpligt heltal h = antal kurvor att testa.
    Försök beräkna $k\bar{P}$ i $\bar{C}(\mathbb{Z}_n)$.
    if Beräkningen av $k\bar{P}$ lyckades eller gav den triviala faktorn n **then**
      if h kurvor har testats **then**
        Välj slumpvis ett heltal b mellan 1 och n.
        Välj slumpvis en punkt $P = (x_1, y_1)$, där x_1 och y_1 är mellan 1 och n.
        Öka värdet på k.
      else
        $b := b + 1$
      end if
    else // Beräkningen av $k\bar{P}$ gav en icke-trivial faktor.
      Returnera faktorn.
    end if
  end if
end loop

Lenstras algoritm kräver att $k\bar{P}$ beräknas (om det är möjligt). Att göra detta genom att beräkna summan $\bar{P} + \bar{P} + \bar{P} + \ldots$ kommer att ta orimligt lång tid. Ett effektivare sätt är att skriva k på formen

$$k = b_0 \cdot 2^0 + b_1 \cdot 2^1 + b_2 \cdot 2^2 + \cdots + b_m \cdot 2^m$$

och sedan beräkna $k\bar{P}$ som

$$k\bar{P} = b_0 \cdot 2^0\bar{P} + b_1 \cdot 2^1\bar{P} + b_2 \cdot 2^2\bar{P} + \cdots + b_m \cdot 2^m\bar{P}$$

där $2^r\bar{P}$ beräknas induktivt enligt

$$2^r\bar{P} = 2(2^{r-1}\bar{P}),\ 1 \leq r \leq m$$

På detta sätt krävs mindre än $2\log_2 k$ additioner för att beräkna $k\bar{P}$.

Nu följer en sammanfattning av Lentras resonemang ([4], sidorna 669-670) om vilka värden på parametrarna h och w som är de optimala. Följande sats av Lenstra [4] ger information om hur antalet kurvor h bör väljas.

Sats 6.3. *Låt n och v vara två heltal > 1 sådana att n har åtminstone två olika primtalsfaktorer > 3 och sådana att den minsta primtalsfaktorn $p > 3$ till n uppfyller $p \leq v$. Låt $w \in \mathbb{Z}$ vara > 1 och sådant att*

$$|\{s \in \mathbb{Z} : p + 1 - \sqrt{p} < s < p + 1 + \sqrt{p},\ \text{och varje primtal som delar } s \text{ är} \leq w\}| \geq 3$$

och låt vidare $f(w)$ vara sannolikheten att ett slumpvis valt heltal i intervallet $(p + 1 - \sqrt{p}, p + 1 + \sqrt{p})$ har alla sina primtalsfaktorer $\leq w$. Då är sannolikheten att Lenstras algoritm ska lyckas, för fixerade v och w i algoritmen, åtminstone

$$1 - \alpha^{-hf(w)/\log(v)}$$

för någon konstant α och där $h > 1$ är antalet kurvor som undersöks.

Bevis. Se Lenstra [4], sida 669. □

Med anledning av sats 6.3 så bör antalet kurvor h väljas i samma storleksordning som $\log(v)/f(w)$ för att få en rimlig chans att algoritmen ska lyckas.

Logaritmering av båda led i (18) ger

$$\log k = \log \prod_{r=2}^{w} r^{e(r)} \leq \log \prod_{r=2}^{w} (v + 2\sqrt{v} + 1) \leq (w - 1)\log(v + 2\sqrt{v} + 1)$$

så $\log k = O(w \log v)$. Låt $M(n)$ vara en övre begränsning för tidsåtgången att utföra en addition enligt definition 6.2. Då är tidsåtgången för att undersöka en kurva $O(w\log(v)M(n))$ eftersom det krävs som mest $2\log_2 k$ additioner för att beräkna $k\bar{P}$. Tidsåtgången för att undersöka h kurvor blir då

$$O(hw\log(v)M(n))$$

Om h väljs till $h = \log(v)/f(w)$ så bör det optimala alltså vara att välja w så att $w/f(w)$ blir så litet som möjligt. För att kunna välja ett sådant w så behövs någon form av information om $f(w)$.

Sats 6.4 (**Canfield, Erdös, Pomerance**). *Låt $x \in \mathbb{R}, x > e$ och definiera funktionen*

$$L(x) = e^{\sqrt{\log(x)\log(\log(x))}}$$

Låt a vara något positivt reellt tal. Sannolikheten att ett slumpvis valt positivt heltal $s \leq x$ har alla sina primtalsfaktorer $\leq L(x)^a$ är då

$$L(x)^{-\frac{1}{2a}+o(1)}$$

när $x \to \infty$.

Föregående sats kan användas till att uppskatta det optimala värdet på w om följande förmodan av Lenstra [4] antas vara sann.

Förmodan 6.2. *Sats 6.4 är giltig också om s är ett heltal i intervallet*

$$(x + 1 - \sqrt{x}, x + 1 + \sqrt{x})$$

Sätt $w = L(p)^a$, $x = p$ och använd notationen från sats 6.3. Då ger förmodan 6.2 att

$$f(w) = L(p)^{-\frac{1}{2a}+o(1)}, \text{ då } p \to \infty$$

och då är

$$\frac{w}{f(w)} = L(p)^{\frac{1}{2a}+a+o(1)}, \text{ då } p \to \infty$$

Eftersom $1/(2a) + a$ antar sitt minsta värde då $a = 1/\sqrt{2}$ så gäller för det optimala värdet på w att

$$w = L(p)^{\frac{1}{\sqrt{2}}+o(1)}, \text{ då } p \to \infty$$

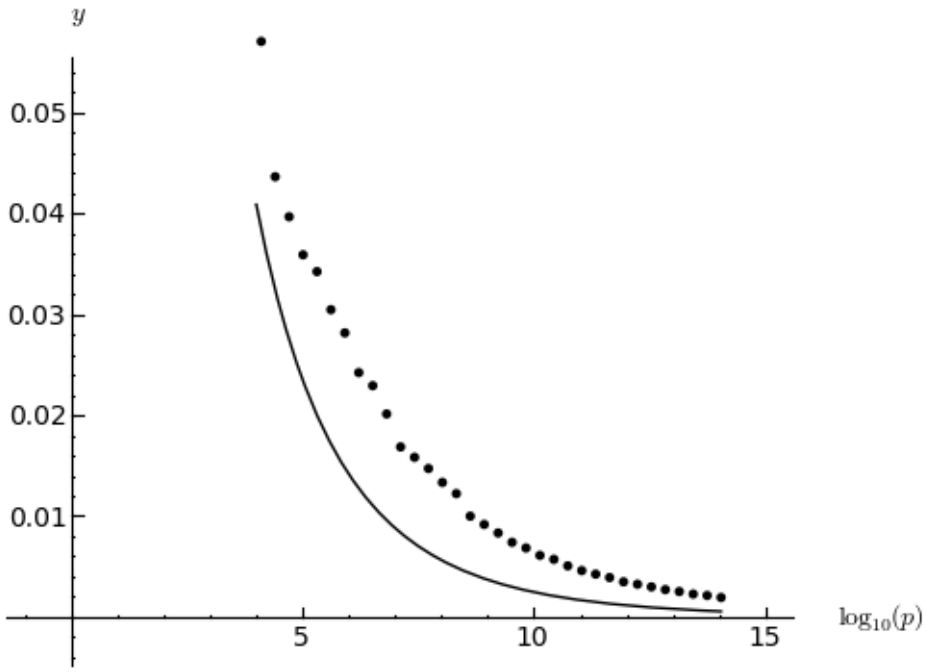
Det är möjligt att bestämma det exakta värdet på $f(w)$ för små tal p med hjälp av en dator så det går att undersöka hur väl approximationen

$$f(w) \approx L(p)^{-\frac{1}{2a}}$$

fungerar för relativt små p. I figur 1 finns en plot av $f(w)$ och $L(p)^{1/(2a)}$ som funktion av p, där $w = L(p)^a$ och $a = 1/\sqrt{2}$. Det bör alltså fungera bra att välja $w = L(p)^{1/\sqrt{2}}$ och $f(w) = L(p)^{-1/\sqrt{2}}$ i en implementation av Lenstras algoritm.

En implementation i C/C++ av Lenstras algoritm, som den har presenterats här med optimeringen av (18), finns i filerna "lenstra.h" och "lenstra.cpp", se bilaga A. I utskriften av filen "lenstra.h" så har vektorn "primtal" kortats ner av utrymmesskäl. Denna vektor har skapats med hjälp av Sage[1] och innehåller i verkligheten alla primtal från 2 till och med 15485863. Programmet använder sig av biblioteken GMP[2] och MPFR[3] för att kunna utföra beräkningar på "godtyckligt" stora heltal och flyttal. Programmet har ett textbaserat gränssnitt och när det körs så frågas efter talet att faktorisera samt en övre begränsning av den faktor som söks. Den övre begränsningen motsvarar v i beskrivningen ovan. Om ingen övre begränsning anges

[1] En fri och öppen matematisk programvara, se http://www.sagemath.org
[2] GNU Multiple Precision Arithmetic Library, se http://gmplib.org
[3] Se http://www.mpfr.org



Figur 1: Plot av $y = f(w)$ (punkter) och $y = L(p)^{1/(2a)}$ (heldragen kurva) som funktion av p, där $w = L(p)^a$ och $a = 1/\sqrt{2}$.

så väljer programmet $v = \sqrt{n}$, där n är heltalet som ska faktoriseras. Rad 16 i algoritm 1 utförs genom att sätta v till v^2.

Notera att programmet inte utför någon primtalsfaktorisering eftersom det saknas funktionalitet för att säkert avgöra om ett tal är ett primtal. Programmet söker istället efter en icke-trivial faktor till n och avslutas sedan. Eftersom det är onödigt att använda Lenstras algoritm om n har små primtalsfaktorer så undersöker programmet först om n är delbart med något primtal < 100. Innan faktorisering påbörjas så försöker programmet också bevisa att n är ett sammansatt heltal med hjälp av Fermats lilla sats tillämpad på alla primtal < 100. Om detta bevis misslyckas så varnas användaren för att det inmatade talet eventuellt kan vara ett primtal.

Det 60-siffriga talet

$$2^{199} - 1 = 803469022129495137770981046170581301261101496891396417650687$$

är en produkt av två primtal. Faktorisering av detta tal med hjälp av programmet ger följande resultat

```
Faktorisera:
803469022129495137770981046170581301261101496891396417650687

Ange en övre begränsning av den faktor som söks (eller 0 om detta är okänt):
200000000000

803469022129495137770981046170581301261101496891396417650687 =
164504919713 * 488416409388394117766004909858632430297754360079 9
```

och programkörningen tog cirka 30 sekunder på den aktuella datorn.

I tabell 1 finns en jämförelse av tidsåtgången för Lenstras algoritm och den triviala algoritmen vid faktorisering av några tal som är produkten av två nästan lika stora primtal. Med den triviala algoritmen så menas att undersöka i tur och ordning, genom division, om $2, 3, 4, \ldots$ är faktorer till talet som ska faktoriseras. Implementationen av den triviala algoritmen som använts finns i filen "trivial.cpp", se bilaga A. Vid körning av programmet som implementerar Lenstras algoritm så har det angetts att faktorernas storlek är okänd. Den redovisade tiden för Lenstras algoritm är ett medelvärde av 100 programkörningar eftersom tiden varierar mycket mellan olika programkörningar. Tider angivna som "$\geq$" i tabellen är uppskattningar. Programmet kördes under det Linux-baserade operativsystemet Ubuntu 8.10 och datorn som användes var utrustad med en AMD Athlon XP 2800+ processor och 1 GB RAM.

Tabell 1: Jämförelse av tidsåtgången för Lenstras algoritm och den triviala algoritmen.

Tal	Lenstras algoritm	Trivial algoritm
$10000019 \cdot 10000079$	0.06 s	4.5 s
$100000007 \cdot 100000037$	0.1 s	45 s
$1000000007 \cdot 1000000009$	0.49 s	7.5 min
$10000000019 \cdot 10000000033$	2.0 s	$\geq$ 75 min
$100000000003 \cdot 100000000019$	3.9 s	$\geq$ 13 h
$1000000000039 \cdot 1000000000061$	11 s	$\geq$ 5.2 dygn
$10000000000037 \cdot 10000000000051$	21 s	$\geq$ 7.4 veckor

A Källkod

lenstra.h

```
/*****************************************************************************/
/* Faktorisering med Lenstras algoritm.                                      */
/*                                                                           */
/* Johan Jonsson, 090603                                                     */
/*****************************************************************************/

#ifndef LENSTRA_H
#define LENSTRA_H

#include <gmp.h>
#include <gmpxx.h>
#include <mpfr.h>
#include <iostream>
#include <ctime>
#include <cstdlib>
#include <climits>

enum Resultat {ODEFINIERAT, SAMMANSATT, OKANT, PRIMTAL,
               BERAKNING_LYCKADES, FAKTOR_HITTAD, GCD_LIKA_MED_N};

unsigned long int primtal[] = {2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61,
67,71,73,79,83,89,97, ... ,15485837,15485843,15485849,15485857,15485863};

unsigned long int STORSTA_PRIMTAL = 15485863; //Det största primtalet i "primtal".
unsigned long int SMA_PRIMTAL = 25; //Antal primtal som anses vara små.

#endif
```

lenstra.cpp

```
/*****************************************************************************/
/* Faktorisering med Lenstras algoritm.                                      */
/*                                                                           */
/* Johan Jonsson, 090603                                                     */
/*****************************************************************************/

#include "lenstra.h"

bool utforlig = 0;

/*****************************************************************************/
/*                w                                                          */
/*               ---                                                         */
/* Beräknar k = | | r^(e(r)),                                                */
/*              r=2                                                          */
/*                                                                           */
/* där r är primtal och e(r) är det största heltal m s.a. r^m <= v+2*sqrt(v)+1. */
/*                                                                           */
/* Förvillkor:   w >= 2, v >= 0, w < STORSTA_PRIMTAL                         */
/* Eftervillkor: resultat = k.                                               */
/*****************************************************************************/
mpz_class berakna_k(mpz_class w, mpz_class v)
{
    // GMP saknar log-funktion så MPFR används istället.

    mpfr_t sqrt_v;
    mpfr_init_set_z(sqrt_v, v.get_mpz_t(), GMP_RNDU);
    mpfr_sqrt(sqrt_v, sqrt_v, GMP_RNDU);

    mpfr_t max;
    mpfr_init_set(max, sqrt_v, GMP_RNDU);
    mpfr_mul_ui(max, max, 2, GMP_RNDU);
    mpfr_add_z(max, max, v.get_mpz_t(), GMP_RNDU);
    mpfr_add_ui(max, max, 1, GMP_RNDU);
```

```
    mpfr_t log_max;
    mpfr_init_set(log_max, max, GMP_RNDU);
    mpfr_log(log_max, log_max, GMP_RNDU);

    mpz_class k(1);
    mpz_class r(primtal[0]);
    for(unsigned int i = 0; r <= w; r = primtal[++i])
    {
        mpfr_t log_r;
        mpfr_init_set_z(log_r, r.get_mpz_t(), GMP_RNDD);
        mpfr_log(log_r, log_r, GMP_RNDD);

        mpfr_t log_r_max;
        mpfr_init(log_r_max);
        mpfr_div(log_r_max, log_max, log_r, GMP_RNDU);

        unsigned long m = mpfr_get_ui(log_r_max, GMP_RNDD);

        mpz_class r_pow_m;
        mpz_pow_ui(r_pow_m.get_mpz_t(), r.get_mpz_t(), m);

        k *= r_pow_m;

        mpfr_clear(log_r);
        mpfr_clear(log_r_max);
    }

    mpfr_clear(log_max);
    mpfr_clear(max);
    mpfr_clear(sqrt_v);

    return k;
}

/*****************************************************************************/
/* Beräknar "optimala" värden för w och h.                                   */
/* Förvillkor:   v >= 3                                                      */
/* Eftervillkor: w = L(v)^(1/sqrt(2)) och h = log(v)/f(w).                   */
/*****************************************************************************/
void optimala_parametrar(mpz_class v, mpz_class &w, mpz_class &h)
{
    // GMP saknar log-funktion så MPFR används istället.

    mpfr_t log_x;
    mpfr_init_set_z(log_x, v.get_mpz_t(), GMP_RNDN);
    mpfr_log(log_x, log_x, GMP_RNDN);

    mpfr_t log_log_x;
    mpfr_init_set(log_log_x, log_x, GMP_RNDN);
    mpfr_log(log_log_x, log_x, GMP_RNDN);

    mpfr_t sqrt;
    mpfr_init(sqrt);
    mpfr_mul(sqrt, log_x, log_log_x, GMP_RNDN);
    mpfr_sqrt(sqrt, sqrt, GMP_RNDN);

    mpfr_t L;
    mpfr_init(L);
    mpfr_exp(L, sqrt, GMP_RNDN);

    mpfr_t sqrt_2;
    mpfr_init(sqrt_2);
    mpfr_sqrt_ui(sqrt_2, 2, GMP_RNDN);

    mpfr_t sqrt_2_inv;
    mpfr_init(sqrt_2_inv);
    mpfr_ui_div(sqrt_2_inv, 1, sqrt_2, GMP_RNDN);

    mpfr_t sqrt_2_inv_minus_sqrt_2;
    mpfr_init(sqrt_2_inv_minus_sqrt_2);
```

```
    mpfr_sub(sqrt_2_inv_minus_sqrt_2, sqrt_2_inv, sqrt_2, GMP_RNDN);

    mpfr_t w_float;
    mpfr_init(w_float);
    mpfr_pow(w_float, L, sqrt_2_inv, GMP_RNDN);

    mpfr_t f_w;
    mpfr_init(f_w);
    mpfr_pow(f_w, L, sqrt_2_inv_minus_sqrt_2, GMP_RNDN);

    mpfr_t log_v;
    mpfr_init_set_z(log_v, v.get_mpz_t(), GMP_RNDN);
    mpfr_log(log_v, log_v, GMP_RNDN);

    mpfr_t h_float;
    mpfr_init(h_float);
    mpfr_div(h_float, log_v, f_w, GMP_RNDN);

    mpfr_get_z(w.get_mpz_t(), w_float, GMP_RNDN);
    mpfr_get_z(h.get_mpz_t(), h_float, GMP_RNDN);
}

/*****************************************************************************/
/* Multiplicera en punkt med 2.                                              */
/* Förvillkor:   gcd(2y, n) = 1                                              */
/* Eftervillkor: (x,y) = 2(x,y) (mod n)                                      */
/*****************************************************************************/
void dubbla_punkt_mod_n(mpz_class &x, mpz_class &y, mpz_class b, mpz_class n)
{
    mpz_class fprim_x = (3*x*x+b) % n;

    mpz_class tva_y = 2*y;
    mpz_class tva_y_inv;
    mpz_invert(tva_y_inv.get_mpz_t(), tva_y.get_mpz_t(), n.get_mpz_t());

    mpz_class lambda = (fprim_x * tva_y_inv) % n;

    mpz_class x_gammal(x);

    x = (lambda*lambda - 2*x) % n;
    y = (-lambda * x - (y - lambda * x_gammal)) % n;

    mpz_mod(x.get_mpz_t(), x.get_mpz_t(), n.get_mpz_t());
    mpz_mod(y.get_mpz_t(), y.get_mpz_t(), n.get_mpz_t());
}

/*****************************************************************************/
/* Addera två punkter.                                                       */
/* Förvillkor:   gcd(x2 - x1, n) = 1                                         */
/* Eftervillkor: (x1,y1) = (x1,y1) + (x2,y2) (mod n)                         */
/*****************************************************************************/
void addera_punkter_mod_n(mpz_class &x1, mpz_class &y1,
                          mpz_class &x2, mpz_class &y2,
                          mpz_class b, mpz_class n)
{
    mpz_class x2_minus_x1 = x2 - x1;
    mpz_class x2_minus_x1_inv;
    mpz_invert(x2_minus_x1_inv.get_mpz_t(), x2_minus_x1.get_mpz_t(),
               n.get_mpz_t());

    mpz_class lambda = ((y2 - y1) * x2_minus_x1_inv) % n;

    x1 = (lambda*lambda - x1 - x2) % n;
    y1 = (-lambda * x1 - (y2 - lambda * x2)) % n;

    mpz_mod(x1.get_mpz_t(), x1.get_mpz_t(), n.get_mpz_t());
    mpz_mod(y1.get_mpz_t(), y1.get_mpz_t(), n.get_mpz_t());
}
```

```
/*****************************************************************************/
/* Försöker beräkna kP (mod n)                                               */
/* Förvillkor:   k > 0                                                       */
/* Eftervillkor: resultat = FAKTOR_HITTAD, GCD_LIKA_MED_N eller              */
/*               BERAKNING_LYCKADES                                          */
/*****************************************************************************/
Resultat berakna_kP_mod_n(mpz_class x1, mpz_class y1, mpz_class b, mpz_class k,
                          mpz_class n, mpz_class &faktor)
{
    mpz_class kP_x, kP_y;
    mpz_class gcd;

    unsigned long int forsta_1 = mpz_scan1(k.get_mpz_t(), 0);
    for(int i = 0; i < forsta_1; i++)
    {
        dubbla_punkt_mod_n(x1, y1, b, n);
    }

    kP_x = x1;
    kP_y = y1;

    size_t size = mpz_sizeinbase(k.get_mpz_t(), 2);
    for(int i = forsta_1+1; i < size; i++)
    {
        mpz_gcd(gcd.get_mpz_t(), y1.get_mpz_t(), n.get_mpz_t());
        if(gcd > 1)
        {
            if(gcd < n)
            {
                faktor = gcd;
                return FAKTOR_HITTAD;
            }
            else
            {
                return GCD_LIKA_MED_N;
            }
        }

        dubbla_punkt_mod_n(x1, y1, b, n);

        if(mpz_tstbit(k.get_mpz_t(), i))
        {
            mpz_class kP_x_minus_x1 = kP_x - x1;
            mpz_gcd(gcd.get_mpz_t(), kP_x_minus_x1.get_mpz_t(), n.get_mpz_t());
            if(gcd > 1)
            {
                if(gcd < n)
                {
                    faktor = gcd;
                    return FAKTOR_HITTAD;
                }
                else
                {
                    return GCD_LIKA_MED_N;
                }
            }
            addera_punkter_mod_n(kP_x, kP_y, x1, y1, b, n);
        }
    }

    return BERAKNING_LYCKADES;
}

/*****************************************************************************/
/* Försöker avgöra om ett tal är sammansatt eller inte.                      */
/* Förvillkor:   n > 3                                                       */
/* Eftervillkor: resultat = OKANT, PRIMTAL, SAMMANSATT eller FAKTOR_HITTAD.  */
/*               OKANT betyder att det är okänt om "n" är ett primtal eller inte.*/
/*               En hittad faktorer returneras i "faktor".                   */
/*****************************************************************************/
```

```
Resultat primtals_test(mpz_class n, mpz_class &faktor)
{
    mpz_class bas;
    mpz_class exponent = n-1;
    mpz_class rest;

    for(int index = 0; index < SMA_PRIMTAL; index++)
    {
        if(n == primtal[index]) return PRIMTAL;

        if(mpz_mod_ui(rest.get_mpz_t(), n.get_mpz_t(), primtal[index]) == 0)
        {
            faktor = primtal[index];
            return FAKTOR_HITTAD;
        }
    }

    for(int index = 0; index < SMA_PRIMTAL; index++)
    {
        bas = primtal[index];
        mpz_powm(rest.get_mpz_t(), bas.get_mpz_t(), exponent.get_mpz_t(),
                 n.get_mpz_t());

        if(rest != 1) return SAMMANSATT;
    }

    return OKANT;
}

/*****************************************************************************/
/* Kontrollerar om ett tal är på formen a^m för något m>= 2                  */
/* Förvillkor:   n > 1                                                       */
/* Eftervillkor: resultat = sant om n är på formen a^m, annars falskt.       */
/*               Om n är på formen a^m så finns en icke-trivial faktor i     */
/*               "faktor".                                                   */
/*****************************************************************************/
bool perfect_power(mpz_class n, mpz_class &faktor)
{
    mpz_class rot;
    mpz_class rest;
    unsigned long int i = 2;

    do
    {
        mpz_rootrem(rot.get_mpz_t(), rest.get_mpz_t(), n.get_mpz_t(), i);

        if(rest == 0)
        {
            if(utforlig) std::cout << "Perfect power\n";

            faktor = rot;
            return true;
        }

        i++;
    } while(rot > 1);

    return false;
}

/*****************************************************************************/
/* (Försöker) faktorisera med Lenstras algorithm                             */
/* Förvillkor:   n > 3 och sgb(n,6) = 1, v_start >= 3 eller v_start = 0      */
/*               Om v_start = 0 så väljer funktionen själv ett v_start.      */
/* Eftervillkor: resultat = en icke-trivial faktor till n. Om n är ett primtal */
/*               så returnerar inte funktionen.                              */
/*****************************************************************************/
mpz_class faktorisera(mpz_class n, mpz_class v_start)
{
```

```
gmp_randstate_t state;
gmp_randinit_default(state);
gmp_randseed_ui(state, time(NULL));

mpz_class v;
if(v_start == 0)
    mpz_sqrt(v.get_mpz_t(), n.get_mpz_t());
else
    v = v_start;
if(utforlig) std::cout << "Väljer v = " << v;

mpz_class w;
mpz_class h;
optimala_parametrar(v, w, h);
if(utforlig) std::cout << ", w = " << w << ", h = " << h << ".\n";

if(w >= STORSTA_PRIMTAL)
{
    std::cout <<
        "Varning, tillräckligt stora primtal för faktorisering saknas!\n";
    w = STORSTA_PRIMTAL - 1;
    if(utforlig) std::cout << "Sätter w = " << w << ".\n";
}

mpz_class k = berakna_k(w, v);
if(utforlig) std::cout << "Antal siffror i k: "
                       << mpz_sizeinbase(k.get_mpz_t(), 10) << ".\n";

mpz_class b(1);
mpz_class n_minus_1 = n-1;
mpz_urandomm(b.get_mpz_t(), state, n_minus_1.get_mpz_t());
b++;

mpz_class x1, y1;
mpz_urandomm(x1.get_mpz_t(), state, n_minus_1.get_mpz_t());
x1++;
mpz_urandomm(y1.get_mpz_t(), state, n_minus_1.get_mpz_t());
y1++;

mpz_class c;
mpz_class D;
mpz_class gcd;
mpz_class faktor;

mpz_class antal_testade_kurvor(0);

Resultat resultat = ODEFINIERAT;
while(resultat != FAKTOR_HITTAD)
{
    c = y1*y1 - x1*x1*x1 - b*x1;
    D = 4*b*b*b + 27*c*c;
    mpz_gcd(gcd.get_mpz_t(), D.get_mpz_t(), n.get_mpz_t());

    if(gcd > 1)
    {
        if(gcd == n)
        {
            b++;
            continue;
        }
        else
        {
            faktor = gcd;
            break;
        }
    }

    resultat = berakna_kP_mod_n(x1, y1, b, k, n, faktor);
    antal_testade_kurvor++;
```

```
        if(utforlig) std::cout << antal_testade_kurvor
                               << " kurvor undersökta\n";

        if(resultat == BERAKNING_LYCKADES || resultat == GCD_LIKA_MED_N)
        {
            if(antal_testade_kurvor >= h)
            {
                mpz_urandomm(b.get_mpz_t(), state, n_minus_1.get_mpz_t());
                b++;

                mpz_urandomm(x1.get_mpz_t(), state, n_minus_1.get_mpz_t());
                x1++;
                mpz_urandomm(y1.get_mpz_t(), state, n_minus_1.get_mpz_t());
                y1++;

                v *= v;
                optimala_parametrar(v, w, h);
                k = berakna_k(w, v);
                antal_testade_kurvor = 0;

                if(utforlig)
                    std::cout << "Väljer v = " << v << ", w = " << w
                              << ", h = " << h << ".\n"
                              << "Antal siffror i k: "
                              << mpz_sizeinbase(k.get_mpz_t(), 10) << ".\n";
            }
            else
            {
                b++;
            }
        }
    }

    if(utforlig)
        std::cout << "Hittade faktorn " << faktor << " då b = " << b << ", c = "
                  << c << ", x1 = " << x1 << ", y1 = " << y1 << ".\n";

    gmp_randclear(state);

    return faktor;
}

/**************************************************************************/
/* main                                                                   */
/**************************************************************************/
int main(int argc, char *argv[])
{
    utforlig = 0;
    for(int i = 1; i < argc; i++)
    {
        if(strcmp(argv[i], "-u") == 0) utforlig = 1;
    }

    mpz_class n;
    bool fel;
    do
    {
        fel = false;
        std::cout << "Faktorisera: ";
        std::cin >> n;

        if(!std::cin || n <= 3)
        {
            if(!std::cin)
            {
                std::cin.clear();
                std::cin.ignore(INT_MAX, '\n');
            }
            fel = true;
            std::cout << "Var god att ange ett heltal > 3.\n\n";
```

```
        }
    } while(fel);

    mpz_class v_start;
    do
    {
        fel = false;
        std::cout << "Ange en övre begränsning till den faktor som söks "
                  << "(eller 0 om detta är okänt): ";
        std::cin >> v_start;

        if(!std::cin || v_start < 0 || v_start == 1 || v_start == 2)
        {
            if(!std::cin)
            {
                std::cin.clear();
                std::cin.ignore(INT_MAX, '\n');
            }
            fel = true;
            std::cout << "Var god att ange ett heltal > 2, eller 0.\n\n";
        }
    } while(fel);

    mpz_class faktor;
    Resultat resultat = primtals_test(n, faktor);

    if(resultat == PRIMTAL)
    {
        std::cout << n << " är ett primtal.\n";
        return 0;
    }

    if(resultat == OKANT)
    {
        std::cout << "Kunde inte avgöra om " << n
        << " är sammansatt eller inte, programmet kanske inte avslutas!\n";
    }

    if(resultat == SAMMANSATT || resultat == OKANT)
    {
        if(!perfect_power(n, faktor))
        {
            faktor = faktorisera(n, v_start);
        }
    }

    std::cout << n << " = " << faktor << " * " << (n/faktor) << std::endl;

    return 0;
}
```

trivial.cpp

```
/*****************************************************************************/
/* Faktorisering med den triviala algoritmen.                                */
/*                                                                           */
/* Johan Jonsson, 090430                                                     */
/*****************************************************************************/

#include <gmp.h>
#include <gmpxx.h>
#include <iostream>
#include <cstdlib>
#include <climits>

/*****************************************************************************/
/* Faktoriserar med den triviala algoritmen.                                 */
/* Förvillkor:   n > 1                                                       */
/* Eftervillkor: resultat = Den minsta faktorn > 1 till n.                   */
/*****************************************************************************/
mpz_class faktorisera(mpz_class n)
{
    mpz_class faktor(2);

    while(faktor <= n)
    {
        if(n % faktor == 0) return faktor;
        faktor++;
    }
}

/*****************************************************************************/
/* main                                                                      */
/*****************************************************************************/
int main(int argc, char *argv[])
{
    mpz_class n;
    bool fel;
    do
    {
        fel = false;
        std::cout << "Faktorisera: ";
        std::cin >> n;

        if(!std::cin || n <= 1)
        {
            if(!std::cin)
            {
                std::cin.clear();
                std::cin.ignore(INT_MAX, '\n');
            }
            fel = true;
            std::cout << "Var god att ange ett heltal > 1.\n\n";
        }
    } while(fel);

    mpz_class faktor;
    faktor = faktorisera(n);

    std::cout << n << " = " << faktor << " * " << (n/faktor) << std::endl;

    return 0;
}
```

Referenser

[1] Joseph H. Silverman, John Tate, *Rational Points On Elliptic Curves*, Springer, USA, 1992.

[2] Joseph H. Silverman, *The Arithmetic of Elliptic Curves*, Springer, New York, 1986.

[3] Susanne Schmitt, *Elliptic Curves: A Computational Approach*, Walter de Gruyter, Berlin, 2003.

[4] H. W. Lenstra (Jr), *Factoring Integers with Elliptic Curves*, The Annals of Mathematics, Second Series, Vol. 126, No. 3 (Nov. 1987), pp. 649-673.