ŠIAULIŲ UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS

INFORMATIKOS KATEDRA

Jurgita Pocienė

Informatikos magistro, mokytojo specialybės II kurso (neakivaizdinio skyriaus) studentė

Elipsinių kreivių taškų skaičiavimo algoritmai ir jų taikymai

MAGISTRO DARBAS

Darbo vadovė:
dr. R. Steuding

Recenzentas:
doc. V. Sirius

Šiauliai, 2005/2006 m.m.

Turinys

8. Priedai:

   8.1 Vartotojo vadovas

   8.2 Programuotojo vadovas

   8.3 Testavimo protokolas

   8.4 CD laikmenos turinys ir kiti nurodymai

Iliustracijų sąrašas

1. Įvadas

Duomenų šifravimas jau buvo naudojamas Julijaus Cezario laikais. Norėdamas nusiųsti žinutę savo generolams ir nepasitikėdamas savo pasiuntiniais, jis savo žinutėje pakeitė visas A raides į D, visas B į E ir t.t. Ir kiekvienas, kuris žinojo šį užkodavimo būdą galėjo perskaityti žinutę.

Tekstas, kurį galima skaityti nepanaudojus jokių specialių priemonių vadinamas paprastu tekstu. Metodas naudojantis paprastą tekstą ir paverčiantis jį į gausybę nieko bendro tarp savęs neturinčių simbolių vadinamas šifravimu. O toks tekstas - užšifruotu tekstu. Pati šifravimo schema atrodytų maždaug taip:

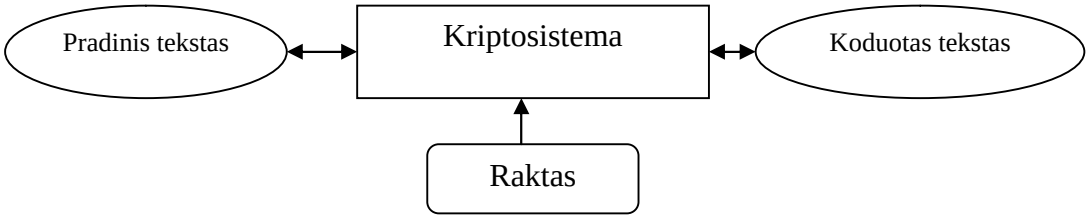
```
|--------|     -------->     |/////|          ---------->     |---------|
paprastas    šifravimas    užšifruotas    dešifravimas    paprastas
tekstas                    tekstas                        tekstas
```

Matematinius metodus, naudojamus duomenų šifravimui ir dešifravimui, nagrinėja kriptografija. Kriptografinės priemonės leidžia saugoti ypač slaptą informaciją ir siųsti ją nesaugiais tinklais (kaip internetas), taigi jos negali perskaityti niekas kitas, kaip tik gavėjas. Kriptografinių schemų saugumą nagrinėja kriptoanalizė. ***Paprasta kriptoanalizė***, tai loginės kombinacijos, matematinių įrankių panaudojimas, kantrybė, pasiryžimas ir žinoma - sėkmė.
Šiuolaikinė kriptografija apima tokias pagrindines sritis:

1. Simetrines kriptosistemas;
2. Kriptosistemas su atviru raktu;
3. Elektroninio parašo sistemas;
4. Valdymą raktais.

Raktas – duomenys, būtini duomenų užšifravimui ir dešifravimui. ***Kriptosistema*** – teksto transformavimo panaudojant raktą taisyklių rinkinys. ***Raktų aibė*** – visos galimos kriptosistemos raktų reikšmės. Simetrinėse kriptosistemose informacijai užšifruoti ir dešifruoti naudojamas vienas ir tas pats raktas. Kriptosistemose su atviru raktu naudojami du raktai viešasis (public) ir privatus (private), kurie susiję tam tikrais matematiniais sąryšiais.

Papildyta duomenų šifravimo schema atrodytų taip:



Yra nusistovėjusi nuomonė, kad žmogus šifruoja duomenis, todėl, kad daro kažką nelegalaus. Dauguma pasakys, kad jie neturi ką slėpti. Tačiau šiame informacijos amžiuje kiekviena gauta informacijos dalelytė gali būti panaudota prieš jus. Tai ypač aktualu konkurentams. Kam naudoti pinigus ir resursus, jei galima visą reikiamą informaciją gauti iš savo priešininkų. Šifravimas taip pat labai svarbi elektroninės komercijos dalis. Norint įgyti klientų pasitikėjimą turi būti užtikrintas jų duomenų saugumas.

Nuo viešojo rakto kriptografijos atsiradimo pradžios, buvo naudojamos dvi pagrindinės kriptosistemos: RSA ir El-Gamal. Jos dažniausiai naudojamos ir dabar. Šios sistemos gali būti naudojamos tiek šifravimui / dešifravimui, tiek elektroniniam parašui. Nemažai saugumo standartų sukurta šių sistemų pagrindu, tačiau šiuo metu vis dažniau taikomos kriptosistemos elipsinių kreivių pagrindu. Elipsinių kreivių kriptosistemų (EKK) saugumui užtikrinti naudojamos kreivės turi tenkinti tam tikrus reikalavimus. Generuojant tinkamas elipsines kreives reikalinga informacija apie jų taškų grupės eilę, tai vienas iš pagrindinių elipsinės kreivės parametrų, nusakančių kriptosistemos su elipsine kreive saugumą. Dėl šios priežasties elipsinių kreivių taškų skaičiavimo algoritmai vaidina svarbų vaidmenį šiuolaikinėje kriptografijoje.

Darbo tikslai:

- Išnagrinėti elipsinių kreivių taškų skaičiavimo algoritmus ir juos palyginti;
- Apžvelgti elipsinių kreivių taškų skaičiavimo algoritmų taikymą;
- Realizuoti Schoof‘o elipsinių kreivių taškų skaičiavimo algoritmą ir panagrinėti jo efektyvizacijos galimybes.

Darbą sudaro:

- Teorinė dalis;
- Projektinė dalis;
- Darbo eigos aprašymas;
- Išvados;
- Priedai:
  - Vartotojo vadovas (1 priedas);
  - Programuotojo vadovas (2 priedas);
  - Testavimo protokolas (3 priedas);
  - CD laikmenos turinys ir kiti nurodymai (4 priedas).

2. Teorinė dalis

2.1 Temos analizė

Elipsinių kreivių kriptografija (EKK) atsirado apie 1985 m. Jos pradininku laikomi Mileris (Miller) ir Koblitsas (Koblitz). Buvo labai daug diskutuojama apie EKK saugumą ir efektyvumą. Informacinės technologijos šiais laikais vystosi labai sparčiai. Tobulėjant technologijoms vis daugiau informacijos apdorojama ne personaliniais kompiuteriais, o mobiliaisiais įrenginiais (delniniai kompiuteriai, „protingieji“ telefonai). Taigi mums reikalinga kriptosistema su „mažais“ parametrais. EKK turi savybes, kurių mums reikia. Tai reiškia, kad aukštesnio lygio saugumui užtikrinti galima naudoti trumpesnius raktus.

2.1.1 Elipsinių kreivių kriptografijos standartai

Standartų vystymasis labai svarbus kriptosistemų panaudojimui, standartas padeda užtikrinti saugumą skirtingai pritaikant tą pačią kriptosistemą. Yra keletas didelių organizacijų, kurios kuria standartus:

- International Standards Organization (ISO),
- American National Standard Institute (ANSI),
- Institute of Electrical and Electronics Engineers (IEEE),
- Federal Information Processing Standards (FIPS).

Labiausiai paplitęs EKK algoritmas – ECDSA (Elliptic Curve Digital Signature Algorithm), ISO standartas buvo priimtas 1998 m., ANSI – 1999 m ., IEEE – 2000 m ir FIPS – 2003 m.

Saugiam duomenų perdavimui tarp kompiuterių yra naudojami tam tikri protokolai. Vienas labiausiai paplitusių – TLS/SSL (Transport Layer Security/Secure Socket Layer) protokolas, naudojantis EKK technologijas. Kitas svarbus protokolas – WAP/WTLS (Wireless Application Protocol), užtikrinantis saugų belaidį ryšį.

2.1.2 Elipsinių kreivių kriptosistemų privalumai

Kriptosistemos elipsinių kreivių pagrindu saugumui užtikrinti reikia žymiai trumpesnių raktų nei, pavyzdžiui, RSA ar DSA sistemoms. Elipsinių kreivių sistemoms užtenka „silpnesnės“ techninės įrangos (mažiau atminties), todėl tokios sistemos naudojamos delniniuose kompiuteriuose bei Smart Phone telefonuose.

Puslapyje http://www.racal.ru/rsp/eliptic_curve_cryptography.htm pateiktas RSA ir kriptosistemos su elipsine kreive modulių ilgių palyginimas:

Sistema su elipsine kreive EKK (bazinis taškas P)	RSA (modulio n ilgis)
106 bitų	512 bitų
1326 bitų	768 bitų
160 bitų	1024 bitų
224 bitų	2048 bitų

Viena didžiausių problemų naudojant kriptosistemas, paremtas elipsinėmis kreivėmis, yra ta, kad šios sistemos nėra dar iki galo išnagrinėtos, atskleistos. Nesenai atlikti tyrimai parodė, kad kai kurios kreivės su tam tikrais parametrais P yra netinkamos kriptosistemoms, nes turi pakankamai paprastus sprendimo būdus, tokios kreivės vadinamos anomalinėmis. Elipsinių kreivių tyrimai vis dar atliekami ir šiomis dienomis, todėl kriptosistemos, paremtos elipsinėmis kreivėmis, kol kas naudojamos gana atsargiai.

2.2 Darbo srities analizė

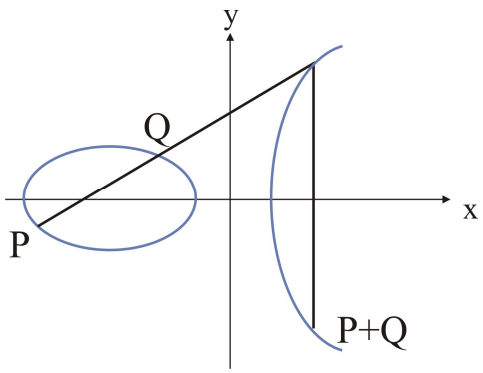
2.2.1 Elipsinės kreivės

Elipsinė kreivė E virš kūno **F** aprašoma lygtimi:

$Y^2 + a_1XY + a_3Y = X^3 + a_2X^2 + a_4X + a_5$, kur $a_i \in F$ ir kūne **F** neegzistuoja taškas, kuriame neegzistuoja nei viena dalinė išvestinė. Šios sąlygos galioja visiems kūnams, tačiau kriptografijoje mus domina tik baigtiniai kūnai, tiksliau kūnas $\mathbf{F}_p$ kur p – pirminis skaičius dėl jo struktūros ypatumų bei $F_{q^m}, q = p^r$, nes kai $p=2$, skaičiavimus lengva pritaikyti kompiuterinei technikai. $\mathbf{F}_p$ – kūnas virš pirminių skaičių lauko.

Kai $p>3$, elipsinės kreivės lygtis virš $\mathbf{F}_p$ įgyja paprastesnį pavidalą: $Y^2 = X^3 + aX + b$, kai $a,b \in F_p$ ir $-(4a^3 + 27b^2) \neq 0$, ši sąlyga užtikrina, kad funkcijos visose taškuose egzistuoja dalinės išvestinės.

Taigi, nagrinėsime lygtį $Y^2 = X^3 + aX + b \ (mod\ p),\ a,b \in F_p$.



2.2.1.1 pav. Operacijų su elipsinės kreivės taškais geometrinis atvaizdavimas

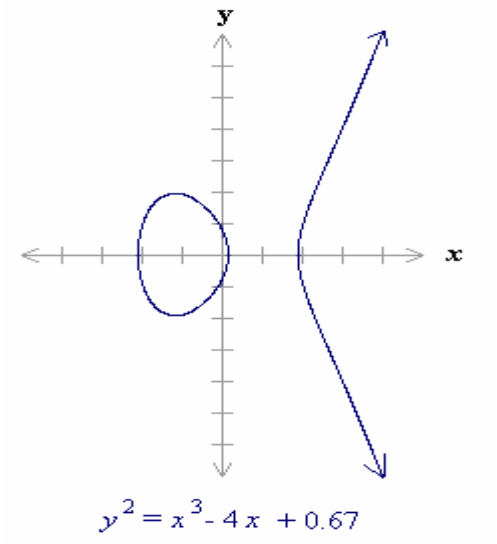
Tarkime turime du elipsinės kreivės taškus P ir Q, juos sujungiame tiese. Tiesė kerta elipsinę kreivę trečiame taške ir jei paimtume veidrodinį atspindį x ašies atžvilgiu tada gausime dar vieną elipsinės kreivės tašką P+Q. Tai leidžia tiksliai nusakyti elipsinės kreivės formą. Kitaip tariant elipsinės kreivės taškai ir taškas 0 (be galo nutolęs taškas) sudaro aibę su joje apibrėžtomis binarinėmis operacijomis (taškų suma, daugyba iš skaliaro).

2.2.2 Elipsinių kreivių taškų aritmetika virš baigtinių kūnų

Su elipsinių kreivių taškais galima atlikti keletą operacijų:

1. Atvirkštinio taško;
2. Taškų sudėtį;
3. Daugybą iš skaliaro.

Operacijos su elipsinių kreivių taškais gali būti apibrėžiamos geometriškai:



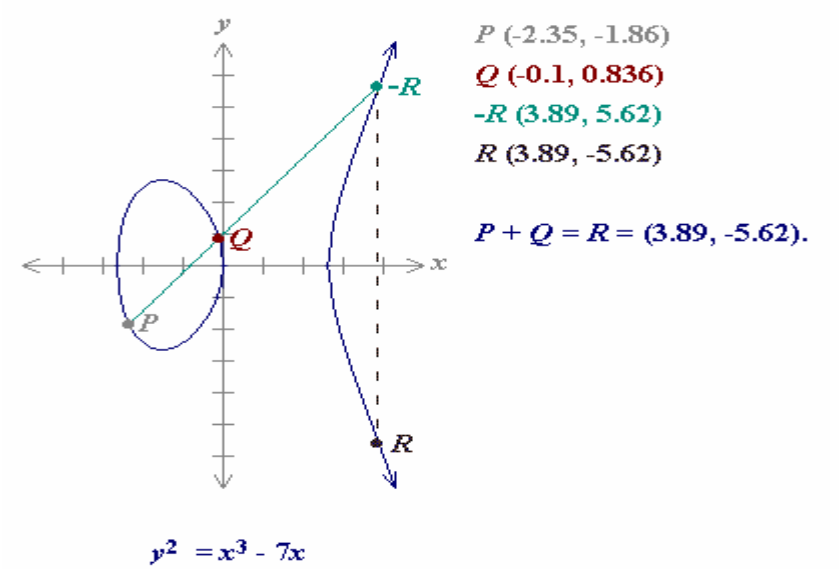
2.2.2-1pav. Elipsinė kreivė

1. Elipsinės kreivės atvirkštinis taškas.

Tarkime, turime tašką $P=(x_P, y_P)$. Tai jam atvirkštinis taškas –P yra taškas, simetriškas taškui P x ašies atžvilgiu $(x_P, -y_P)$.

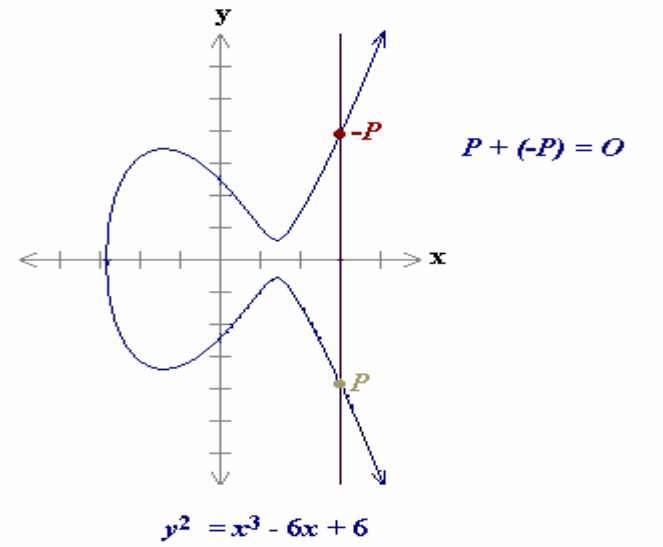
2. Elipsinių kreivių taškų sudėtis.

Per taškus P ir Q brėžiama tiesė, kuri kerta elipsinę kreivę trečiame taške –R. Taško –R atvirkštinis taškas R yra vadinamas P ir Q suma $R=P+Q$.



2.2.2-2pav. Elipsinės kreivės taškų sudėties geometrinis atvaizdavimas

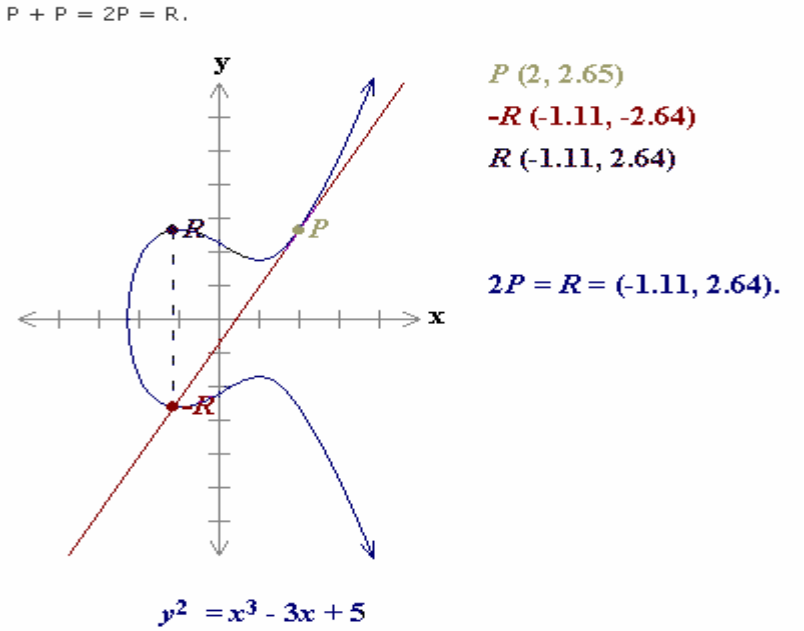
Taškų P ir –P sudėties negalima apibrėžti prieš tai aprašytomis taisyklėmis, nes tiesė einanti per taškus P ir –P nekerta elipsinės kreivės trečiame taške, todėl laikoma, kad $P + (-P) = 0$ ir $P+0=P$.



2.2.2-3pav. Taško sudėties su atvirkštiniu tašku geometrinis atvaizdavimas

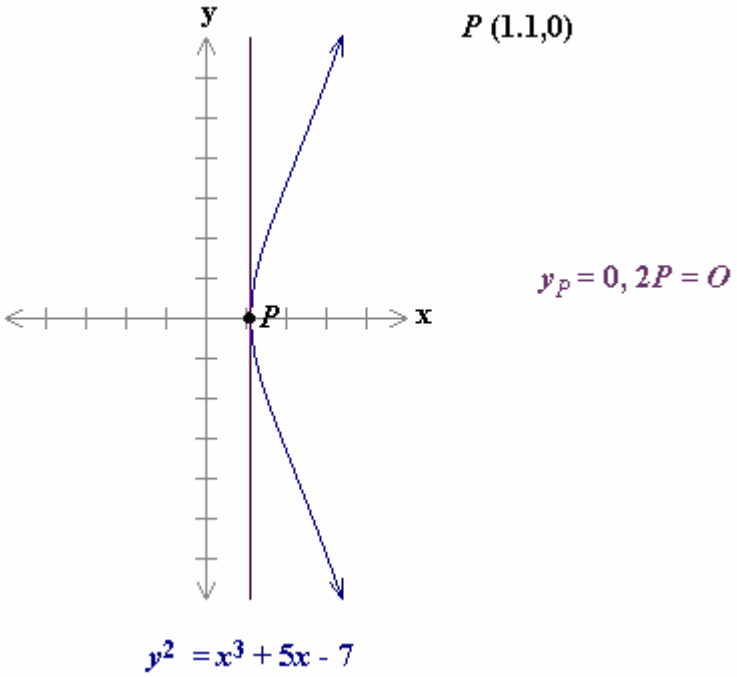
3. Elipsinių kreivių taškų daugyba iš skaliaro.

Norint pridėti tašką P prie jo paties, brėžiama kirstinė linija per tašką P. Jei $y_P \neq 0$, tada linija kerta elipsinę kreivę kitame taške –R. Taško –R atvirkščias taškas R yra vadinamas P + P suma $R=P+P=2P$.



2.2.2-4pav. Elipsinės kreivės taškų daugybos iš skaliaro geometrinis atvaizdavimas

Jei $y_p = 0$, tada kirstinė linija gaunama vertikali ir nekerta elipsinės kreivės kitame taške, todėl tokiu atveju $2P=0$. Jei reikėtų surasti $3P$, tai galima užrašyti, kad $3P=2P+P$, taigi $P+0=P$, todėl $3P=P$, $4P = O$, $5P = P$, $6P = O$, $7P = P$ ir t.t.



2.2.2-5pav. Elipsinė kreivė, kai $y_p=0$ geometrinis atvaizdavimas

Geometrinis operacijų apibrėžimas turi mažai praktinio pritaikymo, todėl svarbesnis algebrinis šių operacijų apibrėžimas.

1. Elipsinės kreivės atvirkštinis taškas.

Taško $P=(x_P, y_P)$ atvirkštinis taškas $-P(x_P, -y_P)$.

2. Elipsinių kreivių taškų sudėtis.

Tarkime, turime du taškus $P = (x_P,y_P)$ ir $Q = (x_Q,y_Q)$ ir $P \neq -Q$, tai $P + Q = R$, kur

$s = (y_P - y_Q) / (x_P - x_Q)$

$x_R = s^2 - x_P - x_Q$

$y_R = -y_P + s(x_P - x_R)$

3. Elipsinių kreivių taškų daugyba iš skaliaro.

Tarkime, turime tašką $P = (x_P,y_P)$, kur $y_P \neq 0$ tai $2P = R$, kur

$s = (3x_P^2 + a) / (2y_P)$

$x_R = s^2 - 2x_P$ ir $y_R = -y_P + s(x_P - x_R)$

Ankščiau apibrėžtos operacijos su elipsinių kreivių taškais taikomos tada, kai elipsinės kreivės apibrėžiamos virš baigtinių kūnų. Kadangi operacijos su realiaisiais skaičiais yra lėtos ir atsiranda apvalinimo paklaidos, kriptografijoje naudojamos elipsinės kreivės virš kūno $\mathbf{F}_P$.

Elipsinės kreivės virš kūno $\mathbf{F}_P$ parametrai a ir b turi būti parenkami iš kūno $\mathbf{F}_P$. Elipsinei kreivei priklauso visi taškai (x, y) kuriems galioja elipsinių kreivių aritmetika moduliu p (x ir y – skaičiai iš $\mathbf{F}_P$). Pavyzdžiui, $y^2 \bmod p = x^3 + ax + b \bmod p$ yra virš kūno $\mathbf{F}_P$ jei a ir b priklauso $\mathbf{F}_P$.

Elipsinės kreivės virš kūno $\mathbf{F}_P$ pavyzdys:

Kai a = 1 ir b = 0, elipsinės kreivės lygtis turi pavidalą $y^2 = x^3 + x$. Panagrinėkime šią elipsinę kreivę virš kūno F_{23}. Tada turime:

$y^2 \bmod p = x^3 + x \bmod p$

Taškas (9, 5) tenkina lygtį, nes

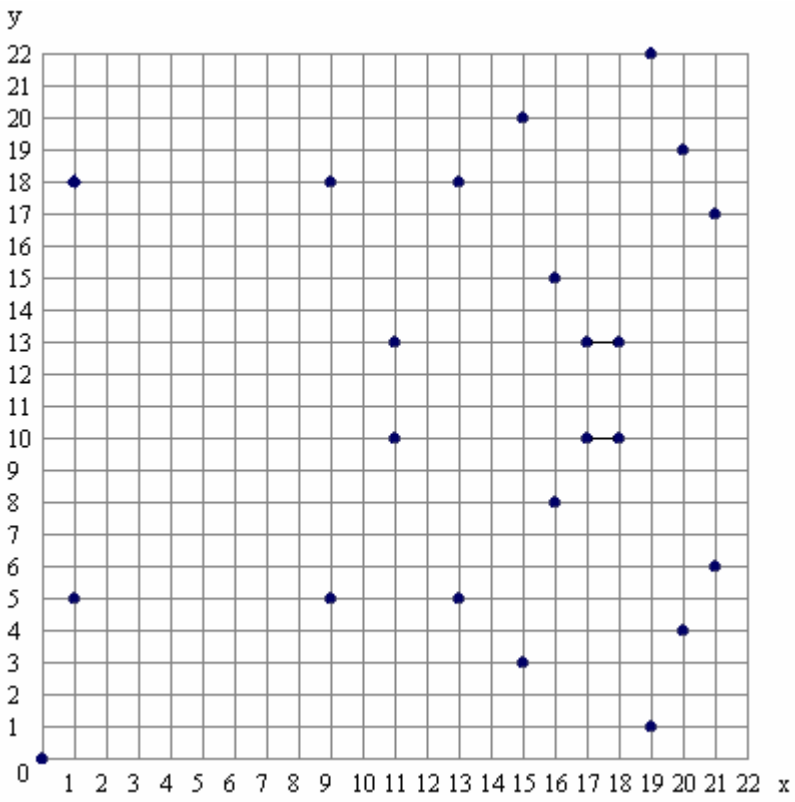
$25 \bmod 23 = 729 + 9 \bmod 23$

$25 \bmod 23 = 738 \bmod 23$

$2 = 2$

Šią lygtį tenkina 23 taškai:

(0,0) (1,5) (1,18) (9,5) (9,18) (11,10) (11,13) (13,5)(13,18) (15,3) (15,20) (16,8) (16,15) (17,10) (17,13) (18,10)(18,13) (19,1) (19,22) (20,4) (20,19) (21,6) (21,17)



2.2.2-6pav. Elipsinės kreivės virš F_{23} grafikas

Kiekvienai x reikšmei egzistuoja du taškai. Iš pirmo žvilgsnio grafiko taškai atrodo išsidėstę atsitiktine tvarka, tačiau jame egzistuoja simetrija, kai y = 11.5. Elipsinės kreivės virš baigtinių kūnų atveju kiekvienam kreivės taškui egzistuoja atvirkštinis taškas. Šiuo atveju atvirkštiniai taškai egzistuoja taip pat tik y reikšmės imamos moduliu p. Tai $-P=(x_p, (-y_p \bmod p))$.

2.2.3 Elipsinių kreivių taškų aritmetika virš kūno $\mathbf{F}_p$

Yra keletas esminių skirtumų tarp elipsinių kreivių virš realiųjų skaičių aibės ir kūno $\mathbf{F}_p$. Elipsinės kreivės virš kūno $\mathbf{F}_p$ turi baigtinį taškų skaičių, ši savybė svarbi elipsines kreives pritaikant kriptografijoje. Kadangi elipsinės kreivės virš kūno $\mathbf{F}_p$ turi keletą diskrečių taškų, nėra aišku kokia tvarka juos sujungti, kad gautume grafiką, panašų į kreivę. Taip pat negalima geometriškai atvaizduoti operacijų su elipsinių kreivių taškais, kaip tai buvo daroma virš realiųjų skaičių aibės. Tačiau matematinės operacijų išraiškos vis tiek gali būti taikomos ir elipsinėm kreivėm virš kūno $\mathbf{F}_p$. Atliekant skaičiavimus čia taip pat nekyla apvalinimo problema, o tai būtina savybė taikant elipsines kreives kriptografijoje. Elipsinės kreivės taškų skaičius virš $\mathbf{F}_p$ vadinamas tos ***kreivės eilė*** virš $\mathbf{F}_p$.

1. Elipsinės kreivės virš kūno F_p atvirkštinis taškas.

Taško $P=(x_P, y_P)$ atvirkštinis taškas $-P = (x_P, -y_P \bmod p)$.

2. Elipsinių kreivių taškų sudėtis virš kūno F_p.

Tarkime, turime du taškus $P = (x_P, y_P)$ ir $Q = (x_Q, y_Q)$ ir $P \neq -Q$, tai $P + Q = R$, kur

$s = (y_P - y_Q) / (x_P - x_Q) \bmod p$

$x_R = s^2 - x_P - x_Q \bmod p$

$y_R = -y_P + s(x_P - x_R) \bmod p$

3. Elipsinių kreivių virš kūno F_p taškų daugyba iš skaliaro.

Tarkime, turime tašką $P = (x_P, y_P)$, kur $y_P \neq 0$ tai $2P = R$, kur

$s = (3x_P^2 + a) / (2y_P) \bmod p$

$x_R = s^2 - 2x_P \bmod p$ ir $y_R = -y_P + s(x_P - x_R) \bmod p$

2.2.4 Elipsinių kreivių diskretaus logaritmo problema

Kiekvienos kriptosistemos pamatas yra sudėtinga matematinė problema, kurios negalima išspręsti. Diskretaus logaritmo problema yra daugelio kriptosistemų saugumo pagrindas. Ne išimtis ir elipsinių kreivių kriptosistemos, kurių pagrindas yra elipsinių kreivių diskretaus logaritmo problema (EKDLP).

Ankščiau apibrėžėme operacijas su elipsinių kreivių taškais: sudėtį ir daugybą iš skaliaro. Tarkime, kad turime elipsinės kreivės tašką P. Galime jį padauginti ir gauti 2P, tada galime sudėti P ir 2P, kad gautume 3P. nP gaunamas pagal taško daugybos iš skaliaro taisykles. Diskretaus logaritmo problema: tarkime, turime P ir Q bei pirminį skaičių p. Reikia rasti k tokį, kad $Pk = Q$, $P = Qk \bmod p$. k yra vadinamas Q diskretus logaritmas pagrindu P.

Pavyzdžiui, turime elipsinę kreivę $y^2 = x^3 + 9x + 17$ virš F_{23}

Koks yra diskretus logaritmas k, kai Q = (4, 5) ir P = (16, 5) ?

Vienas būdas rasti k yra skaičiuoti taškų sumas: P = (16, 5) 2P = (20, 20) 3P = (14, 14)

4P = (19, 20) 5P = (13, 10) 6P = (7, 3) 7P = (8, 7) 8P = (12, 17) 9P = (4, 5)

Kadangi 9P = (4, 5) = Q, tai Q diskretus logaritmas bazei P yra k = 9.

Realiuose uždaviniuose k yra pakankamai didelis, todėl sudėtinga surasti k tokiu būdu.

Kuriant kriptosistemas svarbu:

1. Kokio „dydžio“ yra elipsinė kreivė.
2. Kaip rasti tašką ant elipsinės kreivės.
3. Kaip rasti pačią elipsinę kreivę, tinkamą kriptosistemai.

2.2.5 Elipsinių kreivių taškų skaičiavimo algoritmai

Elipsinių kreivių virš kūno F_p taškų skaičius yra baigtinis ir jis patenka į intervalą $\left(\left(\sqrt{p}-1\right)^2,\left(\sqrt{p}+1\right)^2\right)$. Yra sukurti keli algoritmai, kaip galima rasti konkretų elipsinės kreivės taškų skaičių. Shanks‘o-Mestre metodas gali būti taikomas pirminiams skaičiams nedidesniems nei 10^{30} eilės. Be to, kad rasti kreivės eilę šiam algoritmui reikia $O(p^{1/4+})$ operacijų. Kitas kreivės eilės skaičiavimo algoritmas, kurį panagrinėsime detaliau – Schoof‘o metodas, kuriam reikia $O(\ln^k p)$ fiksuotam k operacijų. Šio algoritmo pagalba galima greitai rasti kreivės eilę kai $p \approx 10^{80}$. Vėliau Atkin‘as ir Elkies‘as šį algoritmą dar patobulino. Schoof‘o metodas remiasi Hesse teorema, teigiančia, jog elipsinės kreivės eilė $\#E(\mathbf{F}_p) = p + 1 - a$, *kai* $|a| \le 2\sqrt{p}$. Metodų aprašymai versti iš ***R. Crandall, C. Pomerance*** knygos *„Prime numbers. A computational Perspective“*.

2.2.6 Shanks‘o-Mestre elipsinių kreivių taškų skaičiavimo algoritmas

Tarkime turime elipsinę kreivę $E = E_{a,b}(\mathbf{F}_p)$. Reikia nustatyti elipsinės kreivės eilę #E. Tarkime, turime funkciją *ind*(S, s) grąžinančią indeksą i tokį, kad $s_i = s$, čia $S=\{s_1, s_2,\ldots\}$ ir $s \in S$. Taip pat turime funkciją *shanks*(), kuri pertvarko du globalius sąrašus A ir B pagal koordinates.

Apibendrinta Shanks‘o-Mestre algoritmo schema:

1. *p* dydžio patikrinimas

   Jei $(p \le 229)$ *rezultatas* $p+1+\sum_x\left(\frac{x^3+ax+b}{p}\right)$

2. Shanks‘o paieška

   Apskaičiuojame kvadratinį likinį g;

   $W = [\, p^{1/4}\sqrt{2}\,]$ *(W - Giant Step parametras)*

   $c = g^2 a$ *(c, d – nauji koeficientai naudojami vietoj a, b)*

   $d = g^3 b$

3. Mestre ciklas

   Parenkamas atsitiktinis $x \in [0, p-1]$;

$s = \frac{x^3 + ax + b}{p}$;

Jei $(s == 0)$ *grįžtame į 3 žingsnį;*

Jei $(s == 1)$ $E = E_{a,b}$; *(kreivei naudojami koeficientai a, b)*

Priešingu atveju {

$E = E_{c,d}$; *(kreivei naudojami koeficientai c, d)*

$x = gx$; *(perskaičiuojama nauja x reikšmė)*

}

Apibrėžiamas pradinis taškas $P \hat{I} E$, kad $x(P) = x$;

S = shanks(P, E);

Jei (#S ≠ 1) grįžtame į 3 žingsnį;

s įtraukiame į aibę S;

= ind(A, s); (s indeksas A aibėje)

= ind(B, s); (s indeksas B aibėje)

t = W;

rezultatas p + 1 + σt; (kreivės eilė)

4. funkcija shanks() {

A = {x([p + 1 +)P: Î [0, W -1]};

B = {x([WpP): Î [0, W]}

Rezultatas A∩B;

}

Šio algoritmo trūkumai - veikimo greitis ir tai, kad jį galima taikyti pakankamai nedideliems pirminiams skaičiams *p* ($p<10^{30}$).

2.2.7 Schoof'o elipsinių kreivių taškų skaičiavimo algoritmas

Schoof'o sudarytas taškų skaičiavimo algoritmas yra polinominio laiko. Jo veikimo trukmė fiksuotam k yra $O(\ln^k p)$. Pagal Hesse teoremą kreivės eilė $\#E(\mathbf{F}_p) = p + 1 - a$.

Pagrindinė Schoof'o koncepcija – nustatyti eilę *#E (mod l),* reikia sudaryti lygtis *a mod l* mažiems pirminiams skaičiams *l*. Tarkime, turime pirminių skaičių seką *S={2, 3, 5, 7.....L},* tokią, kad $\prod_{l \in S} l > 4\sqrt{p}$. Jei būtų galima rasti a mod l kiekvienam pirminiam skaičiui $l \hat{I} S$, tada galima rasti $a \bmod \prod l$, o tuomet ir atkurti *a* reikšmę.

Pirma panagrinėkime *#E (mod 2)* atvejį. Grupės eilė yra lyginė tada ir tik tada, kai egzistuoja elementas, kurio eilė 2. Galima parodyti, kad 2 eilės taškai yra pavidalo *P=(x,0).*

Kreivės eilė yra lyginė tada ir tik tada jei lygtis x^3+ax+b turi šaknį aibėje $\mathbf{F}_p$. O tai galima patikrinti naudojant polinominio laiko didžiausio bendro daliklio skaičiavimo algoritmą.

Kad rastume *#E (mod l)* mažiems pirminiams skaičiams kai *l>2*, įvedame dar kelias priemones elipsinėms kreivėms virš baigtinių kūnų. Tarkime, turime elipsinę kreivę $E(\mathbf{F}_p)$, nagrinėsime kreivės taškus su koordinatėmis aibėje $\boldsymbol{F}_p$. Kėlimas p-tuoju laipsniu tašką $(x, y) \in E(\overline{F}_p)$ nukelia į kitą tašką iš $E(\overline{F}_p)$. Taškų sudėties taisyklėse yra apibrėžiamos racionalios lygties $\mathbf{F}_p$ koeficiento išraiškos, toks išdėstymas yra grupinis $E(\overline{F}_p)$ automorfizmas. Tai Frobenius'o endomorfizmas Φ . Taigi, jei $(x, y) \in E(\overline{F}_p)$, tai $\Phi(x, y) = (x^p, y^p)$ (o taip pat Φ (0) apibrėžiamas kaip O). Kyla klausimas, kam reikia nagrinėti $\boldsymbol{F}_p$ algebrinį uždavinį, jei mus domina tik $\mathbf{F}_p$ apibrėžti taškai. Ryšys matomas iš teoremos: jei elipsinės kreivės grupės $E(\boldsymbol{F}_p)$ eilė yra *p+1-t*, tada

$$\Phi^2(P) - [t]\Phi(P) + [p]P = O \text{ kiekvienam taškui } P \in E(\overline{F}_p).$$

T.y., Frobenijus'o endomorfizmas tenkina kvadratinę lygtį, polinomo x^2-tx+p šaknų suma yra t, o to skaičiaus t pagalba galima apskaičiuoti $E(\boldsymbol{F}_p)$ eilę.

Dabar bet kuriam teigiamam sveikam skaičiui n panagrinėkime tokius taškus P iš $E(\overline{F}_p)$, kuriems *[n]P =O*. Ši aibė žymima *E[n]*, ją sudaro tie eilės taškai, kurių eilė dalina n. Lemiami yra šie du faktai: *E[n]* yra $E(\overline{F}_p)$ pogrupis, ir Ō atvaizduoja *E[n]* į save. Tada turime

$$\Phi^2(P) - [t \bmod n]\Phi(P) + [p \bmod n]P = O, \text{ visiems } P \in E[n]. \quad (1)$$

Schoof'as sugalvojo panaudoti šią lygtį skaičiuojant likinį *t mod n* bandymų ir klaidų metodu tol, kol bus rasta teisinga reikšmė, tenkinanti (1). Šiuo tikslu naudojami vadinamieji dalybos polinomai.

Elipsinei kreivei $E_{a,b}(F_p)$ priskiriame dalybos polinomus $\Psi_n(X,Y) \in F_p[X,Y]/(Y^2 - X^3 - aX - b)$, kaip aprašyta toliau:

$$\Psi_{-1} = -1,\ \Psi_0 = 0,\ \Psi_1 = 1,\ \Psi_2 = 2Y,$$

$$\Psi_3 = 3X^4 + 6aX^2 + 12bX - a^2,$$

$\Psi_4 = 4Y(X^6 + 5aX^4 + 20bX^3 - 5a^2X^2 - 4abX - 8b^2 - a^3)$, o visi kiti atvejai gaunami taip:

$$\Psi_{2n} = \Psi_n(\Psi_{n+2}\Psi_{n-1}^2 - \Psi_{n-2}\Psi_{n+1}^2)/(2Y),$$

$$\Psi_{2n+1} = \Psi_{n+2}\Psi_n^3 - \Psi_{n+1}^3\Psi_{n-1}.$$

Dalybos polinomų konstrukcijoje Y laipsniai, didesni už pirmą laipsnį, redukuojami naudojant sąryšį $Y^2=X^3+aX+b$.

Dalybos polinomų savybių teorema. Dalybos polinomas $\Psi_n(X,Y)$, kai n nelyginis, yra polinomas tik X atžvilgiu, o kai n lyginis, jis yra Y laipsnio polinomas tik kintamojo X atžvilgiu.

Kai n nelyginis ir ne p kartotinis, turime $\deg(\Psi_n) = (n^2 - 1)/2$. Kai n lyginis ir ne p kartotinis, Ψ_n laipsnis kintamojo X atžvilgiu yra *(n²-4)/2*. Taškui *(x, y) Î E(*$\overline{F}_p$*)\E* turime *[n]P =O* tada ir tik tada, kai $\Psi_n(x) = 0$ (kai n nelyginis) ir $\Psi_n(x, y) = 0$ (kai n lyginis). Be to, jei (x, y) ∈ *E(*$\overline{F}_p$*)\E[n]*, tada

$$[n](x, y) = \left(x - \frac{\Psi_{n-1}\Psi_{n+1}}{\Psi_n^2}, \frac{\Psi_{n+2}\Psi_{n-1}^2 - \Psi_{n-2}\Psi_{n+1}^2}{4y\Psi_n^3} \right)$$

Jei *y=0*, n turi būti nelyginis, nes *y=0* žymi 2 eilės tašką, o mes gauname, kad *(x, y)*∉ *E[n]*, todėl šiuo atveju natūralu laikyti šią išraišką lygią *0*.

Kai nelyginis pirminis $l \neq p$, yra toks vienintelis sveikasis t iš [0, *l*-1], kad

$$(x^{p^2}, y^{p^2}) + [p \bmod l](x, y) = [t](x^p, y^p), \text{ visiems } (x, y) \in E[l] \setminus \{O\}. \quad (2)$$

Iš (1) ir dalybos polinomų savybių teoremos gauname, kad *E(*$\overline{F}_p$*)* tikrai turi taškus eilės *l*. Jei būtų galima apskaičiuoti šį vienintelį t, žinotume, kad *E(F*$_p$*)* eilė lygsta su *p+1-t* moduliu *l*.

Šio sąryšio svarba tai, kad naudojant dalybos polinomus galima patikrinti įvairius *t* pasirinkimus, kad rastume tinkamą. Tai daroma taip:

1) Taškai - tai polinomų poros aibėje $\mathbf{F}_p[X, Y]$.
2) Kadangi taškai priklauso *E*, visada galime redukuoti moduliu $Y^2 - X^3 - aX - b$, kad *Y* laipsniai nebūtų aukštesni už pirmą, o kadangi nagrinėjami taškai priklauso *E[n]*, galima redukuoti ir polinomu Ψ_n, kad būtų kontroliuojami X laipsniai. Ir pagaliau koeficientai priklauso $\mathbf{F}_p$, todėl mod *p* redukcijas galima atlikti su koeficientais, kai tik tai patogu. Šių trijų rūšių redukcijas galima atlikti bet kuria tvarka.
3) Aukšti X, Y laipsniai turi būti redukuojami „laipsniavimo laiptais“.
4) Atliekama (2) lygties kairės pusės sudėtis.

Išplėstinis Schoof‘o algoritmas kreivių eilei apskaičiuoti. Tegul *p*>3 yra pirminis. Kreivei $E_{a,b}(\mathbf{F}_p)$ šis algoritmas duoda *t* reikšmę (mod *l*), kur *l* - pirminis (žymiai mažesnis už *p*), o kreivės eilė #E=*p*+1-*t*. Tada tiksli kreivės eilė gaunama panaudojus šį algoritmą pakankamam pirminių skaičių *l* kiekiui, kad $\prod l > 4\sqrt{p}$, o po to, panaudojus kinų liekanos teoremą, gaunama tiksli *t* reikšmė. Mes laikome, kad iki tam tikros ribos $L \geq l$ su galimomis l reikšmėmis, turime jau apskaičiuotus dalybos polinomus $\Psi_{-1}, ..., \Psi_{L+1} \bmod p$, kuriuos galima paversti vienanariais (panaikinus aukšto koeficiento moduliu *p*).

Apibendrinta Schoof'o algoritmo schema:

1. Parenkama pirminių skaičių l aibė $S = \{2, 3, ..., L\}$ tokia, kad $\prod_{l \in S} l > 4\sqrt{p}$, pirminiai l daug mažesni už p;
2. Atliekami skaičiavimai kai $l = 2$;

   Jei $DBD(X^3 + AX + B, X^P - X) \neq 1$ tada $t = 0 \bmod 2$ kitu atveju $t = 1 \bmod 2$;
3. Visiems likusiems $l \in S$ atliekami skaičiavimai:
   a. $p_l = p \bmod l$;
   b. $u(X) = X^P \bmod (\Psi_l, p)$;

      $v(X) = (X^3 + aX + b)^{(p-1)/2} \bmod (\Psi_l, p)$;

      $p_0 = (u(X), Yv(X))$;

      $p_1 = (u(X)^p \bmod (\Psi_l, p), Yv(X)^{p+1} \bmod (\Psi_l, p))$;

      $p_2 = [\bar{p}](X, Y)$ $(N(X)/D(X), YM(X)/C(X))$,

      Jei $p_1 + p_2 = 0$, tai $t = 0 \bmod l$

      $p_3 = p_0$

      Kol $(1 \le k \le l/2)$ {

      Jei $(p_1 + p_2)$ taško X koordinatė sutampa su p_3 taško X koordinate tada,

      Jei sutampa ir Y taško koordinatės tada $t = k \bmod l$, priešingu atveju $t = (l-k) \bmod l$

      $p_3 = p_3 + p_0$

      }

Reikia palyginti ar sutampa $(P1+P2)$ ir $P3$ ir tokiam ryšiui patikrinti naudojamos elipsinių kreivių taškų sudėties taisyklės. Visas aprašomas realizavimas apima tik polinomų daugybą $\bmod(\Psi_l, p)$ ir redukcijas. Ir polinomų daugybą, ir redukciją galima atlikti gana efektyviai.

Skaičiavimų pavyzdys.

Kai p=101 ir kreivė $Y^2 = X^3 + 3X + 4$ virš F_p, pasirinkus l=2,3,5,7, gauname rezultatą t mod $2=0$, $t \bmod 3=1$, $t \bmod 5=0$, $t \bmod 7=3$, iš ko gauname, kad $\#E=92$. (Galėjome praleisti pirminį 5, nes kitų pirminių sandauga viršija $4\sqrt{p}$). Turime tarpinius skaičiavimus, pvz.,

$$\Psi_3 = 98 + 16X + 6X^2 + X^4,$$

$$(X^{p^2}, Y^{p^2}) = (32 + 17X + 13X^2 + 92X^3, Y(74 + 96X + 14X^2 + 68X^3)),$$

$$[2](X, Y) = \left(\frac{12 + 53X + 89X^2}{16 + 12X + 4X^3}, Y \frac{74 + 10X + 5X^2 + 64X^3}{27 + 91X + 96X^2 + 37X^3} \right),$$

$$(X^P, Y^P) = (70 + 61X + 83X^2 + 44X^3, Y(43 + 76X + 21X^2 + 25X^3)),$$

Kiekvienas polinomas, atsirandantis taško koordinatėse, yra redukuotas $\bmod(\Psi_3, p)$.

Gauname, kad paskutinis taškas čia iš tiesų yra dviejų ankstesnių taškų elipsinė suma ir tai atitinka prielaidą, kad *t mod 3 =1*.

Pirminių skaičių laipsniai gerai tinka algoritmo realizavimui. Bendra visų panaudotų pirminių skaičių laipsnių l reikšmių sandauga (bet ne daugiau, kaip po vieną kiekvienam pirminiam) negali viršyti $4\sqrt{p}$.

Pagal šią pagrindinę Schoof'o schemą panaudojant pirminius laipsnius *l<100* galima įvertinti kreivių eilę pirminiams skaičiams kai $p \approx 10^{80}$.

Ar galima atlikti taškų skaičiavimą žymiai didesniems pirminiams skaičiams? Iš tiesų Schoof'o algoritmo transformavimas į Schoof'o-Elkies'o-Atkino (SEA) variantą pagal [Atkin 1986m., 1988m., 1992m.] ir [Elkies 1991m., 1997m.] su skaičiavimų patobulinimais [Motrain 1995m.], [Couveignes ir Morain 1994m.] [Couveignes ir kiti 1996m.] įgijo precedento neturintį taškų skaičiavimo našumą. Elkies'as pastebėjo, kad kai kuriems l (priklausomai nuo a, b, p, iš tiesų apie pusei galimų l reikšmių), galima panaudoti tam tikrą polinomą f_l, Ψ_l dalijantį tik (*l-1)/2* laipsnio. Schoof'o sąryšį (2) galima supaprastinti. Elkies'o metodo esmė buvo rasti vertę l tenkinančią lygybę:

$$(X^p, Y^p) = [l](X, Y),$$

kur visi skaičiavimai atliekami mod *(f_l, p)*, kai *#E=p+1-t* ir $t \equiv l + p/l \pmod{l}$.

Kadangi f_l laipsniai tokie maži, šis svarbus atradimas efektyviai „ištraukia" kai kuriuos *ln p* laipsnius iš sudėtingumo vertinimo ir gaunamas $O(\ln^6 p)$ lyginant su Schoof'o algoritmo sudėtingumu $O(\ln^8 p)$. Pvz., Atkin panaudojo tokius pagerinimus ir 1992m. rado mažiausią pirminį, turintį 200 dešimtainių skaitmenų, o būtent:

p=1000000000000000000000000000000000000000000000000\
0000000000000000000000000000000000000000000000000\
0000000000000000000000000000000000000000000000000\
000000000000000000000000000000000000000000000153,

o kreivė virš $\mathbf{F}_p$, kurią apibrėžia kubinis polinomas $Y^2=X^3+105X+78153$, taškų eilę

#E=1000000000000000000000000000000000000000000000000\
0000000000000000000000000000000000000000000000000\
0678975028800422411808031436546027764192804964188 8\
399915913929600322106305617600290508586136896317 53.

Morain'as sugebėjo padaryti kitus skaičiavimų patobulinimus, kad nustatytų tikslią eilę kreivei virš $\mathbf{F}_p$, kai p - 500 dešimtainių skaitmenų pirminis.

Modifikuota Schoof'o algoritmo schema:

$E(\mathbf{F}_p)$: $y^2 = x^3 + Ax + B$, $p > 3$;

Reikia rasti kreivės eilę $\#E(\mathbf{F}_p) = p + 1 - t$;

1. Parenkama pirminių skaičių l aibė $S = \{2, 3, ..., L\}$ tokia, kad $\prod_{l \in S} l > 4\sqrt{p}$, pirminiai l daug mažesni už p;
2. Atliekami skaičiavimai kai $l = 2$;

   Jei $DBD(x^3 + Ax + B, x^P - x) \neq 1$ tada $t = 0 \bmod 2$ kitu atveju $t = 1 \bmod 2$;
3. Visiems likusiems $l \in S$ atliekami skaičiavimai:
   c. $p_l = p \bmod l$;
   d. $x_0 = x^p \bmod (\Psi_l, P)$; $x_1 = (x_0)^p \bmod (\Psi_l, P)$; $x_2 = x - \frac{\Psi_{p_{l-1}} \Psi_{p_{l+1}}}{\Psi_{p_l}^2}$; $x_3 = x_0$;
   e. Kiekvienam $j = 1, 2 \ldots (l-1)/2$ tikrinama ar taškų $P_1 + P_2$ ir P_3 x koordinatės sutampa:
      i. $G = x_1 - x_2$;
      ii. $\alpha = (x_1 x_2 + A)(x_1 + x_2) + 2B$;
      iii. $\beta = (x_1 x_2 - A)^2 - 4B(x_1 + x_2)$;
      iv. $pol(x) = G^2 x^2 - 2\alpha x + \beta$;
      v. jei $pol(x_3) = 0$ tai $x_1 + x_2 = x_3$; $t = j \bmod l$; $t = -j \bmod l$;
      vi. jei $pol(x_3) \neq 0$ tai $x_3 = x_0 + x_3$ ir grįžtame į punktą c;

Abiejų algoritmo versijų pradžia vienoda - kol dirbama su x koordinatėmis. Gauname lyginių poras $t = a \bmod l$ ir $t = -a \bmod l$. Originaliame Schoof'o algoritme dirbama su koordinatėmis y, kad nuspręsti kuri iš tų dviejų lyginių palikti $t = a \bmod l$ ar $t = -a \bmod l$. Tai atliekame reikiamam kiekiui pirminiu skaičių l. Gaunama lyginių sistema ir iš jos remiantis kinų teorema liekanoms atstatoma t reikšmė, o po to ir pati kreives eilė.

Modifikuotame algoritme kiekvienam pirminiam l gaunama reikšmių pora $t = a \bmod l$ ir $t = -a \bmod l$ (kaip ir anksčiau). Tačiau dabar nebeatliekame skaičiavimų su y koordinatėmis. Gauname dvigubai daugiau lyginių. Nagrinėjame galimas lyginių kombinacijas ir jų sistemos sprendinius.

Esminis klausimas - kaip atrinkti reikiamus lyginius, kad rastumėm reikiamą t.

Gal būt, vienas iš būdų galėtų buti tiesiog galimų lyginių sistemų variantų perrinkimas. Galima pamėginti nagrinėti visas galimas lyginių sistemas ir jų sprendinius. Idėja dirbti su t reikšmių poromis (modifikacija) yra pasiūlyta Crandall ir Pomerance knygoje „Prime numbers. A computational perspective". Autoriai nerašo, ką konkrečiai reikia daryti su tomis

poromis, bet siūlo ieškoti efektyvaus būdo, ką su jomis daryti, kad rastume reikiamą *t*. Gal būt tai būtų greičiau, nei dirbti su *y* koordinatėmis.

2.2.8 Kriptosistemų, paremtų elipsinėmis kreivėmis, saugumas

Elipsinės kreivės pagrindiniai parametrai, apsprendžiantys kriptosistemos saugumą yra:

1. Kūno $\boldsymbol{F}$, virš kurio aprašoma elipsinė kreivė, eilė $q = p^k$.
2. Koeficientai a ir b $\hat{I}$ $\boldsymbol{F}_p$ – apibūdina skaičiavimus su elipsine kreive ($y^2 = x^3 + ax^2 + b$).
3. Du lauko $\boldsymbol{F}_p$ elementai x_p ir y_p, apibūdinantys konkretų elipsinės kreivės tašką $P = (x_p, y_p)$.
4. Elipsinės kreivės eilė n.
5. Daugiklis $h = \#E(\boldsymbol{F}_p)/n$.

Kad kriptosistema būtų nepažeidžiama Pohlig-Hellman'o ir Polard rho atakų reikia, kad kreivės eilė #E($\mathbf{F}_p$) dalytųsi iš pakankamai didelio pirminio skaičiaus n ($n>2^{160}$). Taip pat kreivės eilė #E($\mathbf{F}_p$) turi būti pirminis skaičius arba kreivės eilė $\#E(\boldsymbol{F}_p) = h\,n$, kur n – pirminis skaičius, o h – mažas koeficientas (pvz. 1, 2, 3 arba 4).

Kad kriptosistema būtų nepažeidžiama anomalinių kreivių virš pirminių kūnų atakų reikia, kad #E($\mathbf{F}_p$) $\neq p$.

Kad sistema būtų atspari Weil'o ir Tate'o atakoms reikia, kad n nesidalintų iš $p^k - 1$ visiems $1 \le k \le C$, kur C pakankamai didelis koeficientas (pvz. Kai $n > 2^{160}$, $C = 20$).

2.2.9 Raktų apsikeitimo schema

Tam, kad du dalyviai A ir B galėtų apsikeisti koduotais duomenimis, pirmiausia kiekvienas turi turėti savo kodavimo raktų porą: viešąjį raktą ir privatų raktą. Raktų generavimui pirmiausia paimamas pirminis skaičius $p \approx 2^{180}$ ir parametrai a ir b elipsinei kreivei E gauti. Tada gaubiama elipsinės kreivės taškų aibė E_p (a,b), iš kurios išrenkamas generuojantis taškas $G = (x_1,y_1)$. Išrenkant tašką G svarbu, kad mažiausia n reikšmė, su kuria $n \times G = 0$, būtų pakankamai didelis pirminis skaičius. Parametrai E_p (a,b) ir G žinomi visiems dalyviams.

Raktų apsikeitimas tarp dalyvių A ir B vyksta pagal schemą:

1. Dalyvis A išsirenka sveikąjį skaičių n_A mažiau už n. Šis skaičius tai privatus dalyvio A raktas. Tada dalyvis A sugeneruoja viešąjį raktą $P_A = n_A \times G$.
2. Dalyvis B analogiškai pasirenka privatųjį raktą n_B ir sugeneruoja viešąjį raktą P_B.
3. Dalyviai A ir B apsikeičia viešaisiais raktais ir sugeneruoja bendrąjį raktą K:

a. Dalyvis A: $K = n_A \times P_B$
b. Dalyvis B: $K = n_B \times P_A$

2.2.10 Duomenų kodavimas / dekodavimas panaudojant elipsines kreives

Apžvelgsime patį paprasčiausią kodavimo / dekodavimo atvejį. Taigi reikia užkoduoti pranešimą M, kuris gali būtį laikomas elipsinės kreivės tašku $P_m (x,y)$.

Kaip ir raktų apsikeitimo atveju, reikalinga elipsinė kreivė $E_p (a,b)$ ir taškas G ant jos. Dalyvis B pasirenka privatų raktą n_B ir sugeneruoja viešąjį raktą $P_B = n_B \times G$. Pranešimo užkodavimui naudojamas pranešimo gavėjo B viešasis raktas P_B. Dalyvis A išsirenka atsitiktinį sveikąjį teigiamą skaičių k ir užkoduotą pranešimą C_m, kuris taip pat yra elipsinės kreivės taškas $C_m = \{k \times G, P_m + k \times P_B\}$

Kad atkoduotumėme pranešimą, dalyvis B pirmąją taško koordinatę daugina iš savo privataus rakto.

$P_m + k \times P_B - n_B \times (k \times G) = P_m + k \times (n_B \times G) - n_B \times (k \times G) = P_m$

Dalyvis A užkodavo pranešimą P_m pridėdamas prie jo kxP_B. Niekas nežino k reikšmės, todėl nors P_B ir yra viešasis raktas, tačiau niekas nežino $k \times P_B$. Jei kas mėgins atkoduoti pranešimą, jam reiks rasti k reikšmę žinant G ir $k \times G$. Tai padaryti bus nelengva.

Pranešimo gavėjas taip pat nežino skaičiaus k, tačiau padauginęs $k \times G$ iš savo privataus rakto gavėjas gauna reikšmę, kurią siuntėjas pridėjo prie nekoduoto pranešimo, taigi gavėjas nežinodamas k, bet turėdamas savo privatų raktą gali atkoduoti pranešimą.

3. Projektinė dalis

3.1 Uždavinys ir jo analizė

Uždavinys: reikia sukurti programą, realizuojančią Schoof‘o taškų ant elipsinių kreivių skaičiavimo algoritmą.

3.2 Darbo priemonių parinkimas

3.2.1 Programavimo sistemos pasirinkimas

Sparčiai tobulėjant kompiuterinei įrangai, keičiasi ir programinė kompiuterių įranga, o kartu ir programavimo sistemos bei kalbos. Programavimo kalbos ir sistemos pasirinkimą dažniausiai lemia pastarųjų galimybės. Šis darbas atliktas naudojant tris programavimo sistemas: ***Delphi 2005*** sistemą, kur realizuota ***Object Pascal*** programavimo kalba, ***Visual C++*** ir ***Python***. Visos trys programavimo sistemos palaiko klasių mechanizmą. Taikant objektinio programavimo metodiką, programa sudaroma iš abstraktaus tipo objektų, kurie turi tokias savybes:

- yra aprašomi parametrų rinkiniais, kurie nusako jų būvį;
- turi būvių apklausos ir valdymo priemonių (metodų) rinkinius;
- turi specialius objektų konstravimo ir naikinimo metodus;
- gali turėti vidinius (uždarus) ir išorinius (atvirus) elementus;
- iš kiekvienos abstraktaus tipo objektų klasės gali būti kuriamos ištisos išvestinių tipų šeimos, paveldinčios bazinio tipo savybes;
- tipų šeimose gali būti automatiškai keičiamos metodų savybės.

Projektuojant objektus, reikia turėti priemones, kurios leistų duomenų struktūras ir jų apdorojimo metodus sujungti į vientisą struktūrinį vienetą. Toks jų sujungimas realizuojamas klasėse. Delphi sistema turi savo klasių biblioteką VCL (Visual Component Library), sistema palaiko PME (Property – Method – Event) modelį, t.y. objektai turi savybes (Property),

apibūdinančias to objekto stovį, turi metodus (Method)– funkcijas, apdorojančias savybių reikšmes ir kitus duomenis. Metodai iškviečiami įvykus kokiam nors įvykiui (Event).

Visual C++ sistema turi MFC (Microsoft Foundation Classes) klasių biblioteką, čia realizuotos didesnės klasių kūrimo galimybės. Be to, C kalbos sintaksė žymiai efektyvesnė lyginant su Object Pascal. Tačiau lyginant Visual C++ ir Delphi sistemas, nemažai daliai programuotojų, o ypatingai su mažesne programavimo patirtimi, priimtinesnė Delphi, nes čia yra tokios priemonės kaip Object Inspector, kur galima greitai nustatyti pradines objektų savybių reikšmes. Be to, čia komponentų (VCL ir kitų klasių objektų) pradiniai nustatymai gali būti valdomi vizualiai. Tai leidžia greitai ir efektyviai kurti patogią ir modernią vartotojo grafinę sąsają. Nepaisant mažesnio Object Pascal kalbos efektyvumo ir mažesnių klasių mechanizmo galimybių lyginant su C kalba, Delphi sistema tinka projektuoti ir kurti tiek mažom, tiek pakankamai didelėm sistemom.

Python yra interpretuojama, interaktyvi programavimo kalba sukurta 1990 metais. Pirmiausiai ji buvo scenarijų kalba AmoebaOS operacinei sistemai. Python dažniausiai lyginama su Perl, Java. Python kuriama kaip atviro kodo projektas. Python - daugiaparadigminė programavimo kalba - ji leidžia naudoti keletą programavimo stilių: objektinį, struktūrinį, funkcinį. Python naudoja dinaminį tipų tikrinimą. Python kalbos savybės:

- viskas yra objektai, galima sukurti klases, praplečiančias standartinius duomenų tipus;
- galimas paveldėjimas iš keletos klasių;
- labai svarbus kodo išdėstymas (indentation);
- modulių sistema;
- dėl indentacijos (indentation), galimas praktiškai vienintelis būdas (skiriasi tik tarpų/tabuliacijos ženklų vartojimas) parašyti kodą, todėl lengva dirbti grupėse;
- kodas gali būti kompiliuojamas į vidinę formą, kas leidžia greičiau įkrauti daug kartų naudojamus modulius ir pan.

3.2.2 Papildomos bibliotekos

Kriptosistemoms naudojami pakankamai dideli pirminiai skaičiai > 2^{160}, todėl standartinių duomenų tipų, realizuotų programavimo sistemose nepakanka. Todėl reikalingos papildomos priemonės, praplečiančios programavimo sistemą ir suteikiančios galimybę atlikti operacijas su dideliais skaičiais. Šiame darbe naudojama viena tokių priemonių – biblioteka ***Miracl***, skirta C++ programavimo sistemoms, kurioje aprašytas naujas duomenų tipas **Big** bei galimos aritmetinės operacijos su šio tipo duomenimis. Taip pat šioje bibliotekoje realizuotas pirminių skaičių paieškos bei skaičiaus patikrinimo ar jis yra pirminis algoritmai. Ši biblioteka taip pat

pateikia priemones darbui su polinomais. Visos šios priemonės reikalingos elipsinių kreivių aritmetikai bei taškų ant elipsinių kreivių skaičiavimo algoritmams realizuoti.

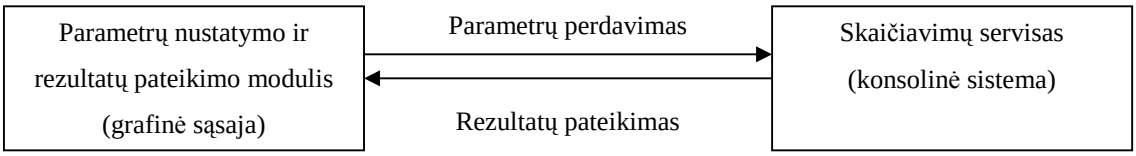
Kita darbe naudojama biblioteka – **NZMATH,** skirta Phyton programavimo sistemai. Čia realizuotos klasės, skirtos darbui su elipsinėmis kreivėmis, polinomais. Bibliotekos trūkumas – gali atlikti skaičiavimus su palyginti nedideliais skaičiais ir skaičiavimai atliekami pakankamai lėtai.

4. Darbo eigos aprašymas

4.1 Projekto moduliai

Sistema skaičiuoja nurodytos elipsinės kreivės eilę naudojant Schoof‘o algoritmą. Skaičiavimams atlikti naudojama biblioteka Miracl skirta C++ programavimo sistemai. Tačiau sistema Delphi turi kur kas paprastesnį grafinės vartotojo sąsajos kūrimo mechanizmą. Todėl šiam uždaviniui realizuoti buvo pasirinktos abi sistemos ir užduotis išskaidyta į du modulius:

1. Skaičiavimų atlikimo servisai. Šie servisai iškviečiami konsolės rėžime perduodant elipsinės kreivės bei skaičiavimo rėžimo parametrus ir atlieka visus pagrindinius skaičiavimus.
2. Serviso valdymo modulis – Windows 32 programa, suteikianti patogią (vizualią) aplinką parametrų nustatymui bei rezultatų peržiūrai. Šis modulis iškviečia skaičiavimų servisą su nustatytais parametrais ir apdoroja serviso skaičiavimų rezultatus.



4.1-1pav. Serviso valdymas

Skaičiavimų servisai realizuoti kaip konsolinės programos, kurioms per komandinę eilutę paduodami parametrai:

- Pirminis skaičius P;
- Elipsinės kreivės parametras A;
- Elipsinės kreivės parametras B;
- Skaičiavimo rėžimai:
- Kito pirminio skaičiaus P paieškos rėžimas. Jei servisui pateiktas parametras P nėra pirminis skaičius, servisas pats suranda artimiausią pirminį skaičių didindamas arba mažindamas pateiktą parametrą.
- Kreivės, kurios eilė – pirminis kaičius paieškos rėžimas. Servisas gali keisdamas parametrą B surasti kreivę, kurios eilė – pirminis skaičius.

Grafinės sąsajos modulyje realizuotas mechanizmas iškviečia šį servisą su atitinkamai nustatytais parametrais, nuskaito konsolės išvestį (serviso darbo rezultatus) bei juos apdoroja. Šios technologijos pagalba suderinamos dvi programavimo sistemos ir išnaudojami kiekvienos jų privalumai.

Dar vienas tokios architektūros privalumas, jog bet kada sistemą galima papildyti naujais servisais (pvz. kitais taškų ant elipsinių kreivių skaičiavimo algoritmais), be to galima iškviesti kelis servisus ir atlikti kelis skirtingus skaičiavimus tuo pačiu metu.

4.1.1 Skaičiavimų atlikimo serviso veikimo schema

Realizuoti skaičiavimų atlikimo servisai :

1. Originalus Schoof‘o algoritmas MS Visual C++ programavimo sistema pasinaudojant MIRACL biblioteka;
2. Originalus Schoof‘o algoritmas Phyton programavimo sistema pasinaudojant NZMATH biblioteka;

Originalaus Schoof‘o algoritmo veikimo schema:

I. Nustatomi parametrai, $A, B, p, E(\mathbf{F}_p): Y^2 = X^3 + AX + B, p > 3$; Reikia rasti kreivės eilę $\#E(\mathbf{F}_p) = p + 1 - t$;

II. Atliekami skaičiavimai:

1. Parenkama pirminių skaičių l aibė $S = \{2, 3, ..., L\}$ tokia, kad $\prod_{l \in S} l > 4\sqrt{p}$, pirminiai l daug mažesni už p;
2. Atliekami skaičiavimai kai $l = 2$;
   Jei $DBD(X^3 + AX + B, X^P - X) \neq 1$ tada $t = 0 \bmod 2$ kitu atveju $t = 1 \bmod 2$;
3. Visiems likusiems $l \in S$ atliekami skaičiavimai:
   f. $p_l = p \bmod l$;
   g. $u(X) = X^P \bmod (\Psi_l, p)$;
   h. $v(X) = (X^3 + aX + b)^{(p-1)/2} \bmod (\Psi_l, p)$;
   i. $p_0 = (u(X), Yv(X))$;
   j. $p_1 = (u(X)^p \bmod (\Psi_l, p), Yv(X)^{p+1} \bmod (\Psi_l, p))$;
   k. $p_2 = [\bar{p}](X, Y)$ $(N(X)/D(X), YM(X)/C(X))$,
   l. Jei $p_1 + p_2 = 0$, tai $t = 0 \bmod l$
   m. $p_3 = p_0$
   n. Kol $(1 \le k \le l/2)$ {
      Jei $(p_1 + p_2)$ taško X koordinatė sutampa su p_3 taško X koordinate tada,

Jei sutampa ir *Y* taško koordinatės tada *t = k mod l*, priešingu atveju *t = (l-k) mod l*

$p_3=p_3+p_0$

}

4. Iš gautų lygčių pasinaudojant CRT (kinų teorema) išskaičiuojamas parametras *t*.

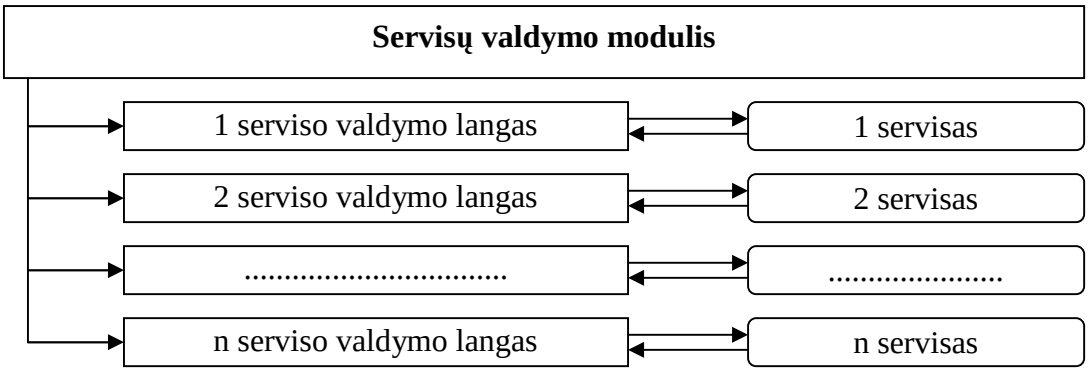
III. Išvedami rezultatai.

Skaičiavimams atlikti realizuota klasė:

```
class TSchoof {
        Big A,B;        //koeficientai a ir b
        Big P;          //pirminis skaičius p
        //Elipsinės kreivės taškų dauginimo funkcija
        void el_tasku_sandauga(PolyMod& X,PolyMod& Y,PolyMod& Z);
        //kreivės taškų sudėties funkcija
        void el_tasku_sudetis(PolyMod& XT,PolyMod& YT,PolyMod& ZT,PolyMod&
X,PolyMod& Y,PolyMod& Z);
        //kėlimas laipsniu
        mr_utype klaipsniu(mr_utype x,int y);
        //kreivės taškas
        void el_taskas(Big p, Big nrp);
public:
        TSchoof(ZZn a, ZZn b, Big p) {A=a; B=B; P=p; }
        ~TSchoof() {};
        Big sschoof(Big a, Big b, Big p, int pbits, BOOL search, BOOL anomalous);
};
```

4.1.2 Servisų valdymo modulis

Serviso valdymo modulis skirtas paprastesniam ir greitesniam programos naudojimuisi. Kadangi skaičiavimams atlikti naudojami servisai ir tie servisai skaičiavimus gali atlikti keli vienu metu, tai ir valdymo modulis realizuotas taip, kad vienu metu būtų galima valdyti keletą skaičiavimų servisų. Tokiai technologijai realizuoti panaudojamas lygiagretaus programavimo mechanizmas. T.y. kiekvienam skaičiavimo servisui iškviesti ir jo rezultatams apdoroti sukuriamas atskiras procesas (thread). Kiekvienos užduoties parametrams įvesti bei rezultatams išvesti dinamiškai kuriamas serviso valdymo langas.



4.1.2.-1 pav. Servisų valdymo modulis

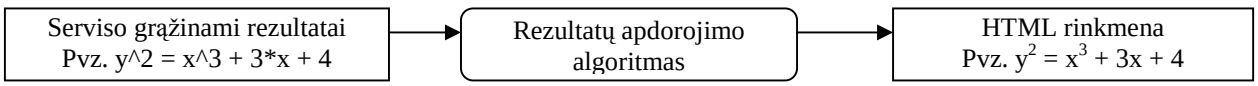
4.2 Problemų ir jų sprendimų aprašymai ir pagrindimai

Dirbant su kriptosistemomis tenka operuoti dideliais ($\approx 2^{150}$) skaičiais. Standartiniai programavimo sistemų duomenų tipai yra nepakankami. Todėl kyla problema tuos didelius skaičius aprašyti bei atlikti operacijas su jais. Šią problemą galima išspręsti aprašius naują vartotojo duomenų tipą bei operacijas su šio tipo duomenimis. Toks sprendimas realizuotas C++ programavimo sistemai skirtoje bibliotekoje MIRACL.

Kriptosistemose naudojami dideli pirminiai skaičiai. Kita problema – didelių pirminių skaičių generavimo ir patikrinimo ar skaičius pirminis algoritmo realizavimas. Šią problemą vėl padeda išspręsti MIRACL biblioteka, kurioje realizuotas pirminių skaičių sekos generavimo algoritmas bei priemonės nustatymui, ar skaičius yra pirminis.

Realizuojant Schoof‘o algoritmą reikia atlikti operacijas su polinomais (daugianariais). Tokias operacijas aprašyti programavimo sistema pakankamai sudėtinga. MIRACL biblioteka suteikia galimybę atlikti polinomų sudėtį, atlikti skaičiavimus pagal Kinų teoremą.

Kita problema – korektiškas rezultatų atvaizdavimas. Paprastuose tekstiniuose laukuose išvesti sudėtingas polinomų formules sudėtinga. Kartais tam naudojami specialūs žymėjimai pvz. a kėlimas laipsniu n – a^n. Tačiau tokia forma peržiūrėti rezultatus nėra patogu, tuo labiau kai išvedamos ilgos polinomų formulės. Todėl geresniam rezultatų atvaizdavimui buvo nuspręsta panaudoti HTML formatą, kuriuo nesunku atvaizduoti matematinius simbolius. Skaičiavimų servisas valdymo moduliui grąžina sudėtingai skaitomas lygčių išraiškas, o valdymo modulis apdoroja serviso rezultatus ir iš jų generuoja HTML formato rinkmeną, kuri atvaizduojama interneto naršymo programų pagalba.



4.2-1pav. Rezultatų atvaizdavimas

4.3 Galutinio projekto stovio aprašymas

Pagrindinė praktinė darbo dalis – realizuoti Schoof‘o algoritmą. Schoof‘o algoritmas buvo realizuotas. Sistemoje taip pat realizuota keletas papildomų funkcijų, tokių kaip pirminio skaičiaus P paieška, elipsinės kreivės paieška keičiant parametrus jei vartotojo pateikta kreivė yra neleistina arba kai norima rasti elipsinę kreivę, kurios eilė yra pirminis skaičius. Taigi pagrindinis uždavinys įgyvendintas. Tačiau sistemą dar galima tobulinti siekiant ją pagreitinti bei galima realizuoti daugiau papildomų funkcijų tokių kaip atsitiktinių kreivių generavimo bei kreivių tinkamumo kriptografijai įvertinimo algoritmai.

5. Išvados

1. Apžvelgti ir palyginti taškų ant elipsinių kreivių skaičiavimo algoritmai:
    a. Shanks‘o-Mestre metodas – lėtas, reikalaujantis $O(p^{1/4+})$ operacijų (sudėtingumo), netinkamas dideliems p;
    b. Schoof‘o metodas – žymiai greitesnis už Shanks‘o-Mestre metodą, tinkamas didesniems p ($p \approx 10^{80}$), reikalaujantis mažesnio sudėtingumo $O(\ln^k p)$ fiksuotam k ($k \approx 8$) operacijų;
    c. Atkin‘o ir Elkies‘o Schoof‘o algoritmo modifikacija SEA tinka p, turintiems kelis šimtus skaitmenų ir gaunamas mažesnis $O(\ln^6 p)$ sudėtingumas lyginant su Schoof‘o algoritmo sudėtingumu $O(\ln^8 p)$.
2. Sukurta programa, realizuojanti Schoof‘o metodą:
    a. jos architektūra leidžia vienu metu atlikti keletą skirtingų skaičiavimų;
    b. numatyta galimybė praplėsti skaičiavimus integruojant naujus skaičiavimų servisus, kuriuose realizuoti kiti taškų skaičiavimo algoritmai;
    c. programą galima praplėsti papildomomis funkcijomis (atsitiktinių kreivių generavimo bei kreivių tinkamumo kriptografijai įvertinimo algoritmai).
    d. lygiagretaus programavimo technologijos programai suteikia daugiau efektyvumo.
3. Schoof‘o elipsinių kreivių taškų skaičiavimo algoritmas realizuotas dviem skirtingomis programavimo sistemomis MS Visual C++ ir Phyton kiekvienai sistemai naudojant skirtingas bibliotekas. Algoritmas realizuotas MS Visual C++ sistema skaičiavimus atlieka žymiai greičiau.

6. Literatūra

1. Blonskis J. Bukšnaitis B., Dagienė V. ir kt. Programavimas Delphi. V., 2003
2. Crandall R., Pomerance C., Prime numbers. A computational Perspective, Berlin: Springer 2002.
3. Hankerson D., Henezes A., Vanstone S., Guide to Elliptic Curve Cryptography, Berlin: Springer 2003.
4. Šleževičienė R., Kriptografijos įvadas, Šiauliai: ŠU leidykla 2005.
5. Vidžiūnas A., Blonskis J. Bukšnaitis B., Brazdaitis V. Delphi 5 . K., 2000
6. Vidžiūnas A., C++ ir C++ Builder pradmenys, K., 2002.
7. Washington L. C., Elliptic curves. Number Theory and Cryptography.
8. Operacijos su elipsinių kreivių taškais http://www.intuit.ru/department/security/networksec/11/1.html
9. Online Elliptic Curve Cryptography Tutorial http://www.certicom.com/index.php?action=ecc_tutorial,home
10. Kriptografijos samprata http://www.elektronika.lt/theory/theme/160/793/
11. Цифровая подпись. Эллиптические кривые http://www.morepc.ru/security/crypt/os200207010.html?print
12. Dr. Michael Ganley, Метод эллиптических кривых http://www.racal.ru/rsp/eliptic_curve_cryptography.htm
13. ELLIPTIC CURVE DEMO http://www.cryptomathic.com/labs/ellipticcurvedemo.html#Key-Generation (testavimui)
14. Elliptic Curve http://mathworld.wolfram.com/EllipticCurve.html
15. А.Г. Ростовцев, О выборе эллиптической кривой над простым полем для построения криптографических алгоритмов http://www.ssl.stu.neva.ru/ssl/publications/magazine/1999/3/4/rostovtsev.pdf
16. А. Г. Ростовцев, Е. Б. Маховенко, ИЗОГЕНИИ СТЕПЕНИ 2 И БЫСТРАЯ АРИФМЕТИКА ЭЛЛИПТИЧЕСКИХ КРИВЫХ http://ns.ssl.stu.neva.ru/ssl/archieve/izogenii.pdf
17. В В Острик, М А Цфасман Алгебраическая геометрия и теория чисел расиональные и эллиптические кривые http://www.math.ru/lib/files/pdf/mp-seria/book.8.pdf
18. Phyton programavimo sistemos puslapis http://www.python.org

7. Anotacija

Pocienė, Jurgita. Informatikos magistro baigiamasis darbas. Elipsinių kreivių taškų skaičiavimo algoritmai ir jų taikymai. Darbo vadovė dr. R. Steuding. Šiaulių universitetas. – Šiauliai, 2006. – 35 lapai.

Šiame darbe nagrinėjami taškų ant elipsinių kreivių virš kūno F_p (kūnas virš pirminių skaičių lauko) skaičiavimo algoritmai ir jų taikymo galimybės.

Magistro darbe iškelti pagrindiniai tikslai (išnagrinėti elipsinių kreivių taškų skaičiavimo algoritmus ir juos palyginti, apžvelgti elipsinių kreivių taškų skaičiavimo algoritmų taikymą, realizuoti Schoof‘o elipsinių kreivių taškų skaičiavimo algoritmą ir panagrinėti jo efektyvizacijos galimybes) buvo pasiekti. Buvo išnagrinėti taškų ant elipsinių kreivių skaičiavimo algoritmai, jie palyginti. Buvo sukurta sistema, realizuojanti Schoof‘o algoritmą - vieną iš svarbiausių taškų ant elipsinių kreivių skaičiavimo algoritmų.

Pagrindinės iškilusios problemos buvo: standartiniai programavimo sistemų duomenų tipai yra nepakankami operuoti dideliais ($\approx 2^{150}$) skaičiais, tam buvo panaudota biblioteka MIRACL, leidžianti programoje naudoti didelius skaičius, nustatyti ar skaičius yra pirminis, atlikti skaičiavimus su polinomais. Rezultatų išvedimas vartotojui priimtinesniu pavidalu (polinomų lygčių išvedimui) panaudotas HTML formatas.

Taip pat padarytos išvados, kad elipsinių kreivių eilės nustatymas svarbus kriptosistemai parenkant saugią kreivę. Kreivės eilei suskaičiuoti sukurta keletas metodų, iš jų vienas svarbiausių – Schoof‘o algoritmas. Elipsinių kreivių kriptosistemos gali būti taikomos silpnesniems įrenginiams (mobiliems ir „SmartPhone“ telefonams, delniniams kompiuteriams).

Summary

Pociene, Jurgita. Informatics Master's Final Thesis. Elliptic Curve Points Calculation Algorithms and their Application. Work leader dr. R. Steuding. Siauliai University. Siauliai, 2006. 35 pages

In the work I analyse calculation algorithms of the points on elliptic curves above a body F_p (the body is above primary numbers' field) and their application opportunities.

Basic aims, set for the master's thesis (to analyse elliptic curve points calculation algorithms and to compare them, to review application of elliptic curve points calculation algorithms, to realize Schoof elliptic curve points calculation algorithm and to analyse their effectivization possibilities) were attained. Elliptic curve points calculation algorithms were analysed and compared. System, realizing Schoof algorithm – one of the most important of elliptic curve points calculation algorithms – was created.

Main problems encountered include: standard programming system data types are insufficient to operate with large ($\sim 2^{150}$) numbers, therefore MIRACL library was employed, enabling use of large numbers in a program to find out whether the number is primary and to perform calculations with polynomials. For result output HTML was used as the form more acceptable for user (to derive polynomial equations).

Also it was concluded that finding order of elliptic curves is important for cryptosystem to select a safe curve. A number of methods were created to calculate order of a curve, including one or the most important – Schoof algorithm. Elliptic curve cryptosystems can be applied in the weaker devices (mobile and "Smart Phone" telephones, palm PC).