KAUNO TECHNOLOGIJOS UNIVERSITETAS

MATEMATIKOS IR GAMTOS MOKSLŲ FAKULTETAS

TAIKOMOSIOS MATEMATIKOS KATEDRA

Lukas Litvinas

DIFIO IR HELMANO APSIKEITIMO RAKTAIS ALGORITMO ANALIZĖ SKIRTINGOSE GRUPĖSE

Bakalauro darbas

Vadovas
lekt. dr. K. Lukšys

KAUNAS, 2015

KAUNO TECHNOLOGIJOS UNIVERSITETAS

MATEMATIKOS IR GAMTOS MOKSLŲ FAKULTETAS

TAIKOMOSIOS MATEMATIKOS KATEDRA

TVIRTINU
Katedros vedėjas
doc. dr. N.Listopadskis

DIFIO IR HELMANO APSIKEITIMO RAKTAIS ALGORITMO ANALIZĖ SKIRTINGOSE GRUPĖSE

Bakalauro darbas
Taikomoji matematika (612G10002)

Vadovas
lekt. dr. K. Lukšys
2015 06 02

Recenzentas
lekt. R. Vilkas
2015 06 05

Atliko
MGTM 1/3 gr. stud.
L. Litvinas
2015 06 02

KAUNAS, 2015

KAUNO TECHNOLOGIJOS UNIVERSITETAS

Matematikos ir gamtos mokslų fakultetas

(Fakultetas)

Lukas Litvinas

(Studento vardas, pavardė)

Taikomoji matematika (612G10002)

(Studijų programos pavadinimas, kodas)

Bakalaurinio darbo „Difio ir Helmano apsikeitimo raktais algoritmo analizė skirtingose grupėse“

AKADEMINIO SĄŽININGUMO DEKLARACIJA

20 ___ m. _____ ____ d.

Kaunas

Patvirtinu, kad mano, **Luko Litvino**, bakalaurinis darbas tema „Difio ir Helmano apsikeitimo raktais algoritmo analizė skirtingose grupėse“ yra parašytas visiškai savarankiškai ir visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

Aš suprantu, kad išaiškėjus nesąžiningumo faktui, man bus taikomos nuobaudos, remiantis Kauno technologijos universitete galiojančia tvarka.

_____ _____

(vardą ir pavardę įrašyti ranka) *(parašas)*

TURINYS

Litvinas, L. Diffie and Hellman key exchange algorithm analysis in different groups. *Bachelor's Degree in Applied Mathematics* final thesis / supervisor lect. dr. Kęstutis Lukšys; Kaunas University of Technology, Faculty of Mathematics and Natural Sciences, Department of Applied Mathematics.

Kaunas, 2015. 61 p.

SUMMARY (SANTRAUKA UŽSIENIO KALBA)

The Diffie-Hellman key exchange is a specific method of securely exchanging cryptographic keys over a public channel and was one of the first public-key protocols. In the begining of this paper there is a short overview of the most common key exchange algortihms. DH is one of the earliest practical examples of public key exchange implemented within the field of cryptography. Diffie-Hellman key exchange method allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure channel. This key can then be used to encrypt subsequent communications. Primarily algorithm was implemented and executed in big integer multiplicative and elliptic curves points of the plane groups. Later on, average execution durations were measured and compared. This measurement was made to simply determine the efficiency of the examined algorithm. In the comparison of the same strength security levels between both types of groups, we have discovered that DH algorithm works faster in elliptic curves. Moreover, we have comfirmed that as we expected, algorithm speed in these groups has a power dependence, when increasing the row of a prime number. Secondly, we have managed to perform its implementation in matrices subgroups, where DH algorithm mechanism was based on matrices multiplication instead of exponentiation. This time, its security was based on a generating element features, but not a discrete logarithm problem, as it was in a previously mentioned types of groups. Since there are no officially formulated requirements to compare if Diffie-Hellman key exchange algorithm average execution time is shorter in elliptic curves or matrices subgroups, we did not try to obtain that. The main target considered in this paper was to find matrix row dependence on prime number row at fixed elliptic curves security strength. After we have successfully accomplished estimations and analysis procedure, as a final result of our task we have managed to formulate this principle as a law. We have discovered that a connection between matrices and prime numbers rows describes power dependence. During the whole realization procedure of Diffie-Hellman key exchange algorithm in all mentioned groups above, we have used a lot of specific methods, such as Miller-Rabin primality test. A great amount of iterations have been made to guarantee an extremely precise results.

ĮŽANGA

Senovėje kriptografija naudota apsaugoti ypač slaptas bendravimo formas – šinpų ir diplomatų siunčiamus pranešimus ir panašiai. Šiais laikais kriptografija plačiai naudojama ypač slaptos informacijos saugojimui bei jos siuntimui atvirais tinklais, kaip internetas. Pirmasis sąvoką kriptografija panaudojo Džonas Vilijamsas 1661 metais. Jei kriptosistema naudoja raktą ar slaptažodį, tai dažniausiai yra silpniausia sistemos vieta, tada atakose naudojama nuosekli paieška. Simetrinio rakto kriptografijai dažniausiai taikomi tiesinės ir diferencinės kriptoanaliznės metodai. Šis trūkumas paskatino toliau domėtis iškilusia saugumo problema, kuri vėliau tapo baigiamojo darbo pasirinkimo priežastimi. Problemai išspręsti taikoma viešojo rakto (asimetrinė) kriptografija – moderni kriptografijos forma, eliminuojanti simetrinės kriptografijos silpnybę – šifravimui skirto rakto perdavimą nesaugiais ryšio kanalais. Jos algoritmai paprastai naudoja du raktus – privatų, kuris yra saugomas paslaptyje ir viešą, kuris yra viešinamas. Deja, palyginus su simetrinio rakto kriptografija, asimetrinės kriptografijos algoritmai paprastai būna lėtesni, o patys raktai ilgesni [1].

Sparčiai augant ambicijoms ir žinioms susijusia tema, per palyginus trumpą laiką buvo aišku, kad pradinis baigiamojo darbo tikslas – ištirti asimetrinėmis savybėmis pasižyminčią kriptosistemą. Tam pasirinkome Difio ir Helmano apsikeitimo raktais protokolą (DH RAP). Šiek tiek vėliau suformulavome keletą konkrečių tikslų: Ištirti nagrinėjamo protokolo efektyvumą skirtingose grupėse. Taip pat jį įvykdžius matricų pusgrupėje nustatyti matricos ir pirminio skaičiaus eilės sąryšį. Tolimesnei analizei pasirinktas protokolas yra asimetrinis kriptografinis primityvas, kuris naudojamas slaptam kriptografinių raktų apsikeitimui viešojo ryšio kanalais ir yra žinomas kaip vienas pirmųjų viešojo rakto protokolų. Šis Whitfieldo Difio ir Martino Helmano apsikeitimo raktais algoritmas viešai buvo pristatytas 1976 metais. 2002 metais Helmanas algoritmą pasiūlė pavadinti Difio-Helmano-Merklio raktų apsikeitimu dėl Ralpho Merklio indėlio į viešojo rakto kriptografiją, tačiau algoritmas jau buvo pagarsėjęs savo pirmuoju vardu ir pasiūlytas naujasis pavadinimas neprigijo [1, 2].

Trumpai pristatysime ataskaitos turinį. Analitinėje dalyje, pirmiausiai suformuluojami reikalavimai programinei įrangai ir atliekamiems tyrimams. Toliau aptariamos labiausiai paplitusios kriptografinės sistemos bei apsikeitimo raktais algoritmai. Plačiausiai nagrinėjamas apibendrintas DH apsikeitimo raktais protokolas. Pabrėžiama, jog algoritmo saugumas paremtas diskretinio logaritmo problema. Metodologinje dalyje pateikiamos svarbiausios sąvokos bendram Difio ir Helmano apsikeitimo raktais algoritmo veikimo supratimui. Sudaromas apibendrinto DH matematinis modelis. Antrojoje metodologinės darbo dalyje aiškinama, kaip buvo sukurta programa ir jos vartotojo vadovo sąsaja algoritmo sprendimui skirtingose grupėse. Jį realizuojant matricų pusgrupėse jo veikimo principas buvo paremtas ne generuojančio elemento kėlimu laipsniu, o generuojantį elementą atitinkačios matricos ir dviejų subjektų pasirinktų privačių slaptųjų matricų daugyba. Baigimojo darbo tiriamosios dalies

pradžioje, pateikiami algoritmo vidutinių vykdymo trukmių nustatymo eksperimentiniai skaičiavimai. Pagal oficialiai suformuluotus reikalavimus prie atitinkamo stiprumo saugumo lygmenų, sulyginome vidutines veikimo trukmes sveikųjų skaičių multiplikacinėse ir elpsinių kreivių plokštumos taškų grupėse. Dar vienas tyrimo tikslas buvo nustatyti, kaip matricos eilė priklauso nuo pirminio skaičiaus eilės, algoritmą vykdant matricų pusgrupėse. Galiausiai nustačius šią priklausomybę, ją suformulavome kaip dėsnį. Pabaigoje formuluojamos išvados, pateikiamos rekomendacijos dėl tolimesnio sukurtų modelių, algoritmų, metodų bei programinių priemonių tobulinimo ir tolimesnių tyrimų atlikimo.

1. ANALITINĖ DALIS

1.1 TECHNINĖS BEI PROGRAMINĖS ĮRANGOS APŽVALGA

Studijuojant technologinių sričių modulius KTU buvo naudojama šiuolaikinė matematikos programinė įranga, t. y. naudojami daugiausiai paplitę standartiniai programiniai įrankiai. Studentai yra apmokomi spręsti įvairių sričių problemas kompiuteriu. Tam naudojami populiariausi matematikos (PTC MathCad, Matlab), statistikos (SAS, SPSS) ir kiti paketai. Studijų pradžioje studentai yra supažindinami su programavimu – tai yra būtinas pagrindas tiek sprendžiant sudėtingas užduotis programavimu, tiek naudojantis tokiais paketais kaip SAS. Dažniausiai naudojamos programavimo aplinkos yra Microsoft Visual Studio, NetBeans ir Matlab (matematinis programavimas). Microsoft Excel yra universalus įrankis, su kuriuo galima atlikti įvairius skaičiavimus. Dažnai naudojamas matematikos, informatikos, ekonomikos ir panašiose studijų programose. Ataskaitų rengimui naudojama Microsoft Word, pateikčių rengimui – Microsoft PowerPoint. Baigiamojo darbo metu panaudota programinė įranga ir jos trumpi aprašymai pateikti **1.1 lentelėje**.

Toliau pateiksime detalesnį pasirinktos programinės įrangos pagrindimą. PTC MathCad naudojome didelių matricos eilių determinantams skaičiuoti. Tokiu būdu galėjome greitai ir efektyviai patikrinti matricų pusgrupėse parinktų matricų reikalavimus. Taip pat ši programinė įranga padėjo pasitikrinimui atliekant didelės eilės matricų daugybai. OpenSSL programinės įrangos pagalba sugeneravome labai didelių eilių saugius pirminius skaičius, kurie buvo būtini norint sulyginti skaičiavimus sveikųjų skaičių multiplikacinėse ir elipsinių kreivių plokštumos taškų grupėse. Ši programinė įranga yra labai patogi darbui su kriptografinėmis sistemomis ir apsikeitimo raktais algoritmais, o parametrų generavimas atliekamas labai efektyviai lyginant su kitais panašios paskirties įrankiais. Pagrindinę programos realizacija atilikome Microsoft Visual Studio programinėje aplinkoje. Ši programa dažniausiai naudojama kuriant programas Windows sistemai, todėl DH algoritmas įvairiose grupėse buvo įgyvendintas Windows Forms pagalba.

1.1 lentelė. Programinė įranga ir jos aprašymai

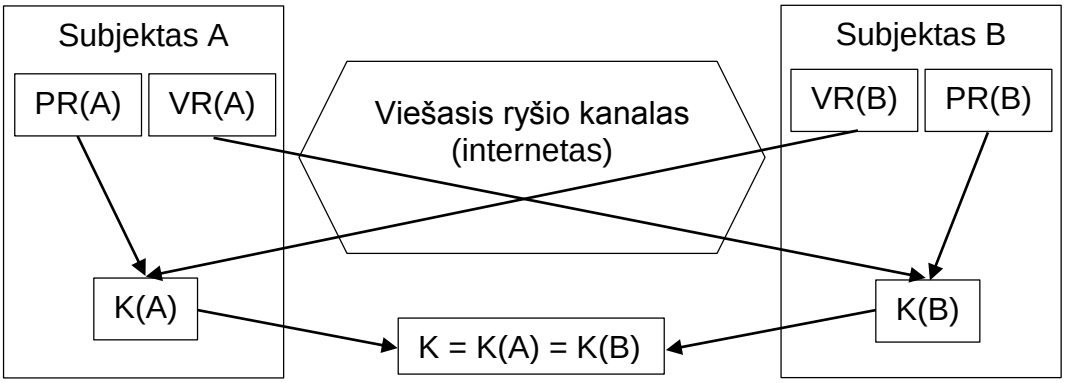
Programos logotipas	Programa	Aprašymas
	Microsoft Word	Teksto redagavimo programa, Microsoft Office paketo dalis. Naudojama dokumentų, ataskaitų ruošimui ir peržiūrai.
	Microsoft Excel	Universali elektroninė skaičiuoklė. Microsoft Office paketo dalis. Skirta atlikti lengvo-vidutinio sudėtingumo skaičiavimus, dirbti su duomenų įrašais, grafiškai vaizduoti duomenis.
	Microsoft PowerPoint	Skaidrių (pateikčių) ruošimo programa. Microsoft Office paketo dalis. Naudojama paruošti pateiktis pristatymams ir paskaitoms.
	PTC MathCad	Matematinių skaičiavimų programa. Ji pasižymi funkcijų gausa, todėl yra universali. Galima atlikti ir sudėtingus skaičiavimus. Pagrindinis bruožas – lengvai suprantama vartotojo sąsaja.
	OpenSSL	Universali programa atlikti su kriptografija susijusius ir kitus matematinius skaičiavimus. Pasižymi greitu saugių parametrų generavimu bei raktais apsikeitimo algoritmų vykdymu.
	Microsoft Visual Studio	Integruota programavimo ir kūrimo aplinka. Dažniausiai naudojama kuriant programas, skirtas Windows sistemai. Labai plačiai naudojama įvairiems skaičiavimams atlikti.

1.2 APSIKEITIMO RAKTAIS ALGORITMŲ APŽVALGA IR JŲ VEIKIMO SCHEMŲ ANALIZĖ

Šiame poskyryje apžvelgsime dviejų subjektų keitimosi raktais algoritmus. Apsikeitimo raktais algoritmas – tai kriptografinis metodas, kurio dėka du ar daugiau subjektų apsikeisdami kriptografiniais raktais gali susitarti dėl bendros paslapties. Simetrinio rakto kriptosistemai naudojat kažkokį slaptą raktą ar slaptažodį, kenkėjai dažniausiai naudoja įvairius metodus ir nelegalias paieškas šiai slaptai informacijai perimti. Apsisaugojimui nuo tokių ir kitų panašių atakų naudojama viešojo rakto kriptografija, o dar tiksliau – asimetriniai apsikeitimo raktais algoritmai. Tai galinga asimetrinės kriptografijos forma, panaikinanti simetrinės kriptografijos trūkumus bei šifravimui skirto rakto persiuntimą nesaugiais ryšio kanalais. Viešojo rakto apsikeitimo raktais algoritmai dažniausiai naudoja du pagrindinius raktus – privatų, kuris yra slaptai sugalvotas individualaus subjekto ir viešasis raktas, kuris yra prieinamas visiems, įskaitant ir kenkėjus. Simetrinės šifravimo sistemos veikia efektyviai, jose

naudojami sąlygiškai trumpi raktai, tačiau dideliame vartotojų tinkle raktus administruoti yra sudėtinga. Tuo tarpu asimetrinės šifravimo sistemos veikia lėtai, naudoja ilgus raktus, tačiau jie paskirstomi kurkas paprasčiau. Taigi, norime ištirti algoritmus bei sistemas, kurios pasižymėtų visomis geriausiomis simetrinių ir asimetrinių šifrų savybėmis.

Dviejų subjektų RAP principas pateiktas **1.1 pav.** Schemoje sutrumpintai pažymėsime kiekvieno vartotojo turimą porą raktų – viešąjį (VR) ir slaptąjį (PR). Viešieji raktai nelaikomi paslaptyje, ir kiekvienas vartotojas, naudodamas savo slaptąjį ir kito vartotojo viešąjį raktus, apskaičiuoja bendrą paslaptį. RAP sukurtas taip, kad abu vartotojai, naudodami tokį pat algoritmą ir skirtingus įvesties duomenis, gautų vienodus išvesties rezultatus. Paveiksle pateiktos rodyklės išvedamos iš viešųjų raktų simbolizuoja jų siuntimą viešuoju ryšio kanalu, o rodyklės prasidedančios privačiųjų raktų srityse nurodo subjektų individualiųjų slaptųjų raktų pasirinkimą. Įvykdžius apsikeitimo raktais algoritmą gaunama bendroji paslaptis, kurią vaizduoja apatinėje schemos dalyje susijungiančios rodyklės [2].



1.1 pav. RAP principas [2]

Toliau apžvelgsime konkrečius atvejus – populiariausią kriptosistemą ir paplitusius apsikeitimo raktais algoritmus. Dažniausiai naudojama RSA kriptografinė sistema (**1.1 algoritmas**) bei DH (**1.2 algoritmas**), ECDH (**1.3 algoritmas**), MQV (**1.4 algoritmas**) algoritmai. Taip pat žinomi PSK ir SRP raktais apsikeitimo algoritmai.

RSA sistema buvo sukurta 1978 metais ir nuo to laiko ji tapo žinomiausia asimetrinės kriptografijos sistema. Naudojant šią sistemą, galima sukonstruoti šifravimo, e. parašo, rakto apsikeitimo sistemas. RSA sistema realizuojama laipsnio kėlimo ir liekanos moduliu n operacijomis, o jos teorinį pagrindą sudaro skaičių teorija. RSA sistemos saugumas pagrįstas skaičių faktorizacijos problemomis. Šios sistemos idėja – užšifravimui ir iššifravimui taikyti modulinės eksponentės veiksmus. RSA sistemos parametrai parenkami taip, kad užšifruojant ir iššifruojant duomenis iš esmės atliekamas tas pats veiksmas, skiriasi tik parametrai. Tarkime, kad tekstograma yra t. RSA kriptosistemos raktai $PR = d$, viešasis $VR = (n, e)$; čia $n = pq$, p ir q yra dideli pirminiai skaičiai. Užšifruojant atliekamas veiksmas $c = t^e \ (mod\ n)$, o iššifruojant – $t = c^d \ (mod\ n)$ [2].

1.1 algoritmas. RSA kriptosistema [2]

Raktų generavimo procedūra:

- Generuojami du dideli, maždaug vienodo dydžio pirminiai skaičiai p ir q.
- Apskaičiuojama jų sandauga $n = pq$ ir Oilerio funkcijos reikšmė $\varphi(n) = (p-1)(q-1)$.
- Atsitiktinai parenkamas toks e $(1 < e < \varphi(n))$, kad $(e, \varphi) = 1$ (realiatyviai pirminiai).
- Pasitelkus išplėstinį Euklido algoritmą, randamas toks d, kad $ed = 1\ (mod\ \varphi(n))$.
- Viešasis raktas $VR = (n, e)$ privatusis raktas $PR = d$.

Užšifravimo procedūra:

- Šalis A, norėdama nusiųsti pranešimą šaliai B, pasiima jo viešąjį raktą $VR_B = (n, e)$.
- Tekstograma atvaizduojama į skaičių (arba skaičių seką) t iš intervalo $[0, n-1]$.
- Apskaičiuojamas $c = t^e\ (mod\ n)$.
- Šifrograma c siunčiama šaliai B.

Iššifravimo procedūra: naudodamas savo privatųjį raktą $PR_B = d$, šalis B iš gautos šifrogramos c atkuria gautą pranešimą $t = c^d\ (mod\ n)$.

RSA (Rivest-Shamir-Adelman) sistemos saugumo lygis priklauso nuo pasirinktųjų viešųjų parametrų. Jei n yra pakankamai nedidelis, tai potencialus kenkėjas gali perrinkti visas įmanomas tekstogramų t reikšmes, kol ras atitinkančią šifrogramą c. Tai pat vienas iš galimų alternatyvių sprendimo būdų būtų elemento n išskaidymas dauginamaisiais [2].

DH (Diffie-Hellman) apsikeitimo raktais algoritmas – stipriai paplitęs apsikeitimo raktais algoritmas. Jis, kaip ir visi kiti raktais apsikeitimo algoritmai – dviems subjektams padeda susitarti dėl bendrojo slaptojo rakto. Išsamų jo aprašymą, apibendrinimą ir matematinius modelius pateiskime šį algoritmą nagrinėjant atskirai, metodologinėje dalyje, o jo preliminarus algoritmas pateikteiktas čia (**1.2 algoritmas**).

1.2 algoritmas. DH apsikeitimas raktais [2, 9]

- Dvi šalys A ir B viešais ryšio kanalais nori susitarti dėl bendro slapto rakto.
- Pirmiausiai susitariama dėl bendros baigtinės ciklinės grupės $\langle G; * \rangle$ $(|G| = n)$ ir jos generatoriaus g.
- A atsitiktinai pasirenka $x \in \mathbb{Z}_n^*$, apskaičiuoja $a = g^x$ ir siunčia rezultatą B.
- B atsitiktinai pasirenka $y \in \mathbb{Z}_n^*$, apskaičiuoja $b = g^y$ ir siunčia rezultatą A.
- A apskaičiuoja $b^x = (g^y)^x = g^{xy} = K$.
- B apskaičiuoja $a^y = (g^x)^y = g^{xy} = K$.

ECDH (Elliptic Curve Diffie-Hellman) apsikeitimo raktais algoritmas (**1.3 algoritmas**) yra anoniminis raktų susitarimo protokolas, kuris leidžia kiekvienam turinčiam viešojo ir privačiojo raktų porą elipsnių kreivių grupėje rasti bendrą paslaptį per nesaugų ryšio kanalą. Ši bendra paslaptis gali būti naudojama tiesiogiai, arba ji gali padėti išsivesti kitą raktą, kuris vėliau gali būti naudojamas užšifruoti senesnius veiksmus. Kitaip tariant, tai yra Difio ir Helmano protokolo varianto (**1.2 algoritmas**) panaudojimas elipsinių kreivių plokštumos taškų grupėje. Toliau pateiksime ECDH algoritmo struktūrą.

1.3 algoritmas. ECDH apsikeitimas raktais [3]

- Dvi šalys A ir B viešais ryšio kanalais nori susitarti dėl bendro slapto rakto elipsinių kreivių plokštumos taškų grupėje.
- Susitariama dėl bendros baigtinės ciklinės elipsinių kreivių plokštumos taškų grupės $E(p_a, p_b, \mathbb{Z}_n^*)$ ir jos generatoriaus $g = (g_1, g_2)$.
- A atsitiktinai pasirenka $x \in \mathbb{Z}_n^*$, apskaičiuoja $a = (a_1, a_2) = (g_1, g_2)^x = g^x$ ir siunčia jį B.
- B atsitiktinai pasirenka $y \in \mathbb{Z}_n^*$, apskaičiuoja $b = (b_1, b_2) = (g_1, g_2)^y = g^y$ ir siunčia jį A.
- A apskaičiuoja $b^x = (b_1, b_2)^x = ((g_1, g_2)^y)^x = (g_1, g_2)^{xy} = (K_{A1}, K_{A2}) = K$.
- B apskaičiuoja $a^y = (a_1, a_2)^y = ((g_1, g_2)^x)^y = (g_1, g_2)^{xy} = (K_{B1}, K_{B2}) = K$.

MQV (Menezes-Qu-Vanstone) yra patvirtintas protokolas bendrojo slaptojo rankto susitarimui, remiantis DH algoritmo schema. Šis algoritmas pirmą kartą buvo pristatytas Menezes, Qu ir Vanstone 1995 metais. Kaip ir kitos autentifikuotos Difio ir Helmano schemos, MQV užtikrina apsaugą nuo įvairių kenkėjų. Šis protokolas gali būti modifikuotas darbui baigtinėse ciklinėse grupėse arba darbui elipsinių kreivių grupėse. Toliau pateikta MQV algoritmo veikimo schema (**1.4 algoritmas**).

1.4 algoritmas. MQV apsikeitimas raktais [4]

Šalis A turi raktų porą (A, a), kur A yra viešasis raktas ir a – privatusis raktas. Šalis B turi raktų porą (B, b), kur B yra viešasis raktas ir b – privatusis raktas. Tegul $R = (x, y)$ – taškas ant elipsinės kreivės. Tada $\bar{R} = (x \bmod 2^L) + 2^L$, kur $L = \left\lceil \frac{\lfloor \log_2 n \rfloor + 1}{2} \right\rceil$ ir n yra panaudoto generuojančio taško P tvarka.

- Dvi šalys A ir B viešais ryšio kanalais nori susitarti dėl bendro slapto rakto.
- Šalis A sugeneruoja porą (X, x) atsitiktinai generuodama x ir skaičiuodama $X = x \cdot P$ su tašku P, esančiu ant elipsinės kreivės.
- Šalis B sugeneruoja porą (Y, y) atsitiktinai generuodama y ir skaičiuodama $Y = y \cdot P$ su tašku P, esančiu ant elipsinės kreivės.
- Šalis A apskaičiuoja $S_a = x + \bar{X}a$ ir rezultatą nusiunčia šaliai B.

- Šalis B apskaičiuoja $S_b = y + \bar{Y}b$ ir rezultatą nusiunčia šaliai A.
- A apskaičiuoja $h \cdot S_a(Y + \bar{Y}B) = K$, kai h yra kofaktorius.
- B apskaičiuoja $h \cdot S_b(X + \bar{X}A) = K$, kai h yra kofaktorius.

Taip pat paminėsime, kad egzistuoja ir kiti PSK (Pre-Shared Key) ir SRP (Secure Remote Password) – mažiau paplitę apsikeitimo raktais algoritmai. Jie nėra tokie populiarūs, kaip anksčiau minėtieji. Šių algoritmų strukūrų darbe nedetalizuosime. Jie yra naudojami tik tam tikrais atskirais atvejais.

1.3 DISKRETINIO LOGARITMO PROBLEMA

Matematikoje, diskretinio logaritmo problema yra vadinamas lygties $a = g^x$ sprendimas siekiant rasti sveikąjį skaičių x, kai a ir g yra elementai iš tos pačios baigtinės grupės $\langle G; * \rangle$. Diskretinio logaritmo problemos sprendimas yra laikomas sudėtingu uždaviniu. Tokiam uždaviniui išspręsti nėra žinoma jokio efektyvaus bendrojo metodo įprastų kompiuterių pagalba ir būtent šios problemos sprendimo sudėtingumu yra paremtas daugelio svarbių viešojo rakto (asimetrinės) kriptografijos algoritmų saugumas [5].

Pateiksime trumpą pavyzdį. Diskretinio logaritmo problemos sudėtingumą lengviausiai suprasti sveikųjų skaičių multiplikacinėse grupėse $\langle \mathbb{Z}_p^*; \cdot \rangle$. Čia atliekama daugybos operacija pirminio skaičiaus p moduliu, t. y. jei norime suskaičiuoti kažkurio, šiai grupei priklausančio sveikojo skačiaus x-ąjį laipsnį, vadinasi sveikąjį skaičių pakėlę laipsniu x, jį turėsime padalinti iš p, o gauto rezultato liekana – elemento reikšmė šioje grupėje. Tarkime, kad turime grupę $\langle \mathbb{Z}_{17}^*; \cdot \rangle$. Skaičiuodami $3^4 = 81$, gautą skaičių 81 padalinus iš 17, liekana 13. Matematiškai šį veiksmą galime užrašyti taip: $3^4 \ (mod\ 17) = 13$. Diskretinis logaritmas skaičiuojamas atvirkšiai. Pavyzdžiui, diskretinio logaritmo problemos atveju, turime lygtį $3^x \ (mod\ 17) \equiv 13$. Vienas iš lygties sprendinių $x = 4$, tačiau tai nėra vienintelis sprendinys. Kadangi $3^{16} \equiv 1 \ (mod\ 17)$, tai $3^{4+16n} \equiv 3^4 \cdot (3^{16})^n \equiv 13 \cdot 1^n \ (mod\ 17)$. Taigi, lygtis turi be galo daug $4 + 16n$ formos sprendinių.

Bendrai kalbant apie diskretinio logaritmo problemą – ją galima spręsti bet kokioje baigtinėje ciklinėje grupėje, o elementai a ir g, tenkinantys lygtį $a = g^x$, yra elementai iš šios grupės. Tada betkuris lygtį tenkinantis sveikasis skaičius x bus vadinamas a diskretiniu logaritmu su pagrindu g. Matematiškai tai galime užrašyti šitaip: $x = \log_g a$. Priklausomai nuo g ir a yra įmanoma, kad diskretinis algoritmas neegzistuoja, arba egzistuoja daugiau negu vienas [5].

1.4 DARBE SPRENDŽIAMI UŽDAVINIAI

Patikslinsime baigiamojo darbo metu sprendžiamus uždavinius. Baigiamojo darbo tema – Difio ir Helmano raktais apsikeitimo algoritmo analizė skirtingose grupėse. Jau baigiamosios praktikos metu tyrėme Difio ir Helmano raktais apsikeitimo algoritmo greitį, jo veikimo principą skirtingose grupėse. Tai buvo kaip įvadas į šį darbą, kurio pagrindinius uždavinius suskirstėme į kelias dalis:

1. Susipažinti su apsikeitimo raktais algoritmais:
    - atlikti apsikeitimo raktais algoritmų apžvalgą,
    - atlikti apsikeitimo raktais algoritmų formuluočių analizę,
    - sudaryti skirtingų apsikeitimo raktais algoritmų apibendrintus paaiškinimus.
2. Susipažinti su Difio ir Helmano apsikeitimo raktais algoritmu:
    - atlikti algoritmo formuluotės analizę,
    - atlikti išsamų algoritmo aprašymą ir jo apibendrinimą,
    - sudaryti modifikuoto algorimto aprašymą, jo atlikimui matricų pusgrupėse.
3. Sudaryti Difio ir Helmano apsikeitimo raktais algoritmą personaliniame kompiuteryje skaičiuojant vidutinę algorimo vykdymo trukmę:
    - sudaryti programų aprašymus ir vartotojo vadovą,
    - atlikti eksperimentinius skaičiavimus,
    - pateikti testavimo rezultatus.
4. Patikrinti Difio ir Helmano apsikeitimo raktais algoritmo vidutines veikimo trukmes skirtingose grupėse:
    - apskaičiuoti vidutinę algoritmo veikimo trukmę sveikųjų skaičių multiplikacinėse grupėse,
    - apskaičiuoti vidutinę algoritmo veikimo trukmę elipsinių kreivių plokštumos taškų grupėse,
    - apskaičiuoti vidutinę algoritmo veikimo trukmę matricų pusgrupėse,
    - atlikti analizę ir pateikti vidutinių algoritmo veikimo trukmių suvestinę prie skiringų parametrų bei palyginti skaičiavimus skirtingose grupėse.
5. Suformuluoti matricos eilės priklausomybę nuo pirminio skaičiaus eilės apibūdinantį dėsnį.
    - nustatyti, kuriose pusgrupėse prie fiksuotų saugumo lygmenų algoritmas veikia efektyviausiai.
    - tikrinti vidutines veikimo trukmes matricų pusgrupėse, keičiant pirminio skaičiaus ir matricos eiles.
    - Aprašyti matricos eilės priklausomybę nuo pirminio skaičiaus eilės apibūdinantį dėsnį.

2. METODOLOGINĖ DALIS

2.1 PAGRINDINĖS SĄVOKOS, APIBRĖŽIMAI IR TEOREMOS

Norint suprasti apsikeitimo raktais algoritmų veikimo principą, būtina žinoti bei suprasti juose naudojamas pagrindines sąvokas, apibrėžimus ir teoremas, kuriuos pateiksime šiame poskyryje.

2.1 Apibrėžimas. ***Pusgrupe*** yra vadinama netuščia elementų aibė A su apibrėžta dvinare operacija *, kad $A \times A \to A$, t. y. bet kuriai elementų $a, b \in A$ porai (a, b) priskiriamas tos pačios aibės elementas: $(a, b) \to a * b \in A$ ir tenkinamos aksiomos:

a) aibė A yra uždara operacijos * atžvilgiu;
b) operacija * yra asociatyvioji su visais $a, b, c \in G$, t. y. $(a * b) * c = a * (b * c)$ [2].

2.2 Apibrėžimas. ***Grupe*** yra vadinama netuščia elementų aibė G su tokia apibrėžta dvinare operacija *, kad $G \times G \to G$, t. y. bet kuriai elementų $a, b \in G$ porai (a, b) priskiriamas tos pačios aibės elementas: $(a, b) \to a * b \in G$ ir tenkinamos tokios aksiomos:

a) aibė G yra uždara operacijos * atžvilgiu;
b) operacija * yra asociatyvioji su visais $a, b, c \in G$, t. y. $(a * b) * c = a * (b * c)$;
c) egzistuoja toks vienintelis neutralusis operacijos * elementas e, kad su visais $a \in G$ teisinga tokia lygybė: $a * e = e * a = a$;
d) kiekvienam $a \in G$ egzistuoja toks atvirkštinis (simetriškasis) elementas $a^{-1} \in G$, kad teisinga tokia lygybė: $a * a^{-1} = a^{-1} * a = e$ [2].

Grupę žymėsime simboliu $\langle G; * \rangle$. Šios grupės elementų skaičių žymėsime $|G|$. Jis gali būti baigtinis arba begalinis. Jeigu grupėje $\langle G; * \rangle$ reikalaujame, kad būtų tenkinama komutatyvumo aksioma $b * a = a * b$, tuomet sakome, kad grupė yra komutatyvioji (Abelio) grupė [2].

Dažnai operacija * grupėse žymėti vartosime daugybos ženklą, t. y. rašysime $a \cdot b$ vietoj $a * b$ ir vadinsime ***multiplikacine***, t. y. $\langle G; \cdot \rangle$. Jeigu vartosime sudėties ženklą, $\langle G; + \rangle$, tuomet grupę vadinsime ***adicine*** [2].

2.3 Apibrėžimas. Grupė $\langle G; * \rangle$ vadinama ***baigtine*** n-tos eilės, kai aibė G yra baigtinė ir $|G| = n$ [2].

2.4 Apibrėžimas. Grupė $\langle G; * \rangle$ vadinama ***cikline***, jeigu joje yra toks elementas g (generatorius), kad bet kuris kitas grupės elementas $x \in G$ užrašomas elemento g laipsniu, t. y. $\forall x \in G: \exists k \in \mathbb{Z}: x = g^k$ [2].

2.5 Apibrėžimas. Skaičius p vadinamas ***pirminiu***, jei jis yra natūralusis skaičius, didesnis nei 1 bei dalinasi tik iš savęs ir vieneto. Vienetas nelaikomas nei pirminiu, nei sudėtiniu skaičiumi [2].

2.6 Apibrėžimas. Pirminis skaičius p vadinamas ***saugiu***, jeigu egzistuoja toks pirminis skaičius q, kad teisinga lygybė: $p = 2q + 1$ [2].

2.7 Apibrėžimas. Sveikųjų skaičių multiplikacinė grupė $\langle \mathbb{Z}_p^*; \cdot \rangle$ ir jos generatoriai. $\langle \mathbb{Z}_p^*; \cdot \rangle$ savybės:

a) $\forall x, y \in \mathbb{Z}_p^*: x \cdot y \in \mathbb{Z}_p$

b) $\forall x, y, z \in \mathbb{Z}_p^*: (x \cdot y) \cdot z = (x \cdot y \cdot z)\ (mod\ p) = x \cdot (y \cdot z)$

c) $e = 1 \in \mathbb{Z}_p^*: x \cdot 1 = 1 \cdot x = x$

d) $\forall x \in \mathbb{Z}_p^*: \exists \tilde{x} =\ x^{p-2}\ (mod\ p) \in \mathbb{Z}_p^*$

e) $\forall x, y \in \mathbb{Z}_p^*: x \cdot y = x \cdot y\ (mod\ p) = y \cdot x\ (mod\ p) = y \cdot x$

Grupė $\mathbb{Z}_p^*$, kai p – pirminis, yra ciklinė. Simbolis * šiuo atveju reiškia, kad į grupės elementus neįeina „0“, o operacija „·“ – daugyba moduliu p. Norint rasti generatorių g, kai duotas p, bendruoju atveju sudėtinga. Net patikrinti, ar g duotajam p yra generatorius, problematiška. Tačiau jei žinoma, kad $p = 2q + 1$, o q – pirminis skaičius, tuomet rasti generatorių lengviau. Priminsime, kad $|\mathbb{Z}_p^*| = p - 1$, tada $|\mathbb{Z}_p^*| = 2q$ [2].

2.1 Teorema. Jeigu $p = 2q + 1$ yra pirminis ir q – pirminis, tuomet $g \in \mathbb{Z}_p^*$ yra generatorius tada ir tik tada, kai $g^q \neq 1$ ir $g^2 \neq 1$.

Kitaip tariant, kai q yra žinomas, paprasta patikrinti, ar g yra generatorius. Remdamiesi šia teorema patikrinę skirtingus elementus g tikriname, ar tai nėra generatorius [2].

2.2 Teorema. Jeigu g yra generatorius ir i nesidalija iš q bei 2, tuomet g^i yra generatorius [2].

2.8 Apibrėžimas. Elipsinės kreivės taškų grupė. ***Elipsinė kreivė*** virš pirminės eilės p baigtinio lauko – tai plokštumos taškų, tenkinančių lygtį $y^2 = x^3 + ax + b\ (mod\ p)$, aibė:

a) veiksmai yra atliekami lauke $\langle GF(p); +; \cdot \rangle$, p – nelyginis;

b) $a, b \in GF(p)$;

c) reikalaujama, kad $4a^3 + 27b^2 \neq 0$;

d) dažniausiai žymima $E(a, b, GF(p))$;

e) į šią taškų aibę papildomai įvedamas be galo nutoęs taškas *O*;

f) be galo nutolęs taškas *O* yra neutralusis elementas ir bet kuriam elipsinės kreivės taškui yra teisinga lygybė $P + O = P$;

g) bendru atveju, dviejų elipsinės kreivės taškų $P(x_1, y_1) + Q(x_2, y_2) = R(x_3, y_3)$:

$$x_3 = \lambda^2 - x_1 - x_2 \text{ ir } y_3 = \lambda(x_1 - x_3) - y_1,\ \lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1},\ kai\ P \neq Q \\ \frac{3x_1^2 + a}{2y_1},\ kai\ P = Q \end{cases}$$ [2].

2.2 DIFIO IR HELMANO APSIKEITIMO RAKTAIS PROTOKOLAS

Difio ir Helmano (DH) sistema yra viešojo rakto kriptografinis protokolas, kurio paskirtis – dviem ar daugiau vartotojų sukurti bendrąjį raktą. Vykdant DH algoritmą, vartotojai nei siunčia, nei šifruoja bendrojo slaptojo rakto. DH algoritmas tokio rakto ir negeneruoja – jis tiesiog padeda

vartotojams savarankiškai apskaičiuoti bendrą slaptąjį raktą ir užtikrina, kad tai padarius raktai sutaps.

Algoritmo saugumas paremtas tuo, kad sveikųjų skaičių multiplikacinėse arba elipsinių kreivių plokštumos taškų grupėse žinant elementus a ir g, rasti atitinkamą x su kuriuo $a = g^x$ yra sudėtingas uždavinys. Tai vadinama diskretinio logaritmo problema, kurią jau aptarėme analitinėje darbo dalyje. Paprasčiausias sprendimo būdas – visų variantų perrinkimas. Difio ir Helmano apsikeitimo raktais algorimo pradinei realizacijai buvo siūloma naudoti sveikųjų skaičių multiplikacinę grupę $\langle \mathbb{Z}_p^*; \cdot \rangle$ [2]. Dabar šioje grupėje minimalus rekomenduojamas pirminio skaičiaus p dydis yra 2048 bitai [6]. Tokio skaičiaus pavyzdys pateiktas žemiau (**2.1 pav.**). Vėliau algoritmas pradėtas vykdyti ir elipsinių kreivių plokštumos taškų grupėse, kuriose rekomenduojamas pirminio skaičiaus p dydis yra žymiai mažesnis dėl atliekamų skaičiavimų sudėtingumo. Čia tokio saugumo užtikrinimui rekomenduojami tik 224 bitai (**2.2 pav.**) [6]. Pateiktų pirminių skaičių pavyzdžių tikslas – parodyti, kaip stipriai skiriantis parametrų dydžiui užtikrinamas toks pats saugumas. Nepaisant nustatytų minimalių saugumo reikalavimų, DH algoritmą realizuodami kompiuteryje, sveikųjų skaičių multiplikacinėse grupėse jį sukūrėme taip, kad galėtume vykdyti su parametrais iki 16384 bitų, o elipsinėse kreivėse 521 bito. Toks algoritmo saugumas sveikųjų skaičių grupėse minimalius reikalavimus viršija 8 kartus, o elipsinėse kreivėse – 7,5 karto. Šių ir kitų darbe naudotų, saugių pirminių skaičių p pavyzdžius galite rasti 2 priede.

```
25195908475657893494027183240048398571429282126204032027777137836043662020707595556264018525880784406918290641249515082189298559149176184502808489120072844992687392807287776735971418347270261896375014971824691165077613379859095700097330459748808428401797429100642458691817195118746121515172654632282216869987549182422433637259085141865462043576798423387184774447920739934236584823824281198163815010674810451660377306056201619676256133844143603833904414952634432190114657544454178424020924616515723350778707749817125772467962926386356373289912154831438167899885040445364023527381951378636564391212010397122822120720357
```

2.1 pav. 2048 bitų pirminis skaičius [6].

```
26959946667150639794667015087019630673557916260026308143510066298881
```

2.2 pav. 224 bitų pirminis skaičius [6].

2.1 algoritmas. DH algoritmas sveikųjų skaičių multiplikacinėse grupėse

1. Alisa ir Bobas viešai susitaria dėl bendrų sistemos parametrų – didelio pirminio skaičiaus p ir grupės $\mathbb{Z}_p^*$ generatoriaus g.
2. Alisa atsitiktinai pasirenka savo slaptąjį raktą $PR_A = x$, apskaičiuoja savo viešąjį raktą $VR_A = a = g^x \ (mod\ p)$ ir viešąjį VR_A nusiunčia Bobui.
3. Bobas atsitiktinai pasirenka savo slaptąjį raktą $PR_B = y$, apskaičiuoja savo viešąjį raktą $VR_B = b = g^y \ (mod\ p)$ ir viešąjį VR_B nusiunčia Alisai.

4. Alisa, turėdama savo slaptąjį raktą $PR_A = x$ ir Bobo viešąjį raktą $VR_B = g^y \ (mod\ p)$ apskaičiuoja bendrą raktą $K_A = (VR_B)^x \ (mod\ p) = g^{xy} \ (mod\ p)$.
5. Bobas turėdamas savo slaptąjį raktą $PR_B = y$ ir Alisos viešąjį raktą $VR_A = g^x \ (mod\ p)$ apskaičiuoja bendrą raktą $K_B = (VR_A)^y \ (mod\ p) = g^{yx} \ (mod\ p)$.
6. Tokiu būdu abu vartotojai apskaičiuoja slaptąjį raktą K, nes $g^{xy} \ (mod\ p) = g^{yx} \ (mod\ p) = K$.

DH algorimą iliustruosime pavyzdžiu **2.1 lentelėje**. Esant tokiai sistemai, potencialus kenkėjas Zita (Z), stebėdama siunčiamus pranešimus, negali atlikti tų pačių veiksmų kaip Alisa ir Bobas. Z žino tik viešuosius raktus VR_A ir VR_B. Turėdama tik juos, ji negali apskaičiuoti bendro slaptojo rakto K, Z turi žinoti kurį nors slaptąjį PR_A arba PR_B. Tačiau iš viešojo rakto gauti slaptąjį neįmanoma, nes reikia išspręsti diskretinio logaritmo problemą. Tokiu būdu DH yra pagrįstas modulinės eksponentės funkcija. Abu vartotojai ne tik apskaičiuoja bendrąjį slaptąjį raktą, bet ir niekas kitas negali jo apskaičiuoti.

2.1 lentelė. DH pavyzdys sveikųjų skaičių multiplikacinėje grupėse

Žingsnio numeris	Alisos veiksmai	Bobo veiksmai
1.	viešai susitaria $p = 113, g = 37$;	
2.	pasirenka $PR_A = 65$; apskaičiuoja $VR_A = 37^{65} \ (mod\ 113) = 19$; siunčia $VR_A = 19$;	
3.		pasirenka $PR_B = 79$; apskaičiuoja $VR_B = 37^{79} \ (mod\ 113) = 68$; siunčia $VR_B = 68$;
4.	gauna $VR_B = 68$; apskaičiuoja $K_A = 68^{65} (mod\ 113) = 89$;	
5.		gauna $VR_A = 19$; apskaičiuoja $K_B = 19^{79} (mod\ 113) = 89$;
6.	$K = K_A = K_B$.	

Apibendrinsime ECDH algoritmą, t. y. DH elipsinių kreivių plokštumos taškų grupėse (**2.2 algoritmas**). Primename, kad žemiau pateikta tik šio apibendrinto algoritmo schema, o patys skaičiavimai atliekami pagal elipsinių kreivių plokštumos taškų grupėms būdingas taisykles.

2.2 algoritmas. DH algoritmas elipsinių kreivių plokštumos taškų atveju

1. Alisa ir Bobas viešai susitaria dėl bendrų sistemos parametrų – didelio pirminio skaičiaus p baigtinės ciklinės elipsinių kreivių plokštumos taškų grupės $E(p_a, p_b, \mathbb{Z}_p^*)$ ir jos generatoriaus $g = (g_1, g_2)$.

2. Alisa atsitiktinai pasirenka savo slaptąjį raktą $PR_A = x \in \mathbb{Z}_p^*$, apskaičiuoja savo viešąjį apskaičiuoja $VR_A = a = (a_1, a_2) = (g_1, g_2)^x \ (mod\ p) = g^x \ (mod\ p)$ ir viešąjį VR_A nusiunčia Bobui.
3. Bobas atsitiktinai pasirenka savo slaptąjį raktą $PR_B = y \in \mathbb{Z}_p^*$, apskaičiuoja savo viešąjį apskaičiuoja $VR_B = b = (b_1, b_2) = (g_1, g_2)^y \ (mod\ p) = g^y \ (mod\ p)$ ir viešąjį VR_B nusiunčia Alisai.
4. Alisa, turėdama savo slaptąjį raktą $PR_A = x$ ir Bobo viešąjį raktą $VR_B = b = (b_1, b_2) = (g_1, g_2)^y \ (mod\ p) = g^y \ (mod\ p)$ apskaičiuoja bendrą raktą $K_A = (VR_B)^x \ (mod\ p) = b^x \ (mod\ p) = (b_1, b_2)^x \ (mod\ p) = (g_1, g_2)^{xy} \ (mod\ p) = (K_{A1}, K_{A2}) = K$.
5. Bobas, turėdamas savo slaptąjį raktą $PR_B = y$ ir Alisos viešąjį raktą $VR_A = a = (a_1, a_2) = (g_1, g_2)^x \ (mod\ p) = g^x \ (mod\ p)$ apskaičiuoja bendrą raktą $K_B = (VR_A)^y \ (mod\ p) = a^y \ (mod\ p) = (a_1, a_2)^y \ (mod\ p) = (g_1, g_2)^{yx} \ (mod\ p) = (K_{B1}, K_{B2}) = K$.
6. Tokiu būdu abu vartotojai apskaičiuoja slaptąjį raktą K, nes $(g_1, g_2)^{xy} \ (mod\ p) = (g_1, g_2)^{yx} \ (mod\ p) = K$.

Dabar DH algoritmą elipsinių kreivių plokštumos taškų grupių atveju iliustruosime pavyzdžiu, analogiškai, kaip tai padarėme ir svekųjų skaičių multiplikacinėmis grupėmis. Pavyzdys **2.2 lentelėje**.

2.2 lentelė. DH pavyzdys elipsinių kreivių plokštumos taškų grupėse

Žingsnio numeris	Alisos veiksmai	Bobo veiksmai
1.	viešai susitaria $p = 11,\ E(7, 9, \mathbb{Z}_{11}^*),\ g = (g_1, g_2) = (7, 7)$;	
2.	pasirenka $PR_A = 6$; apskaičiuoja $VR_A = (7, 7)^6 \ (mod\ 11) = (6, 6)$; siunčia $VR_A = (6, 6)$;	
3.		pasirenka $PR_B = 8$; apskaičiuoja $VR_B = (7, 7)^8 \ (mod\ 11) = (9, 8)$; siunčia $VR_B = (9, 8)$;
4.	gauna $VR_B = (9, 8)$; apskaičiuoja $K_A = (9, 8)^6 \ (mod\ 11) = (5, 2)$;	
5.		gauna $VR_A = (6, 6)$; apskaičiuoja $K_B = (6, 6)^8 \ (mod\ 11) = (5, 2)$;
6.	$K = K_A = K_B$.	

2.2.1 DIFIO IR HELMANO APSIKEITIMAS RAKTAIS MATRICŲ PUSGRUPĖSE

Difio ir Helmano apsikeitimo raktais algoritmą taip pat realizavome ir matricų pusgrupėse. Tokia algoritmo realizacija nėra standartinė ir buvo įgyvendinta atlikus tam tikras modifikacijas, todėl ji pateikiama atskirai. Pirmiausiai reikia žinoti pirminį skaičių bei matricos eilę. Matricos eilė nusako su kokio dydžio (eilučių ir stulpelių skaičius) kvadratinėmis matricomis vykdysime algoritmą. Šiuo atveju algoritmo veikimo principas yra paremtas ne kėlimo laipsniu operacija, bet matricų daugyba.

2.3 algoritmas. DH algoritmas matricų pusgrupėse

1. Alisa ir Bobas viešai susitaria dėl bendrų sistemos parametrų – didelio pirminio skaičiaus p, matricos eilės m ir iš matricų pusgrupės $\mathbb{M}_m(\mathbb{Z}_p^*)$ generatorių atitinkančios, bendros išsigimusios matricos G. Papildomai sutariama dauginamų matricų tvarka.
2. Alisa atsitiktinai pasirenka savo slaptąjį raktą atitinkančią privačiąją matricą $PR_A = X$, apskaičiuoja savo viešąjį raktą $VR_A = A = X \cdot G \ (mod\ p)$ ir viešąją VR_A nusiunčia Bobui.
3. Bobas atsitiktinai pasirenka savo slaptąjį raktą atitinkančią privačiąją matricą $PR_B = Y$, apskaičiuoja savo viešąjį raktą $VR_B = B = G \cdot Y \ (mod\ p)$ ir viešąją VR_B nusiunčia Alisai.
4. Alisa, turėdama savo slaptąjį raktą atitinkančią matricą $PR_A = X$ ir Bobo viešąją matricą $VR_B = B \ (mod\ p) = G \cdot Y \ (mod\ p)$, apskaičiuoja bendrąją matricą $K_A = X \cdot B \ (mod\ p) = X \cdot (G \cdot Y) \ (mod\ p)$.
5. Bobas, turėdamas savo slaptąjį raktą atitinkančią matricą $PR_B = Y$ ir Alisos viešąją matricą $VR_A = A \ (mod\ p) = X \cdot G \ (mod\ p)$, apskaičiuoja bendrąją matricą $K_B = A \cdot Y \ (mod\ p) = (X \cdot G) \cdot Y \ (mod\ p)$.
6. Tokiu būdu abu vartotojai apskaičiuoja slaptąjį raktą atitinkančią matricą K, nes $X \cdot (G \cdot Y) \ (mod\ p) = (X \cdot G) \cdot Y \ (mod\ p) = K$.

Tokioje modifikuotoje Difio ir Helmano apsikeitimo raktais algoritmo formoje saugumas nėra diskreitinio logaritmo problema. Saugumą tokiose pusgrupėse lemia matricos, atitinkačios generatorių, savybės. Būtina, kad ši matrica būtų išsigimusi, t. y. neegzistuotų jai atvirkštinė. Kitaip tariant, šios matricos determinantas privalo būti lygus nuliui. Tik išpildžius tokį reikalavimą, tolimesnis DH algoritmo veikimo principas įgija prasmę. Priešingu atveju, kenkėjai viešąsias matricas padauginę iš generatorių atitinkančiai atvirkštinės matricos galėtų suskaičiuoti šalies A ir B pasirinktas slaptąsias matricas, atitinkančias privačiuosius raktus bei tokiu būdu perimti slaptą informaciją. Matematiškai tokį informacijos perėmimą galėtume užrašyti šitaip: kai $PR_A \cdot G \ (mod\ p) = VR_A$, tuomet $PR_A = VR_A \cdot G^{-1}$. Aukščiau pateiktas DH algoritmas matricų pusgrupėse (**2.3 algoritmas**).

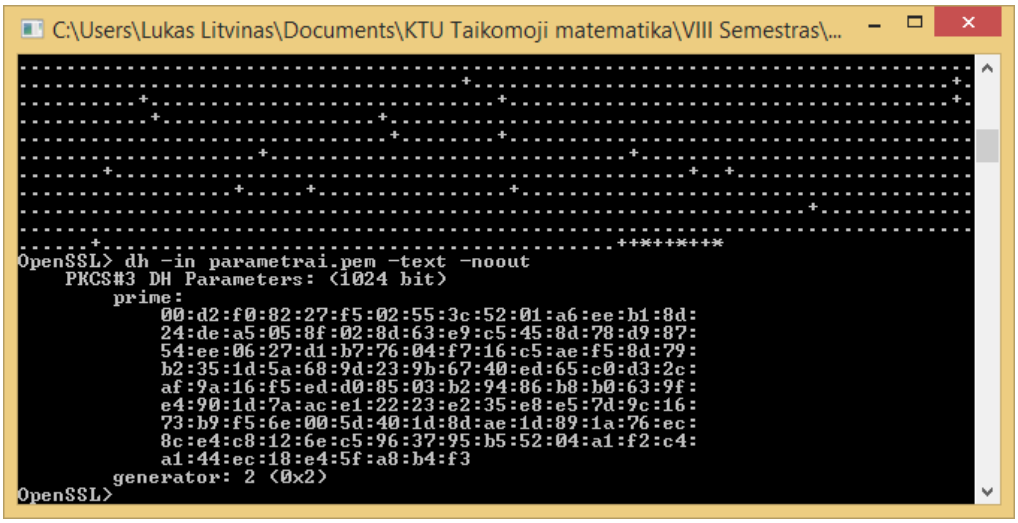
Primename, kad čia labai svarbi dauginamų matricų tvarka, nes matricų daugyba nėra komutatyvi operacija. DH algoritmą matricų pusgrupių atveju iliustruosime pavyzdžiu **2.3 lentelėje**.

2.3 lentelė. DH pavyzdys matricų pusgrupėse

Žingsnio numeris	Alisos veiksmai	Bobo veiksmai
1.	viešai susitaria $p = 7,\ \mathbb{M}_3(\mathbb{Z}_7^*),\ G = \begin{pmatrix} 4 & 6 & 4 \\ 2 & 3 & 2 \\ 1 & 4 & 2 \end{pmatrix}$, kai $\begin{vmatrix} 4 & 6 & 4 \\ 2 & 3 & 2 \\ 1 & 4 & 2 \end{vmatrix} = 0$	
2.	pasirenka $PR_A = \begin{pmatrix} 0 & 0 & 6 \\ 2 & 4 & 0 \\ 5 & 2 & 1 \end{pmatrix}$; $VR_A = \begin{pmatrix} 0 & 0 & 6 \\ 2 & 4 & 0 \\ 5 & 2 & 1 \end{pmatrix} \cdot \begin{pmatrix} 4 & 6 & 4 \\ 2 & 3 & 2 \\ 1 & 4 & 2 \end{pmatrix} (mod\ 7) = \begin{pmatrix} 6 & 3 & 5 \\ 2 & 3 & 2 \\ 4 & 5 & 5 \end{pmatrix}$; siunčia $VR_A = \begin{pmatrix} 6 & 3 & 5 \\ 2 & 3 & 2 \\ 4 & 5 & 5 \end{pmatrix}$;	
3.		pasirenka $PR_B = \begin{pmatrix} 6 & 0 & 0 \\ 5 & 5 & 1 \\ 2 & 3 & 1 \end{pmatrix}$; $VR_B = \begin{pmatrix} 4 & 6 & 4 \\ 2 & 3 & 2 \\ 1 & 4 & 2 \end{pmatrix} \cdot \begin{pmatrix} 6 & 0 & 0 \\ 5 & 5 & 1 \\ 2 & 3 & 1 \end{pmatrix} (mod\ 7) = \begin{pmatrix} 6 & 0 & 3 \\ 3 & 0 & 5 \\ 2 & 5 & 6 \end{pmatrix}$; siunčia $VR_B = \begin{pmatrix} 6 & 0 & 3 \\ 3 & 0 & 5 \\ 2 & 5 & 6 \end{pmatrix}$;
4.	gauna $VR_B = \begin{pmatrix} 6 & 0 & 3 \\ 3 & 0 & 5 \\ 2 & 5 & 6 \end{pmatrix}$; apskaičiuoja $K_A = \begin{pmatrix} 0 & 0 & 6 \\ 2 & 4 & 0 \\ 5 & 2 & 1 \end{pmatrix} \cdot \begin{pmatrix} 6 & 0 & 3 \\ 3 & 0 & 5 \\ 2 & 5 & 6 \end{pmatrix} (mod\ 7) = \begin{pmatrix} 5 & 2 & 1 \\ 3 & 0 & 5 \\ 3 & 5 & 3 \end{pmatrix}$;	
5.		gauna $VR_A = \begin{pmatrix} 6 & 3 & 5 \\ 2 & 3 & 2 \\ 4 & 5 & 5 \end{pmatrix}$; apskaičiuoja $K_B = \begin{pmatrix} 6 & 3 & 5 \\ 2 & 3 & 2 \\ 4 & 5 & 5 \end{pmatrix} \cdot \begin{pmatrix} 6 & 0 & 0 \\ 5 & 5 & 1 \\ 2 & 3 & 1 \end{pmatrix} (mod\ 7) = \begin{pmatrix} 5 & 2 & 1 \\ 3 & 0 & 5 \\ 3 & 5 & 3 \end{pmatrix}$;
6.	$K = K_A = K_B.$	

2.3 DIFIO IR HELMANO APSIKEITIMO RAKTAIS ALGORITMO REALIZACIJA PERSONALINIAME KOMPIUTERYJE

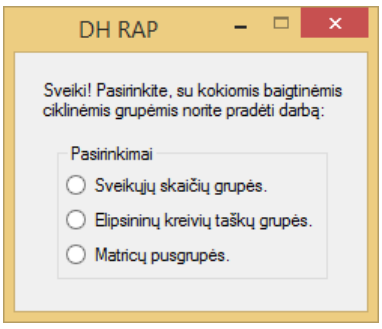
Difio ir Helmano apsikeitimo raktais algoritmui realizuoti naudojome Microsoft Visual Studio programavimo aplinką, o jo didelių pirminio skaičiaus eilių parametrams generuoti – OpenSSL (**2.3 pav.**). Parametrus prie mažesnių pirminio skaičiaus eilių greitai galime sugeneruoti ir Microsoft Visual Studio programinėje aplinkoje mūsų sukurta programa, tačiau prie labai didelių pirminio skaičiaus eilių, pavyzdžiui 8192 bitų, tai užtruktų ilgiau, todėl naudojome jau tam skirtą įrankį – OpenSSL, kuris tokį generavimą, manoma, kad atlieka greičiau. Sugeneruoti parametrai pateikiami šešioliktainiu formatu, kurių formatą Microsoft Visual Studio pagalba pakeitėme į mums reikiamą – dešimtainį.



2.3 pav. OpenSSL vaizdas sugeneravus 1024 bitų pirminį skaičių šešioliktainiu formatu

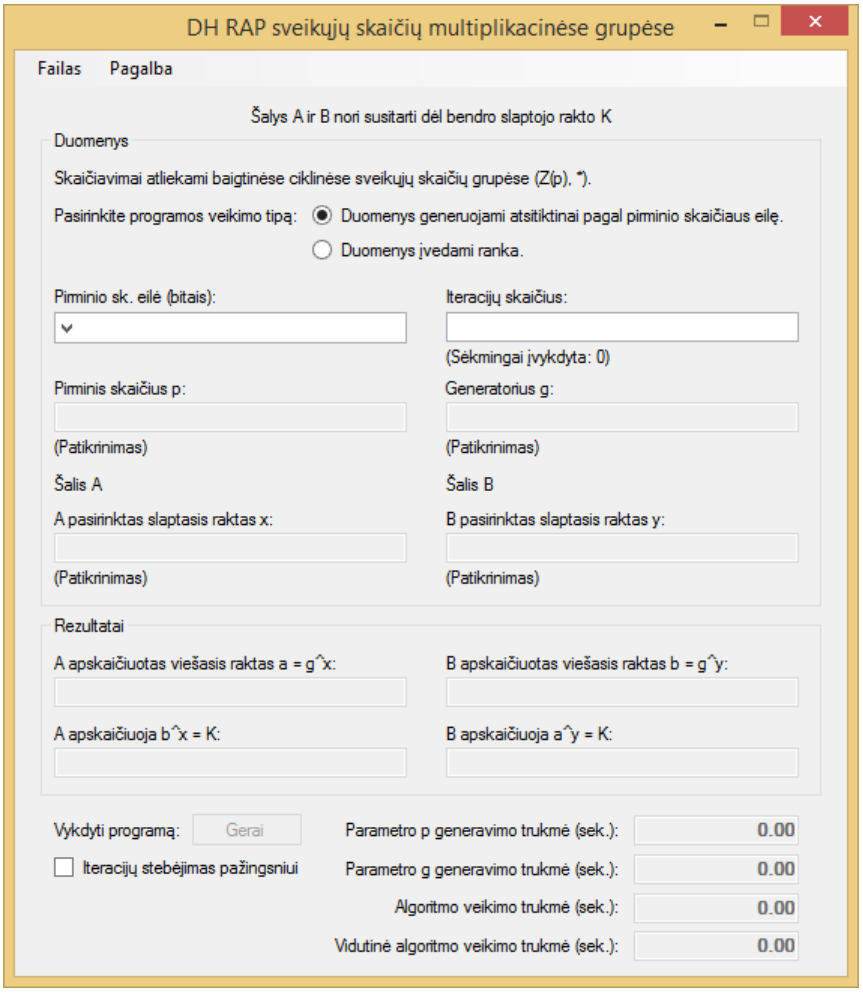
2.3.1 PROGRAMOS VARTOTOJO VADOVAS

Sukurta programa turi galimybę dirbti trijose skirtingų rūšių aplinkose, t. y. sveikųjų skaičių multiplikacinėse ir elipsinių kreivių plokštumos taškų grupėse bei matricų pusgrupėse. Šį pradinį pasirinkimą galima atlikti pirmą kartą, paleidus programą darbui, kai pasirodo pradinis langas (**2.4 pav.**).



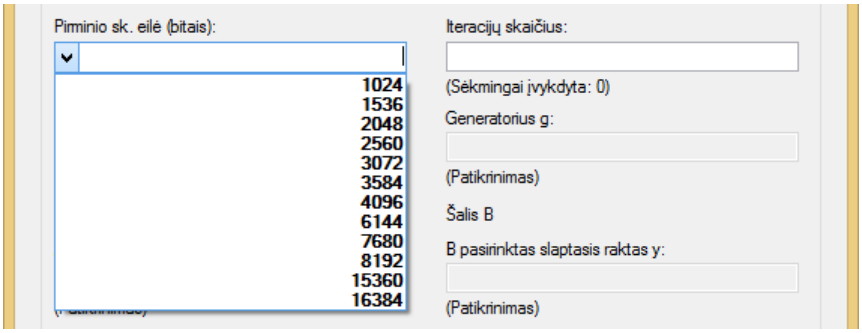
2.4 pav. Pradinis programos langas

Pirmiausiai pateiksime vartotojo vadovo aprašymą, skirtą darbui su sveikųjų skaičių multiplikacinių grupių dalimi. Šis programos langas pavaizduotas **2.5 pav.**, kuris atsiveria pasirinkus „Sveikųjų skaičių grupės.“ pradiniame vartotojo vadovo lange.



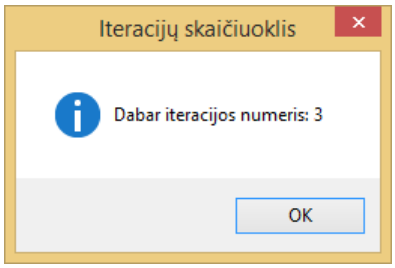
2.5 pav. Programos langas, skirtas darbui su sveikųjų skaičių grupėmis

Dirbti su sveikųjų skaičių multiplikacinėmis grupėmis galime dvejopai, pasirinkdami vieną iš galimų šios programos dalies veikimo tipų. Tik atsivėrus šiam langui, pirmiausiai būna parinktas pirmasis tipas „Duomenys generuojami atsitiktinai pagal pirminio skaičiaus eilę.“. Dirbant pasirinkus šį tipą konkrečių duomenų (pirminio skaičiaus p, generatoriaus g, šalies A slaptojo rakto x ir šalies B slaptojo rakto y) neįvedinėsime. Tokiu atveju, reikalavimus Difio ir Helmano algoritmui atitinkanys parametrai bus sugeneruojami atsitiktinai, pagal nurodytą pirminio skaičiaus eilę (bitais). Pirminio skaičiaus eilę galima įvesti ranka arba pasirinkti iš sąrašo. Sąraše pateiktos labai didelės eilės, kurių parametrai generuojami žymiai ilgiau, todėl patogumo dėlei 1024, 1536, ..., 16384 bitų parametrus galime pasirinkti čia, ir jų generavimo nelaukti (**2.6 pav.**).



2.6 pav. Pirminio skaičiaus eilės pasirinkimas iš sąrašo sveikųjų skaičių grupėje

Taip pat būtina nurodyti norimą atlikti iteracijų skaičių – jis parodo kiek kartų kartosime algoritmą, apribotą įvesta pirminio skaičiaus eile. Neįvedus bent vieno reikalavimo ar parametro programos vykdymas neįmanomas – vykdymo mygtukas „Gerai“ neaktyvus. Šalia vykdymo mygtuko yra suteikta galimybė pažymėti „Iteracijų stebėjimas pažingsniui“ (šis pasirinkimas nėra privalomas). Paleidus programą jis leis iškviesti iteracijų skaičiuoklį, stebėti kiekvienos iteracijos metu generuojamus duomenis, algoritmo trukmės kitimą bei praneš apie esamą iteracijos numerį (**2.7 pav.**). Kiekvienos iteracijos metu pirminis skaičius p ir generatorius g neperskaičiuojami. Keičiantis iteracijoms kinta tik šalies A slaptasis raktas x ir šalies B slaptasis raktas y. Programos vykdymo metu žemiau sugeneruotų parametrų atliekamas patikrinimas – išvedamas pranešimas žalia spalva jei parametrai buvo sugeneruoti sėkmingai, raudona – jei parametrai būtų sugeneruoti klaidingai (tačiau programa parametrus visada generuoja teisingai).



2.7 pav. Iteracijų skaičiuoklis

Rezultatų dalyje bus matomi šalies A ir šalies B suskaičiuoti viešieji raktai, atitinkamai a ir b. Taip pat, galėsime matyti tiek šalies A, tiek šalies B apskaičiuotą bendrąjį slaptąjį raktą K, kuris sutaps. Galiausiai, apatinėje šio vartotojo vadovo lango dalyje pateikiamos tiek parametrų generavimo, tiek algoritmo veikimo trukmės. Vidutinė trukmė pateikiama tik baigus visas iteracijas. Kaip atrodo programos rezultatų dalis ir trukmių matavimo laukeliai prieš jos vykdymą matoma **2.5 pav.**

Kitas galimas programos vykdymo tipas – kai konkretūs duomenys yra įvedami ranka. Tuomet renkamės programos veikimo tipą „Duomenys įvedami ranka“, t. y. priešingas pasirinkimas negu

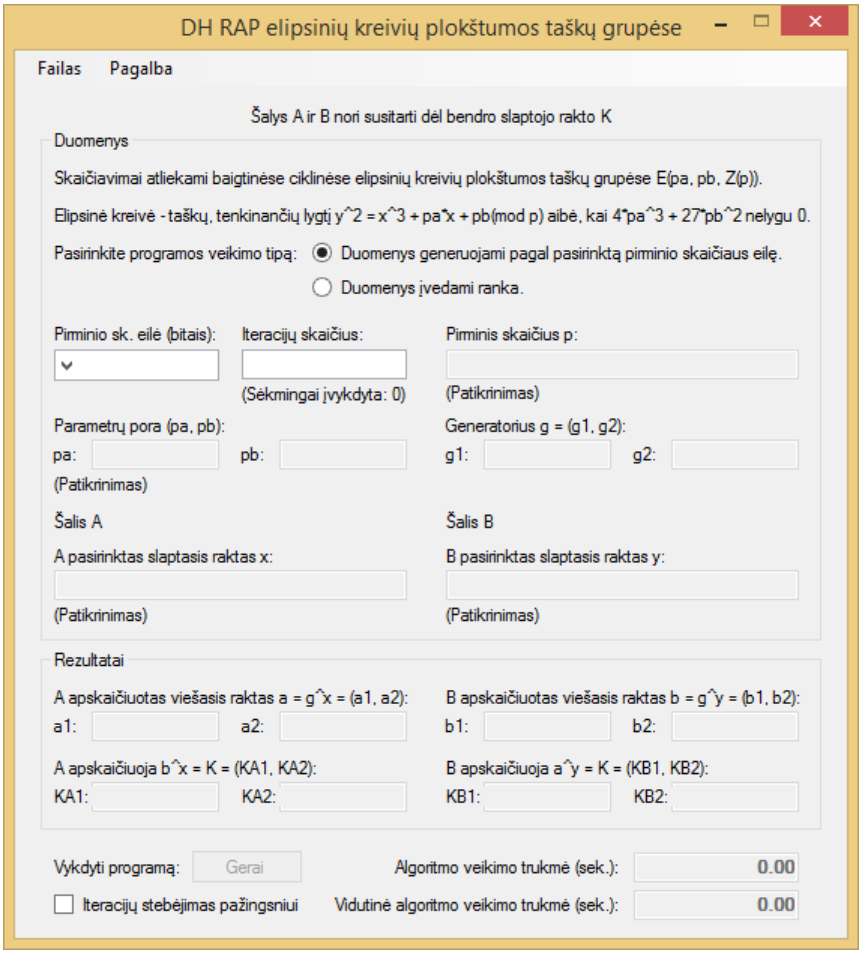
pavaizduota **2.5 pav.** Pasirinkus šį programos veikimo tipą suteikiama galimybė įvesti duomenis šalia parametro pavadinimo, o nereikalingi laukeliai apsaugomi nuo įvedimo. Tokiu atveju patys privalome būti užtikrinti, kad įvedami parametrai teisingai atitinka Difio ir Helmano apsikeitimo raktais algoritmo reikalavimus. Ar parametrus įvedėme teisingai pamatysime įvykdžius programą. Bus atliktas analogiškas patikrinimas kaip ir atsitiktinio parametrų generavimo atveju ir išvestas pranešimas apie įvesties teisingumą. Kadangi duomenys įvesti ranka – iteracija bus tik viena, vadinasi „Iteracijų stebėjimas pažingsniui“ yra apsaugotas nuo pažymėjimo taip pat. Parametrų p ir g generavimo trukmės lygios 0.00, nes jų negeneravome, taigi dėmesį kreipiame tik į algoritmo veikimo trukmes.

Nepriklausomai nuo pasirinkto programos veikimo tipo šio vartotojo vadovo lango viršutinėje dalyje visada matysime meniu juostą, kuri skirta padėti valdyti programą (**2.5 pav.**). Paspaudus mygtuką „Failas“ → „Naujas generavimas“ galime atlikti naują generavimą (visi laukeliai bus išvalyti ir paruošti naujam skaičiavimui). Atlikdami „Failas“ → „Keisti grupę“ mygtuko paspaudimą, galime pakeisti nagrinėjamos grupės rūšį į kitą. Šiuo atveju bus iš naujo iškviestas pradinis programos langas (**2.4 pav.**). Pasirinkdami „Failas“ → „Baigti“ baigsime darbą su programa. Atlikus mygtuko „Pagalba“ → „Apie“ paspaudimą bus pateikta informacija apie nagrinėjamą grupę bei nurodytas programos kūrėjas.

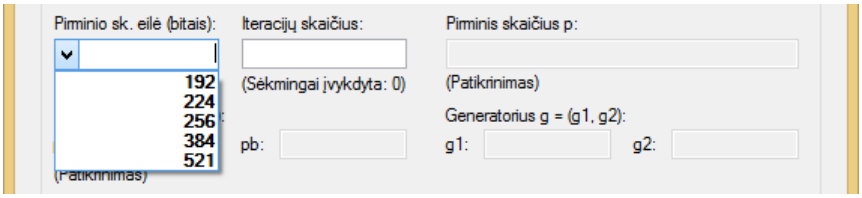
Toliau pateiksime programos vartotojo vadovo aprašymą, skirtą darbui su elipsinių kreivių plokštumos taškų grupėmis. Norint pradėti darbą su šia grupių rūšimi, pasirodžius pradiniam vartotojo vadovo langui (**2.4 pav.**), pasirenkame „Elipsinių kreivių taškų grupės“. Tuomet pasirodo langas, skirtas Difio ir Helmano raktais apsikeitimo algoritmą nagrinėti minėtoje grupėje. Jis pateiktas **2.8 pav.**

Difio ir Helmano raktais apsikeitimo algoritmo nagrinėjimą elipsinėnių kreivių taškų grupėse galime atlikti dvejopai (analogiškai kaip ir sveikųjų skaičių multiplikacinėse grupėse), pasirinkdami vieną iš galimų šios programos dalies veikimo tipų. Atsivėrus šiam langui, jau parinktas pirmasis tipas „Duomenys generuojami atsitiktinai pagal pirminio skaičiaus eilę.“ Dirbant pasirinkus tokį tipą, konkrečių duomenų, reikalingų elipsinių kreivių grupei (pirminio skaičiaus p, parametrų poros (p_a, p_b) generatoriaus (g_1, g_2), šalies A slaptojo rakto x ir šalies B slaptojo rakto y) neįvedinėsime. Tokiu atveju, reikalavimus Difio ir Helmano algoritmui atitinkanys parametrai bus sugeneruojami atsitiktinai, pagal nurodytą pirminio skaičiaus eilę (bitais). Pirminio skaičiaus eilę negalima įvesti ranka, ją galima tik pasirinkti iš sąrašo. Taip yra todėl, kad aukštesnių eilių elipsinių kreivių parametrus (ypač generatorių) ne visada yra įmanoma sugeneruoti iš viso. Sąraše pateiktos tik didelės 192, 224, 256, 384, 521 bitų eilės, kurias pasirinkus bus parenkami rekomenduojamų elipsinių kreivių parametrai. (**2.9 pav.**).

Pasirinkus vieną iš rekomenduojamų elipsinių kreivių parametrų eilių turime nurodyti ir iteracijų skaičių. Efektyviam iteracijų vykdymui, kai įvestas didelis skaičius, šiose grupėse naudojome specialius metodus, tokius kaip „Double-and-add“ [10]. Programos vykdymui spaudžiame mygtuką „Gerai“. Čia yra galimas „Iteracijų stebėjimas pažingsniui“. Tada iškviečiamas iteracijų skaičiuoklis (**2.7 pav.**).



2.8 pav. Programos langas, skirtas darbui su elipsinių kreivių taškų grupėmis



2.9 pav. Pirminio skaičiaus eilės pasirinkimas iš sąrašo elipsinių kreivių grupėje

Apatinėje, rezultatų dalyje bus matomi šalies A ir šalies B suskaičiuoti viešieji raktai, atitinkamai $a = (a_1, a_2)$ ir $b = (b_1, b_2)$. Taip pat, galėsime matyti tiek šalies A, tiek šalies B apskaičiuotą bendrąjį slaptąjį raktą $K = (K_{A1}, K_{A2}) = (K_{B1}, K_{B2})$. Apatinėje šio vartotojo vadovo lango dalyje bus pateikiamos tik algoritmo veikimo trukmės. Parametrai buvo parinkti iš sąrašo pasirinkus pirmino skaičiaus eilę, taigi jų generavimo trukmių nenustatinėsime. Vidutinė trukmė pateikiama tik baigus visas iteracijas.

Galimas ir kitas programos vykdymo tipas – kai duomenys yra įvedami ranka. Tuomet renkamės programos veikimo tipą „Duomenys įvedami ranka“, t. y. priešingas pasirinkimas negu pavaizduota **2.8 pav.** Pasirinkus šį programos veikimo tipą suteikiama galimybė įvesti duomenis šalia parametro pavadinimo, o nereikalingi laukeliai apsaugomi nuo įvedimo. Tokiu atveju patys privalome būti

užtikrinti, kad įvedami parametrai teisingai atitinka Difio ir Helmano raktais apsikeitimo algoritmo reikalavimus. Ar parametrus įvedėme teisingai pamatysime įvykdžius programą. Bus atliktas patikrinimas kaip ir atsitiktinio parametrų parinkimo atveju ir išvestas pranešimas apie įvesties teisingumą. Kadangi duomenys įvesti ranka – iteracija bus tik viena, „Iteracijų stebėjimas pažingsniui“ yra apsaugotas nuo pažymėjimo. Parametrų p ir g generavimo trukmės nematuojamos, nes jie negeneruojami, o šiuo atveju – įvedami. Dėmesį kreipiame tik į algoritmo veikimo trukmes.

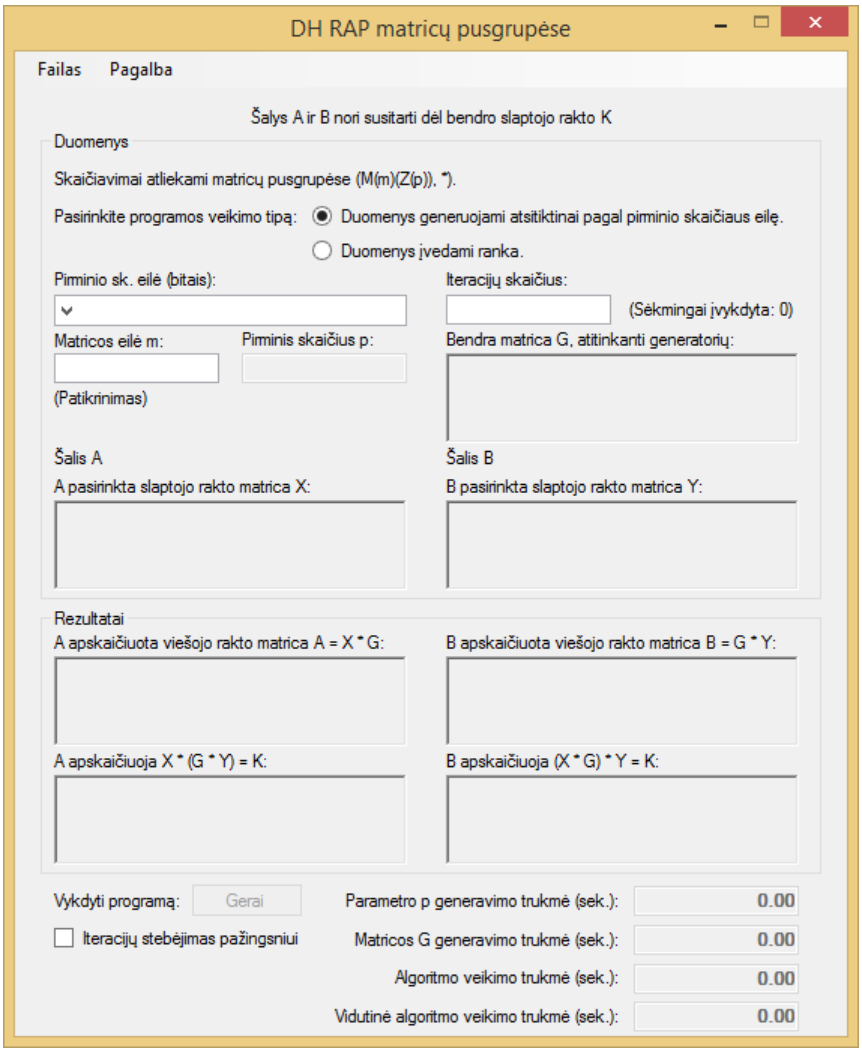
Kaip ir nagrinėjant Difio ir Helmano raktais apsikeitimo algoritmą sveikųjų skaičių grupėse, nepriklausomai nuo pasirinkto programos veikimo tipo šio vartotojo vadovo lango viršutinėje dalyje visada matysime meniu juostą, kuri skirta padėti valdyti programą (**2.8 pav.**). Paspaudus mygtuką „Failas“ → „Naujas generavimas“ galime atlikti naują generavimą (visi laukeliai bus išvalyti ir paruošti naujam skaičiavimui). Atlikdami „Failas“ → „Keisti grupę“ mygtuko paspaudimą galime pakeisti nagrinėjamos grupės rūšį į kitą. Šiuo atveju bus iš naujo iškviestas pradinis programos langas (**2.4 pav.**). Paspaudus „Failas“ → „Baigti“ baigiame darbą su programa. Atlikus mygtuko „Pagalba“ → „Apie“ paspaudimą bus pateikta informacija apie nagrinėjamą elipsinių kreivių taškų grupę bei nurodytas programos kūrėjas.

Galiausiai aptarsime programos veikimą, kai Difio ir Helmano apsikeitimo raktais algoritmas buvo vykdomas matricų multiplikacinėse pusgrupėse. Pradedant darbą, pasirodžius pradiniam vartotojo vadovo langui (**2.4 pav.**), pasirenkame „Matricų pusgrupės“. Tada pasirodo langas, skirtas Difio ir Helmano raktais apsikeitimo algoritmą nagrinėti šiose pusgrupėse (**2.10 pav.**).

Difio ir Helmano raktais apsikeitimo algoritmo nagrinėjimą matricų pusgrupėse galime atlikti, kaip jau įprasta, dvejopai, pasirinkdami vieną iš galimų šios programos dalies veikimo tipų. Atsivėrus programos langui, būna parinktas pirmasis tipas „Duomenys generuojami atsitiktinai pagal pirminio skaičiaus eilę.“ Darbą atliekant pasirinkus tokį tipą, konkretaus pirminio skaičiaus p neįvedinėsime. Tokiu atveju, jis bus sugeneruojamas atsitiktinai, pagal nurodytą pirminio skaičiaus eilę (bitais). Pirminio skaičiaus eilę galima įvesti ranka arba pasirinkti iš sąrašo. Sąraše pateiktos labai didelės eilės, kurių parametrai generuojami žymiai ilgiau, todėl patogumo dėlei 1024, 1536, ..., 16384 bitų parametrus galime pasirinkti čia, ir jų generavimo nelaukti. Visiškai tokį patį eilės pasirinkimą galėjome atlikti ir sveikųjų skaičių multiplikacinėse grupėse (**2.6 pav.**), panašų pasirinkimą – elipsinių kreivių plokštumos taškų grupėse. Matricų pusgrupėse svarbų vaidmenį atlieka matricos eilė m. Jos įvedimas lemia matricų, su kuriomis bus atliekamas Difio ir Helmano apsikeitimo raktais algoritmas, dydį. Tiksliau, parodo kiek eilučių ir stulepelių turės kiekviena, algoritme dalyvaujanti, kvadratinė matrica. Bendroji generatorių atitinkanti, išsigimusi matrica bus generuojama atsitiktinai, pagal tą pačią įvestą matricos eilę m.

Kiek vėliau privalome įvesti iteracijų skaičių. Programos vykdymui spaudžiame mygtuką „Gerai“. Čia yra galimas „Iteracijų stebėjimas pažingsniui“. Tokiu atveju iškviečiamas iteracijų

skaičiuoklis (**2.7 pav.**). Kiti šio programos lango patikrinimai ir apribojimai yra visiškai tokie patys kaip ir jau aprašytose grupėse.



2.10 pav. Programos langas, skirtas darbui su matricų pusgrupėmis

Šiek tiek žemiau pateiktoje, rezultatų sekcijoje bus išvestos šalies A ir šalies B suskaičiuotos matricos, atitinkančios viešuosius raktus. Taip pat, galėsime matyti tiek šalies A, tiek šalies B apskaičiuotą bendrąjį slaptąjį raktą K atitinkančias matricas, kurios yra lygios. Dar žemiau, apatinėje šio vartotojo vadovo lango dalyje pateikiamos tiek parametrų p ir G generavimo, tiek algoritmo veikimo trukmės. Vidutinė trukmė pateikiama tik baigus visas iteracijas. Programos rezultatų dalis ir trukmių matavimo laukeliai prieš jos vykdymą pateikti **2.10 pav.**

Tiriant Difio ir Helmano apsikeitimo raktais algoritmą matricų pusgrupėse, taip pat galimas rankinis vykdymo tipas – kai konkretūs duomenys yra įvedami ranka. Tuomet renkamės programos veikimo tipą „Duomenys įvedami ranka“, t. y. kitas tipo pasirinkimas, nei pavaizduota **2.10 pav.** Atlikus šį pasirinkimą, suteikiama galimybė įvesti pirminį skaičių ranką, o nereikalingi laukeliai apsaugomi nuo

įvedimo. Tuomet privalome žinoti, kad patys teisingai pasirinkome ir įvedėme pirminį skaičių. Ar tai padarėme teisingai, pamatysime įvykdžius programą, kuomet bus atliktas pirminio skaičiaus patikrinimas. Kai duomenys įvesti ranka – iteracija bus vienintelė, tuomet „Iteracijų stebėjimas pažingsniui“ yra negalimas, todėl jis apsaugotas nuo pažymėjimo. Šiuo atveju parametro p generavimo trukmės nematuosime, nes jo negeneravome, tačiau bendroji išsigimusi matrica, kaip ir anksčiau aprašyto programos veikimo tipo atveju, parenkama atsitiktiniu būdu. Taigi, matysime G generavimo ir algoritmo bendrąją bei vidutinę veikimo trukmes.

Nesvarbu kurį programos veikimo tipą matricų pusgrupėje būsime pasirinkę, vartotojo vadovo lango viršutinėje dalyje visada matysime meniu juostą programos valdymui (**2.10 pav.**). Paspaudus mygtuką „Failas“ → „Naujas generavimas“ galime atlikti naują generavimą (visi laukeliai bus išvalyti ir paruošti naujam skaičiavimui). Atlikdami „Failas“ → „Keisti grupę“ mygtuko paspaudimą, galime pakeisti nagrinėjamos grupės rūšį į kitą. Šiuo atveju bus iš naujo iškviestas pradinis programos langas (**2.4 pav.**). Pasirinkdami „Failas“ → „Baigti“ baigsime darbą su programa. Atlikus mygtuko „Pagalba“ → „Apie“ paspaudimą, pateikiama informacija apie pusgrupę, nurodomas programos kūrėjas.

2.3.2 PARAMETRŲ NUSTATINĖJIMO TECHNIKA

Paminėsime, kad atliekant daugiau kaip vieną iteraciją toje pačioje grupėje, kiekviena atskira DH algoritmo veikimo trukmė labai nežymiai, tačiau priklauso ir nuo vykdymo metu naudojamų parametrų individualių savybių, net ir esant tai pačiai pirminio skaičiaus eilei. Pavyzdžiui, prie pasirinktos pirminio skaičiaus eilės atlikome dvi iteracijas. Pirmosios iteracijos parametrai buvo atsitiktinai sugeneruoti taip, kad jų pirmieji skaitmenys yra didžiausi vienženkliai skaičiai tokie kaip 8 ar 9, o antrosios iteracijos atveju toje pozicijoje pasitaikė, tarkime 1 ar 2. Taip nutikus, suprantama, kad pirmoji iteracija truks ilgiau, o kiek ilgiau priklauso jau nuo nagrinėjamos grupės pirminio skaičiaus eilės. Atliekant didesnį iteracijų skaičių ir skaičiuojant vidutinę trukmę, tokie veiksniai esminės įtakos nesudaro.

Parametrų generavimo trukmės neįeina į Difio ir Helmano raktais apsikeitimo algoritmo veikimo trukmės skaičiavimą, jos tiesiog parodo kiek laiko užtruko sugeneruoti atskirą parametrą p ir g (arba G, jei algorimas vykdomas matricų pusgrupėse). Priminsime, kad parametrų generavimas buvo galimas tik sveikųjų skaičių multiplikacinėse ir matricų pusgrupėse, o elipsinėse kreivėse naudojome tik rekomenduojamus, kadangi tam tikrais atskirais atvejais elipsinių kreivių grupėse jų sugeneruoti beveik neįmanoma. Sveikųjų skaičių grupėse parametrų generavimo trukmė absoliučiai priklauso nuo pirminio skaičiaus eilės – ją didinant, trukmė ilgėja. Matricų pusgrupėse tokį trukmės pailgėjimą lemia pirminio skaičiaus ir matricos eilė.

Pirminio skaičiaus generavimui ir patikrinimui naudojome Millerio-Rabino skaičių pirmiškumo testą (**2.4 algoritmas**). Tai praktikoje dažniausiai naudojamas testas, kuris remiasi tokia savybe: jei n yra nelyginis pirminis skaičius, tai galima rasti tokius sveikuosius skaičius r ir s, kad $n-1 = 2^s \cdot r$. Čia r – nelyginis skaičius. Pasirinkus betkokį sveikąjį skaičių a, tenkinantį sąlygą $1 \leq a \leq n-1$, bus tenkinama arba lygybė $a^r \equiv 1\ (mod\ n)$, arba $a^{2/r} \equiv -1\ (mod\ n)$, su kai kuriais j, $0 \leq j \leq s-1$. Kadangi šią lygybę tenkina ir kai kurie sudėtiniai skaičiai, įvedamas saugumo parametras t. Jis tiksliai nusako, keliomis skirtingomis a reikšmėmis reikia patikrinti pirmiškumo sąlygą, siekiant sumažinti tikimybę, jog sudėtinis skaičius bus pripažintas pirminiu. Kadangi operacijos atliekamos moduliu n, tai $-1 \equiv n-1\ (mod\ n)$ [2].

Tikrinant ir generuojant parametrą g rėmėmės jau pateiktomis 2.1 ir 2.2 teoremomis (žr. 18 psl.). Parametrai x ir y – slaptieji raktai, kuriems yra vieninelis reikalavimas. Jie privalo neviršyti pirminio skaičiaus. Elipsinių kreivių atveju, nors parametrų ir negeneruojame, pirminio skaičiaus p ir slaptųjų raktų patikrinimą atliekame. Papildomai atliekamas elipsinių kreivių parametrų a ir b patikrinimas kurių reikalavimai pateikti 2.8 apibrėžime (žr. 18 psl.). Matricų pusgrupėse parametrai X ir Y – slaptuosius raktus atitinkančios matricos, kurių visi elementai yra sveikieji skaičiai, mažesni už p.

2.4 algoritmas. Millerio-Rabino pirmiškumo testas [2]

(įvestis *n*, *t*)

---

n ← *n* – 1
s ← 0
Kol *r* – nelyginis, vykdyti
    r ← *r* / 2 (tik sveikoji dalis)
    s ← *s* + 1
i ← 0
Kol *i* < *t*, vykdyti
    a ← atsitiktinis skaičius nuo 2 iki *n* – 2
    y ← a^r (mod *n*)
    Jei *y* ≠ 1 ir *y* ≠ *n* – 1, vykdyti
    j ← 1
    Kol *j* ≤ *s* – 1 ir *y* ≠ *n* – 1, vykdyti
        y ← y^2 (mod *n*)
        Grąžinti „**sudėtinis**“, jei *j* = 1
        j ← *j* + 1
    Grąžinti „**sudėtinis**“, jei *j* ≠ *n* – 1
Grąžinti „**pirminis**“

(išvestis „**pirminis**“ arba „**sudėtinis**“)

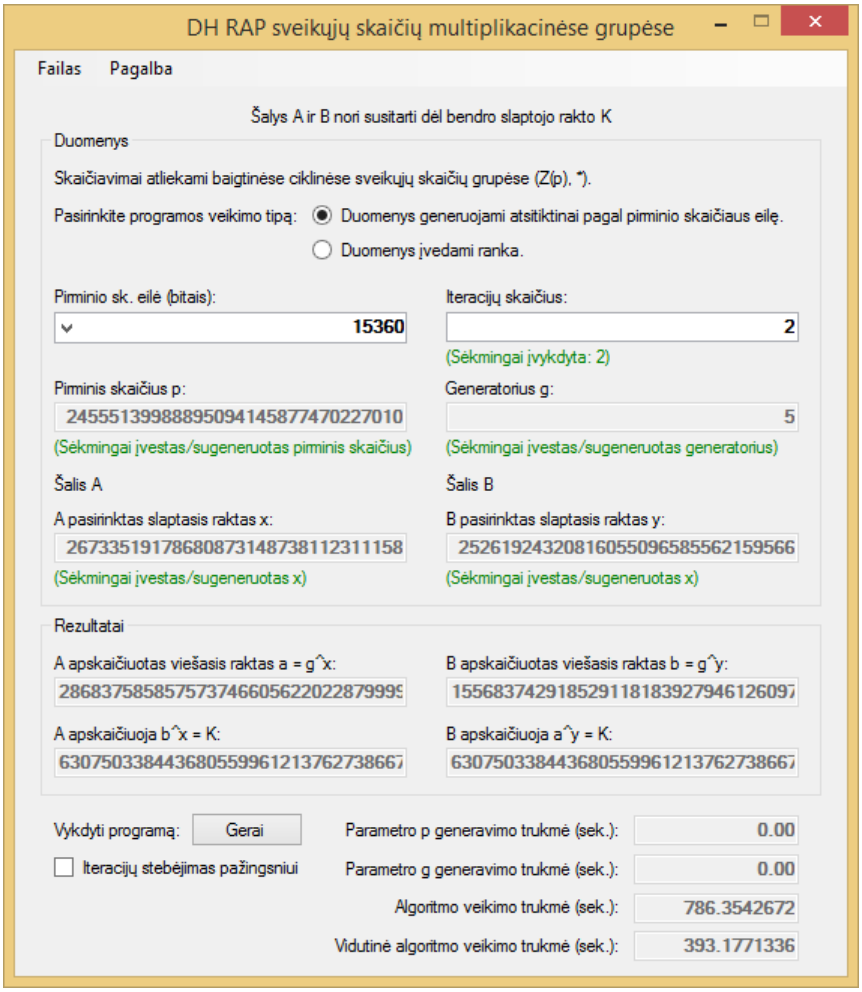
3. TIRIAMOJI DALIS

Trumpai apibūdinsime pagrindinius tyrimus, kuriuos pateiksime šioje dalyje. Pirmiausiai tiriamojoje dalyje aptarsime parametrų generavimą, būdingą visoms grupėms, kur vykdėme nagrinėjamą algoritmą. Taip pat šioje dalyje nustatysime bei ištirsime vidutines Difio ir Helmano apsikeitimo raktais algoritmo veikimo trukmes. Tokį tyrimą atliksime jau gerai pažįstamose sveikųjų skaičių multiplikacinėse bei elipsinių kreivių plokštumos taškų grupėse. Ištyrus, kuriose grupėse Difio ir Helmano apsikeitimo raktais algoritmas vykdomas greičiau, toliau darbą tęsime su tomis, kuriose algoritmas veikė efektyviau. Toliau jose fiksuosime vidutines veikimo trukmes ir tirsime matricos eilės priklausomybę nuo pirminio skaičiaus eilės matricų pusgrupėse. Kiek vėliau šią priklausomybę suformuluosime, kaip dėsnį. Nustatinėjant algoritmo veikimo trukmę, svarbu užtikrinti, kad rezultatai būtų kaip įmanoma tikslesni bei turėtų kuo didesnę prasmę. Tam tikslui sveikųjų skaičių multiplikacinėse, elipsinių kreivių plokštumos taškų grupėse ir matricų pusgrupėse buvo suteikta galimybė algoritmą kartoti kiek norime kartų, pagal pasirinktinai įvestą iteracijų skaičių. Natūralu, kad atliekant daugiau iteracijų bendra Difio ir Helmano algoritmo veikimo trukmė ilgėja tiek kartų, kiek iteracijų numatėme atlikti, tačiau jį kartodami daugiau kartų ir skaičiuodami vidutines algoritmo trukmes gausime tikslesnius rezultatus, nei atlikus tik keletą testavimų. Galiausiai pateiksime testavimo rezultatus lentelių ir grafikų pavidalu, kurių dėka galėsime suformuluoti išvadas.

3.1 EKSPERIMENTINIAI SKAIČIAVIMAI IR TESTAVIMAS

Šiame poskyryje pateiksime Difio ir Helmano apsikeitimo raktais algoritmo skirtingose grupėse eksperimentinius skaičiavimus ir aprašyto vartotojo vadovo testavimo rezultatus. Tai galima sėkmingai padaryti teisingai užpildžius vartotojo sąsajos laukelius ir įvykdžius programą. Pirmiausiai **3.1 pav.** pateikiame programos lango paveikslą po vykdymo, kai Difio ir Helmano apsikeitimo raktais algoritmą nagrinėjame sveikųjų skaičių multiplikacinėse grupėse, pasirinkus „Duomenys generuojami atsitiktinai pagal pirminio skaičiaus eilę.“ tipą.

Pagal šį programos veikimo tipą iš sąrašo buvo pasirinkta pirminio skaičiaus eilė – 15360 bitai ir iteracijų skaičius – 2. Po vykdymo sugeneruoti parametrai bei A ir B šalių slaptieji raktai. Rezultatų dalyje apskaičiuotas viešasis raktas a nusiųstas šaliai B, o viešasis raktas b nusiųstas šaliai A. Abi šalys gautus viešuosius raktus apskaičiavo bendrąjį slaptąjį raktą K, kuris, kaip ir buvo galima tikėtis, sutampa. Matome ir išmatuotas trukmes. Šiuo atveju visą Difio ir Helmano apsikeitimo raktais algoritmą įvykdyti užtruko 786.3542627 sekundžių, o vidutinė algoritmo veikimo trukmė 393.1771336.

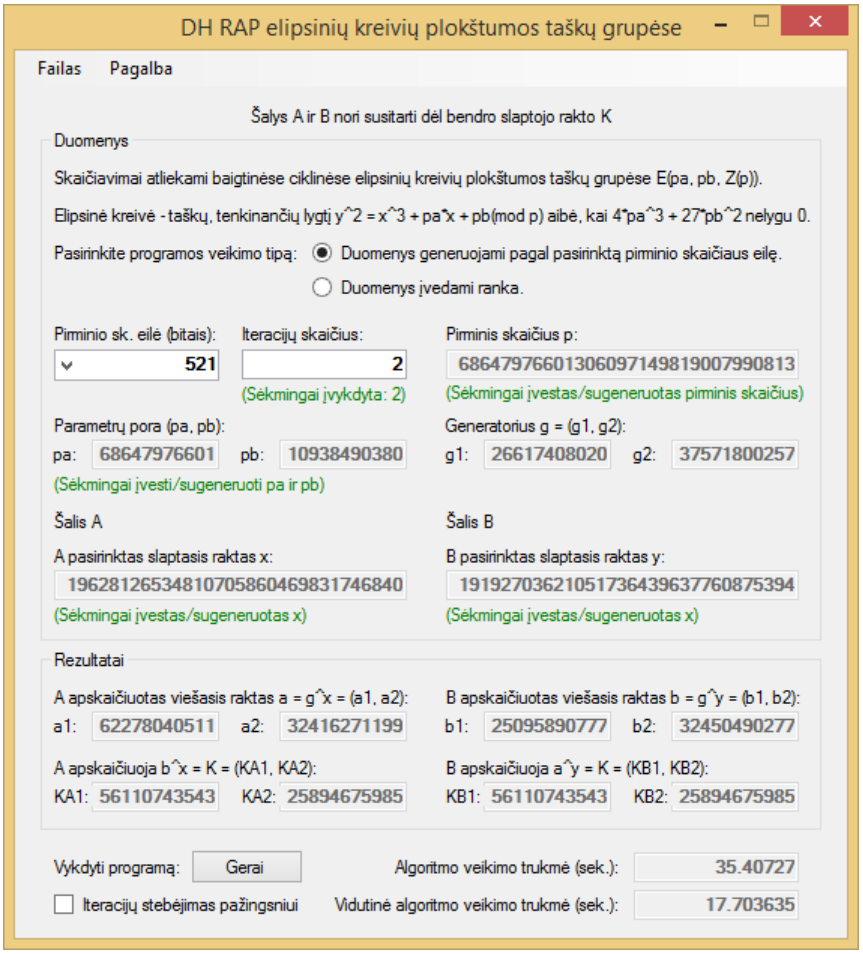


3.1 pav. Programa po vykdymo sveikųjų skaičių grupėse (duomenys generuojami atsitiktinai)

Programos pavyzdžio po vykdymo, kai Difio ir Helmano apsikeitimo raktais algoritmą nagrinėjame sveikųjų skaičių skaičių multiplikacinėse grupėse, pasirinkus „Duomenys įvedami ranka.“ tipą, nepateikiame. Jo veikimo principas visiškai toks pat, tik tuo atveju parametrus vartotojas įveda pagal savo pageidavimą.

Toliau (**3.2 pav.**) pateikiame vartotojo sąsajos lango paveikslą po vykdymo, kai Difio ir Helmano apsikeitimo raktais algoritmą nagrinėjame elipsinių kreivių plokštumos taškų grupėse, pasirinkus „Duomenys generuojami atsitiktinai pagal pirminio skaičiaus eilę.“ tipą.

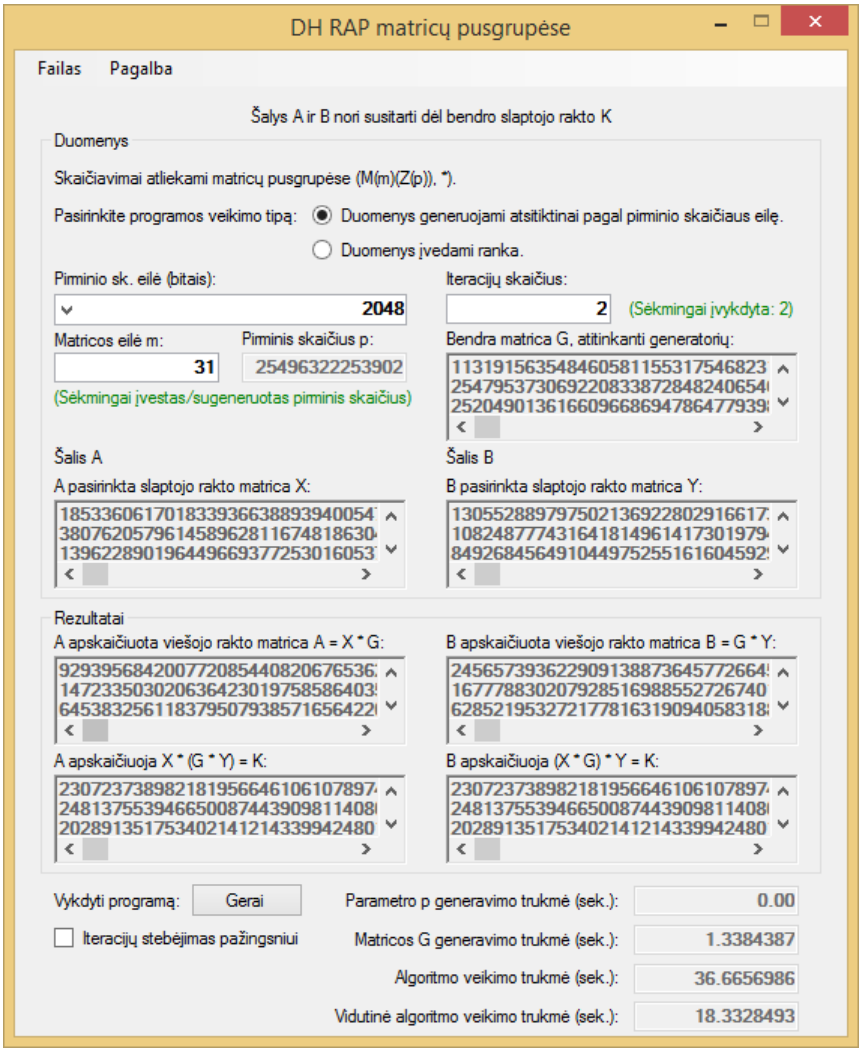
Šiame paveiksle, pagal pasirinktą programos veikimo tipą buvo iš sąrašo pasirinkta pirminio skaičiaus eilė – 521 bitas ir iteracijų skaičius – 2. Po vykdymo sugeneruoti parametrai bei A ir B šalių slaptieji raktai. Rezultatų dalyje apskaičiuotas viešasis raktas $a = (a_1, a_2)$ nusiųstas šaliai B, o viešasis raktas $b = (b_1, b_2)$ nusiųstas šaliai A. Abi šalys gautus viešuosius raktus apskaičiavo bendrąjį slaptąjį raktą $K = (K_{A1}, K_{A2}) = (K_{B1}, K_{B2})$, kuris sutampa. Matome ir išmatuotas Difio ir Helmano apsikeitimo raktais algoritmo trukmes.



3.2 pav. Programa po vykdymo elipsinių kreivių grupėse (duomenys parenkami atsitiktinai)

Šiuo atveju visą Difio ir Helmano apsikeitimo raktais algoritmą įvykdyti užtruko 35.40727 sekundžių, o vidutinė algoritmo trukmė per 2 iteracijas buvo 17.703635 sekundės. Kuomet Difio ir Helmano apsikeitimo raktais algoritmą nagrinėjame elipsinių kreivių plokštumos taškų grupėse, pasirinkus „Duomenys įvedami ranka.“ tipą, programos vaizdo po vykdymo neaprašome – tokiu atveju duomenys įvedami ranka, o skaičiavimai atliekami analogiškai.

Galiausiai pateikiame vartotojo sąsajos lango paveikslą po vykdymo, kai Difio ir Helmano apsikeitimo raktais algoritmą nagrinėjame matricų pusgrupėse, pasirinkus „Duomenys generuojami atsitiktinai pagal pirminio skaičiaus eilę.“ tipą (**3.3 pav.**). Pagal šį programos veikimo tipą iš sąrašo buvo pasirinkta pirminio skaičiaus eilė – 2048 bitai, matricos eilė m – 31 ir iteracijų skaičius – 2. Po vykdymo sugeneruoti parametrai bei A ir B šalių slaptuosius raktus atitinkančios matricos X ir Y. Rezultatų dalyje apskaičiuota viešąjį raktą atitinkanti matrica A ir ji nusiųsta šaliai B, o viešąjį raktą atitinkanti matrica B nusiųsta šaliai A. Abi šalys, gauvusios viešuosius raktus, apskaičiavo bendrąjį slaptąjį raktą K, kuris, kaip žinoma, sutampa. Taip pat galime matyti išmatuotas trukmes. Visą Difio ir Helmano apsikeitimo raktais algoritmą įvykdyti užtruko 36.6656986 sekundžių, o vidutinė algoritmo veikimo trukmė 18.3328493.



3.3 pav. Programa po vykdymo matricų pusgrupėse (duomenys parenkami atsitiktinai)

Kaip ir anksčiau apibūdintais atvejais, programos pavyzdžio po vykdymo, kai Difio ir Helmano apsikeitimo raktais algoritmą nagrinėjame matricų pusgrupėse, pasirinkus „Duomenys įvedami ranka.“ tipą, nepateikiame. Šis tipas skirtųsi tik tuo, kad pirminis skaičius būtų įvestas pačio programos vartotojo, o ne sugeneruotas programos, arba pasirinktas iš sąrašo.

Svarbu paminėti, kad testuojant bei tikrinant DH veikimą programos sveikųjų skaičių multiplikacinėse, elipsinių kreivių plokštumos taškų grupėse bei matricų pusgrupėse vien tik darbo kompiuteriu nepakako. Atliekant skaičiavimus tik šitaip, negalėtumėme garantuoti programos patikimumo dėl galimų logikos ar sintaksės klaidų. Taigi, testuodami tam tikrus eksperimentinius skaičiavimus atlikinėjome ir ranka. Tada juos tikrinome su gautais programoje apskaičiuotais rezultatais. Galiausiai tokiu būdu įsitikinome, kad programa veikia teisingai. Skaičiavimų ranka atlikome daug, todėl pateiksime keletą visiškai primityvių pavyzdžių, kaip tai padarėme. Nuo anksčiau pateiktų pavyzdžių lentelių pavidalu (žr. 2.2 ir 2.2.1 skyrius) šie skiriasi tuom, kad juose pateikiami ne schemizuoti, tačiau visiškai pilni skaičiavimai įskaitant ir detalius tarpinius veiksmus.

3.1 Pavyzdys. Tarkime, dabar nagrinėjame sveikųjų skaičių multiplikacinę grupę $\langle \mathbb{Z}_p^*; \cdot \rangle$. Nagrinėjamoje grupėje duotas generatorius $g = 5$. Subjektas A pasirenka savo slaptąjį raktą $x = 4$, o subjekto B, slaptasis raktas $y = 3$. Abu subjektai nori susitarti dėl bendrojo slaptojo rakto K. Subjektas A apskaičiuoja $a = g^x = 5^4 \ (mod\ 7) = 2$ ir šį viešąjį raktą siunčia B. Subjektas B apskaičiuoja $b = g^y = 5^3 \ (mod\ 7) = 6$ ir jį siunčia A. Tada subjektas A randa $b^x = 6^4 (mod\ 7) = 1 = K$. Analogišką skaičiavimą atlieka ir B rasdamas $a^y = 2^3 (mod\ 7) = 1 = K$. Abu subjektai gavo vienodą rezultatą. Šiuo ir kitais panašiais pavyzdžiais tikrinome Difio it Helmano raktais apsikeitimo algoritmo veikimo teisingumą jau sukurtoje programoje.

3.2 Pavyzdys. Tarkime, dabar naginėjame elipsinių kreivių plokštumos taškų grupę $E(7, 9, \mathbb{Z}_{11}^*)$. Čia parametrai $a = 7$ ir $b = 9$. Joje generatorius $g = (7, 5)$. Subjektas A pasirenka savo slaptąjį raktą $x = 2$, o subjekto B slaptasis raktas yra $y = 3$. A apskaičiuoja $a = g^x = (7, 5)^2 = (7, 5) + (7, 5)$. Primename, kad šioje sumoje pirmas dėmuo yra $P(x_1, y_1)$, o antras $Q(x_2, y_2)$, o jų rezultatas bus $R(x_3, y_3)$,. Randamas tarpinis parametras λ, kai $P = Q$. $\lambda = \frac{3x_1^2 + a}{2y_1} = \frac{(3 \cdot 49 + 7)\ (mod\ 11)}{(2 \cdot 5)\ (mod\ 11)} = \frac{0}{10} = 0 \cdot 10^{-1} = 0$. Su gauta $\lambda = 0$ randame $x_3 = \lambda^2 - x_1 - x_2 = (0^2 - 7 - 7)\ (mod\ 11) = 8$ bei randame $y_3 = \lambda(x_1 - x_3) - y_1 = (0 \cdot (7 - 8) - 5)\ (mod\ 11) = 6$. Tokiu būdu gavome, kad $R(x_3, y_3) = (8, 6)$. Tai yra subjekto A viešasis raktas. Panašiai viešąjį raktą elipsinių kreivų taškų plokštumos grupėje skaičiavome ir subjekto B atveju. Kadangi, B slaptasis raktas yra $y = 3$ čia atliekamas skaičiavimas bus truputį sudėtingesnis: $b = g^y = (7, 5)^3 = (7, 5) + (7, 5) + (7, 5)$. Pirmas sumavimo veiksmas atliekamas analogiškai, kaip ir subjekto A atveju, taigi uždavinį galime perfrazuoti šitaip: $g^y = (7, 5)^3 = (8, 6) + (7, 5)$. Toliau vėl skaičiuojame λ, tik šiuo atveju bus $P \neq Q$. Tokiu atveju $\lambda = \frac{y_2 - y_1}{x_2 - x_1} = \frac{(5-6)(mod\ 11)}{(7-8)(mod\ 11)} = \frac{10}{10} = 1$. Su gauta $\lambda = 1$ randame $x_3 = \lambda^2 - x_1 - x_2 = (1^2 - 8 - 7)\ (mod\ 11) = 8$ bei randame $y_3 = \lambda(x_1 - x_3) - y_1 = (1 \cdot (8 - 7) - 5)\ (mod\ 11) = 5$. Taigi rezultatas $R(x_3, y_3) = (8, 5)$. Dabar jau apskaičiuoti abiejų subjektų viešieji raktai. Analogišką procedūrą atlikinėjame ir skaičiuojant bendrąjį slaptąjį raktą. Subjektas A gavęs B viešąjį raktą $(8, 5)$ skaičiuoja $K_A = b^x = (8, 5)^2 = (8, 5) + (8, 5)$, o subjektas B, gavęs A viešąjį raktą $(8, 6)$ skaičiuoja $K_B = a^y = (8, 6)^3 = (8, 6) + (8, 6) + (8, 6)$. Šį kartą nesiplėsime ir tarpinius veiksmus nepateiksime. Jie atliekami analogišku būdu, kaip iš skaičiuojant viešuosius raktus. Galutinis abiejų suskaičiuotas rezultatas $K_A = K_B = K = (7, 5)$. Šiuo ir kitais analogiškais pavyzdžiais tikrinome programos elipsinių kreivių plokštumos taškų dalies teisingumą.

3.3 Pavyzdys. Tarkime, naginėjame matricų pusgrupę $\mathbb{M}_2(\mathbb{Z}_7^*)$. Pirminis skaičius $p = 7$, o matricos eilė $m = 2$. Parenkame bendrąją, generatorių atitinkančią, išsigimusią matricą $G = \begin{pmatrix} 5 & 3 \\ 5 & 3 \end{pmatrix}$, kurios determinantas lygus nuliui: $\begin{vmatrix} 5 & 3 \\ 5 & 3 \end{vmatrix} = 0$. Subjektas A pasirenka savo slaptąją matricą $X = \begin{pmatrix} 2 & 1 \\ 3 & 0 \end{pmatrix}$, o

subjektas B pasirenka slaptąją matricą $Y = \begin{pmatrix} 1 & 6 \\ 1 & 1 \end{pmatrix}$. Subjektas A apskaičiuoja viešąją matricą $A = X \cdot G = \begin{pmatrix} 2 & 1 \\ 3 & 0 \end{pmatrix} \cdot \begin{pmatrix} 5 & 3 \\ 5 & 3 \end{pmatrix} = \begin{pmatrix} 2 \cdot 5 + 1 \cdot 5 \ (mod\ 7) & 2 \cdot 3 + 1 \cdot 3 \ (mod\ 7) \\ 3 \cdot 5 + 0 \cdot 5 \ (mod\ 7) & 3 \cdot 3 + 0 \cdot 3 \ (mod\ 7) \end{pmatrix} = \begin{pmatrix} 1 & 2 \\ 1 & 2 \end{pmatrix}$. Analogiškai kaip ir subjektas A, subjektas B apskaičiuoja viešąją matricą $B = G \cdot Y = \begin{pmatrix} 5 & 3 \\ 5 & 3 \end{pmatrix} \cdot \begin{pmatrix} 1 & 6 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 5 \cdot 1 + 3 \cdot 1 \ (mod\ 7) & 5 \cdot 6 + 3 \cdot 1 \ (mod\ 7) \\ 5 \cdot 1 + 3 \cdot 1 \ (mod\ 7) & 5 \cdot 6 + 3 \cdot 1 \ (mod\ 7) \end{pmatrix} = \begin{pmatrix} 1 & 5 \\ 1 & 5 \end{pmatrix}$. Pagal tokias pat taisykles apskaičiuojamos ir bendrąjį slaptąjį raktą atitinkančios matricos $K_A = X \cdot B = X \cdot (G \cdot Y)$ ir $K_B = A \cdot Y = (X \cdot G) \cdot Y$. Skiriasi tik patys matricų elementai. Taigi bendras slaptasis raktas $K = K_A = K_B = \begin{pmatrix} 3 & 1 \\ 3 & 1 \end{pmatrix}$.

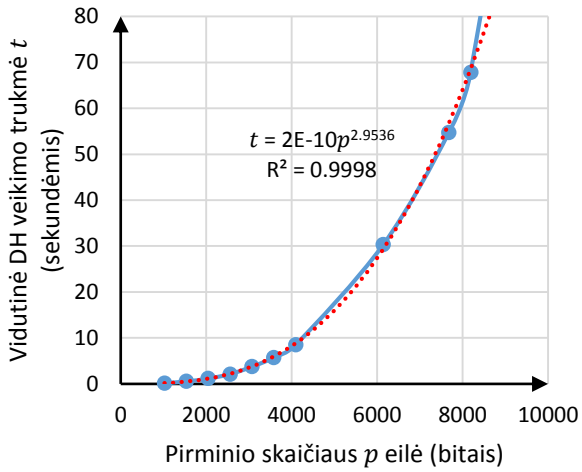
Bendrai kalbant apie visų skirtingų grupių testavimą, laikėmės vieningos strategijos: pradėjome nuo pačių paprasčiausių skaičiavimų, o vėliau ėmėme naudoti vis sudėtingesnius elementus. Galiausiai pasiekėme labai didelį DH algoritmo saugumą, stipriai viršijantį minimalius saugumo reikalavimus.

3.2 DIFIO IR HELMANO APSIKEITIMO RAKTAIS ALGORITMO EFEKTYVUMO TYRIMAS IR ANALIZĖ

Šiame poskyryje aprašėme atliktus tyrimus, kuriose baigtinėse ciklinėse grupėse Difio ir Helmano apsikeitimo raktais algoritmas yra vykdomas efektyviausiai. Kitaip tariant, nustatinėjome, kur tiriamą algoritmą galime įvykdyti per trumpiausią laiko tarpą. Tyrėme sveikųjų skaičių multiplikacines ir elipsinių kreivių plokštumos taškų grupes. Naudojome parametrus pagal pasirinktas įvairaus ilgio (bitais) pirminio skaičiaus eiles. Su kiekviena iš pasirinktų eilių atlikome po 100 Difio ir Helmano raktais apsikeitimo algoritmo iteracijų. **3.1 lentelėje** pateikiame Difio ir Helmano algoritmo bendros ir vidutininės veikimo trukmių rezultatus. Pirminio skaičiaus eilės, kurios lentelėje pažymėtos žvaigždute (*), vėliau bus naudojamos sulyginimui, pagal atitinkamą saugumo lygmenį. Kitos pirminio skaičiaus eilės ir prie jų išmatuotos bendrosios bei vidutinės algoritmo veikimo trukmės pateikiamos, tam kad galėtume matyti nuoseklesnį trukmių kitimą. Tai yra tik tarpiniai rezultatai. Nustatėme, kad vykdydami DH algoritmą sveikųjų skaičių multiplikacinėse grupėse naudojant vis didesnius parametrus, tiek bendra, tiek vidutinė algoritmo veikimo trukmė auga pagal laipsninę priklausomybę. Šį faktą patikrinome atsižvelgdami į aproksimuotą priklausomybės formulę bei naudodami tendencijos linijas, kurios padeda nustatyti, kokia yra panašiausia priklausomybė lyginant su gautąja. Vidutinės algoritmo veikimo trukmės kitimą didinant pirminio saičiaus eilę atvaizdavome ir grafiškai (**3.4 pav.**). Šalia grafiko pateikta ir pati priklausomybės formulė.

3.1 lentelė. DH RAP veikimo trukmių rezultatai sveikųjų skaičių multiplikacinėse grupėse

Pirminio skaičiaus eilė (bitais)	Bendra algoritmo veikimo trukmė (sekundėmis)	Vidutinė algoritmo veikimo trukmė (sekundėmis)	Iteracijų skaičius
1024	15.6328	0.1563276	100
1536 *	51.1174	0.5111738	100
2048 *	114.341	1.1434105	100
2560	210.378	2.1037755	100
3072 *	371.596	3.7159561	100
3584	570.251	5.7025129	100
4096	848.416	8.4841614	100
6144	3027.68	30.2768472	100
7680 *	5471.18	54.7118199	100
8192	6785.24	67.8524089	100
15360 *	45956.6	459.566436	100
16384	54982.4	549.823529	100

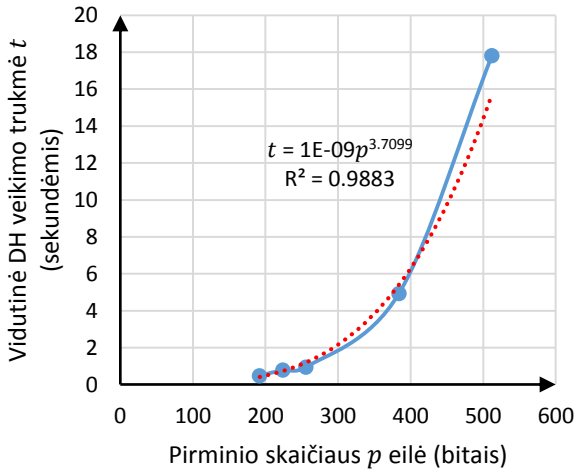


3.4 pav. Vidutinės trukmės kitimas didinant pirminio skaičiaus eilę (sveikųjų skaičių grupėse).

Toliau pateiksime veikimo trukmių rezultatus elipsinių kreivių taškų grupėse. Algoritmo veikimo vidutinę trukmę šiose grupėse tyrėme naudojant parametrus pagal pasirinktas tam tikrų bitų ilgio pirminio skaičiaus eiles. Kaip ir sveikųjų skaičių multiplikacinių grupių atveju, su kiekviena iš pasirinktų eilių atlikome po 100 Difio ir Helmano apsikeitimo raktais algoritmo iteracijų. **3.2 lentelėje** pateikiame Difio ir Helmano algoritmo bendros ir vidutininės veikimo trukmių rezultatus. Nesunku pastebėti, kad vykdydami DH algoritmą elipsinių kreivių taškų grupėse naudojant vis didesnius parametrus, bendra ir vidutinė algoritmo veikimo trukmė buvo vis ilgesnė. Ji elipsinių kreivių taškų grupėse auga pagal laipsninę priklausomybę. Tai nustatėme iš aproksimuotos priklausomybės lygties ir tendencijų linijų. Tokią priklausomybę nustatėme jau anksčiau tirtose sveikųjų skaičių grupėse. Vidutinės algoritmo veikimo trukmės kitimą didinant pirminio saičiaus eilę atvaizdavome **3.5 pav.**

3.2 lentelė. DH RAP veikimo trukmių rezultatai elipsinių kreivių taškų grupėse

Pirminio skaičiaus eilė (bitais)	Bendra algoritmo veikimo trukmė (sekundėmis)	Vidutinė algoritmo veikimo trukmė (sekundėmis)	Iteracijų skaičius
192	47.46556	0.4746556	100
224	78.6971	0.7869712	100
256	94.32528	0.9432528	100
384	493.88185	4.9388185	100
512	1781.6229	17.816229	100



3.5 pav. Vidutinės trukmės kitimas didinant pirminio skaičiaus eilę (elipsinių kreivių grupėse).

Dabar palyginsime Difio ir Helmano raktais apsikeitimo algoritmo veikimo trukmes sveikųjų skaičių multiplikacinėse ir elipsinių kreivių plokštumos taškų grupėse. Lyginimą atliksime pagal penkis pasirinktus saugumo lygmenis. Kadangi, elipsinėse kreivėse atliekami skaičiavimai yra sudėtingesni, jose DH algoritmo rekomenduojamas saugumas užtikrinamas prie mažesnių pirminio skaičiaus eilių. Tam kad galėtume sulyginti nustatytas trukmes pateikiame **3.3 lentelę**, kurioje pateikti lyginamų pirminio skaičiaus eilių atitikmenys. Toliau **3.4 lentelėje** pateikiame apskaičiuotas vidutines DH algoritmo veikimo trukmes skirtingose grupėse prie atitinkamų pirminio skaičiaus eilių.

3.3 lentelė. Reikalavimai grupių sulyginimui pagal pirminio sk. eilę [6]

Saugumo lygis	Eilė sveikųjų skaičių multiplikacinėse grupėse (bitais)	Eilė elipsinių kreivių plokštumos taškų grupėse (bitais)
96	1536	192
112	2048	224
128	3072	256
192	7680	384
256	15360	512

3.4 lentelė. Apskaičiuotų vidutinių DH veikimo trukmių sulyginimas pagal atitinkamą saugumą.

Lyginamų grupių pirminio skaičiaus eilės	DH algoritmo vidutinė veikimo trukmė sveikųjų skaičių multiplikacinėje grupėje (sek.)	DH algoritmo vidutinė veikimo trukmė elipsinių kreivių plokštumos taškų grupėje (sek.)
1536 ↔ 192	0.5111738	0.4746556
2048 ↔ 224	1.1434105	0.7869712
3072 ↔ 256	3.7159561	0.9432528
7680 ↔ 384	54.7118199	4.9388185
15360 ↔ 521	459.566436	17.816229

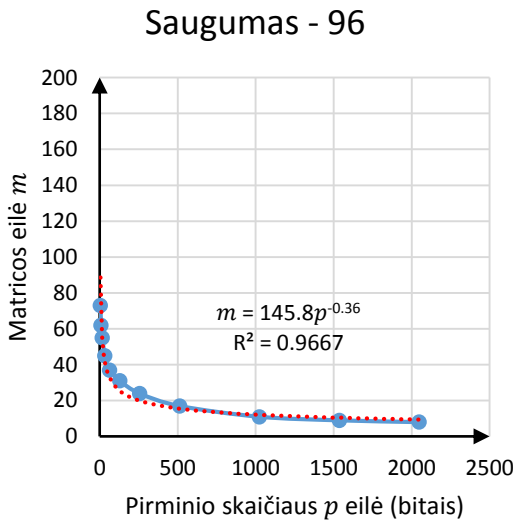
Taigi, atlikus sulyginimą, akivaizdu kad Difio ir Helmano raktais apsikeitimo algoritmo vidutinės veikimo trukmės skiriasi. Visais penkiais atvejais, prie skirtingų saugumo lygmenų, DH algoritmas buvo vykdomas efektyviau elipsinių kreivių plokštumos taškų grupėse, nei sveikųjų skaičių multiplikacinėse grupėse.

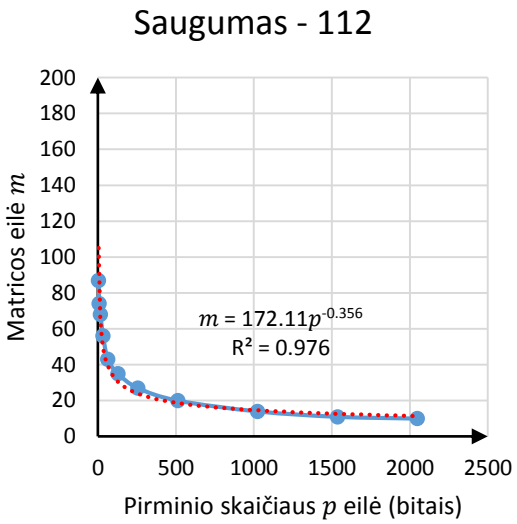
3.3 PIRMINIO SKAIČIAUS IR MATRICOS EILIŲ SĄRYŠIS DH VYKDANT MATRICŲ PUSGRUPĖSE

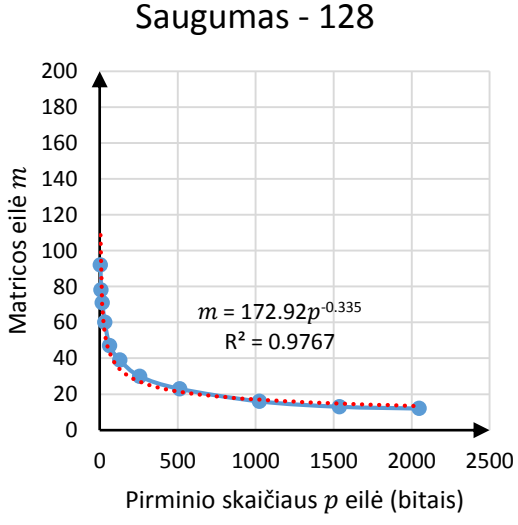
Šiame tiriamosios dalies etape pateikiame ištirtą pirminio skaičiaus ir matricos eilių sąryšį, kai Difio ir Helmano apsikeitimo raktais algoritmą vykdėme matricų pusgrupėse. Priminsime, kad šiam tyrimui atlikti naudojome tik kvadratines matricas – tokias, kurių eilučių ir stulpelių skaičius lygus. Tam, kad galėtume ištirti šią priklausomybę, pasirinkome penkias vidutines DH algoritmo veikimo trukmes prie tam tikrų saugumo lygmenų elipsinių kreivių plokštumos taškų grupėse. Elipsinių kreivių grupes pasirinkome ne atsitiktinai – jose DH algoritmas veikia žymiai sparčiau, ypač prie labai didelių pirminio skaičiaus eilių. Tokį faktą atspindinčią lentelę jau pateikėme anksčiau (**3.4 lentelė**). Taigi, tiriant šį sąryšį, pirmiausiai matricų pusgrupėse pasirinkome pirminio skaičiaus eiles. Tada fiksuodami vidutinę veikimo trukmę elipsinių kreivių plokštumos taškų grupėse prie tam tikro saugumo, nuolat keitėme matricos eilę m. Tokiu būdu matricų pusgrupėse stengėmės gauti kuo artimesnę vidutinę Difio ir Helmano apsikeitimo raktais algoritmo veikimo trukmę tai, kurią fiksavome elipsinėse kreivėse prie to pačio saugumo lygmens. Svarbu paminėti, kad visiškai identiškos trukmės atlikenat šią procedūrą gauti negalėjome. Taip yra todėl, kad matricos eilė privalo būti tik sveikasis skaičius. Pats tokios procedūros aprašymas atrodo pakankamai sudėtingas, tačiau jį paprasčiau suvokti iš toliau pateiktos lentelės.

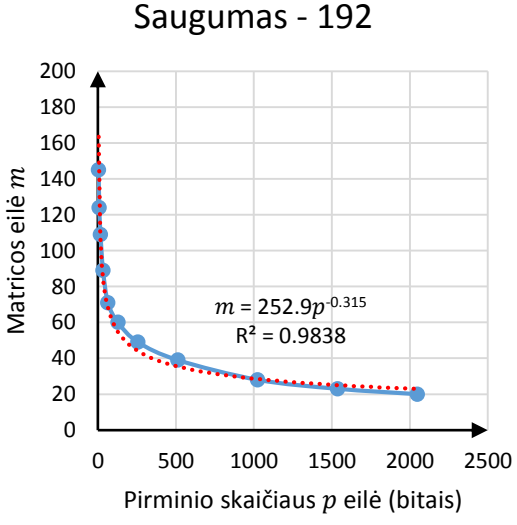
3.5 lentelė. Matricos eilės ir pirminio skaičiaus sąryšis vykdant 100 iteracijų.

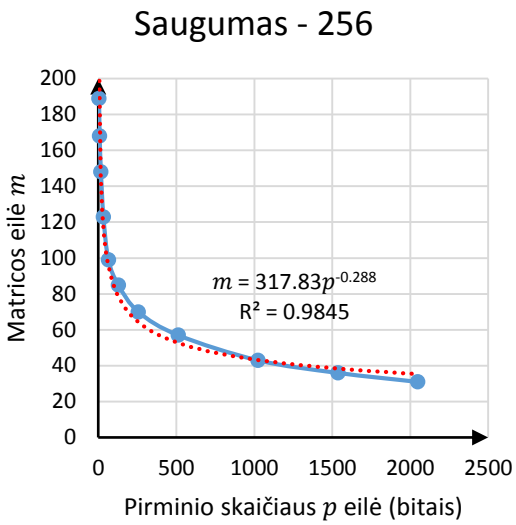
p eilė	Saugumo stiprumas									
	96		112		128		192		256	
	Vid. truk.	m eilė	Vid. truk.	m eilė	Vid. truk.	m eilė	Vid. truk.	m eilė	Vid. truk.	m eilė
4	0.47411	73	0.78980	87	0.94735	92	4.93439	145	17.8195	189
8	0.47105	62	0.78005	74	0.94335	78	4.93623	124	17.8120	168
16	0.47484	55	0.78221	68	0.94293	71	4.93538	109	17.8102	148
32	0.47086	45	0.78674	56	0.94255	60	4.93428	89	17.8118	123
64	0.47352	37	0.78267	43	0.94339	47	4.93542	71	17.8193	99
128	0.47577	31	0.78815	35	0.94419	39	4.93063	60	17.8180	85
256	0.47431	24	0.78294	27	0.94796	30	4.93608	49	17.8128	70
512	0.47048	17	0.78953	20	0.94877	23	4.93551	39	17.8178	57
1024	0.47878	11	0.78358	14	0.94030	16	4.93344	28	17.8165	43
1536	0.47506	9	0.78462	11	0.94570	13	4.93849	23	17.8111	36
2048	0.47404	8	0.78286	10	0.94697	12	4.93909	20	17.8126	31







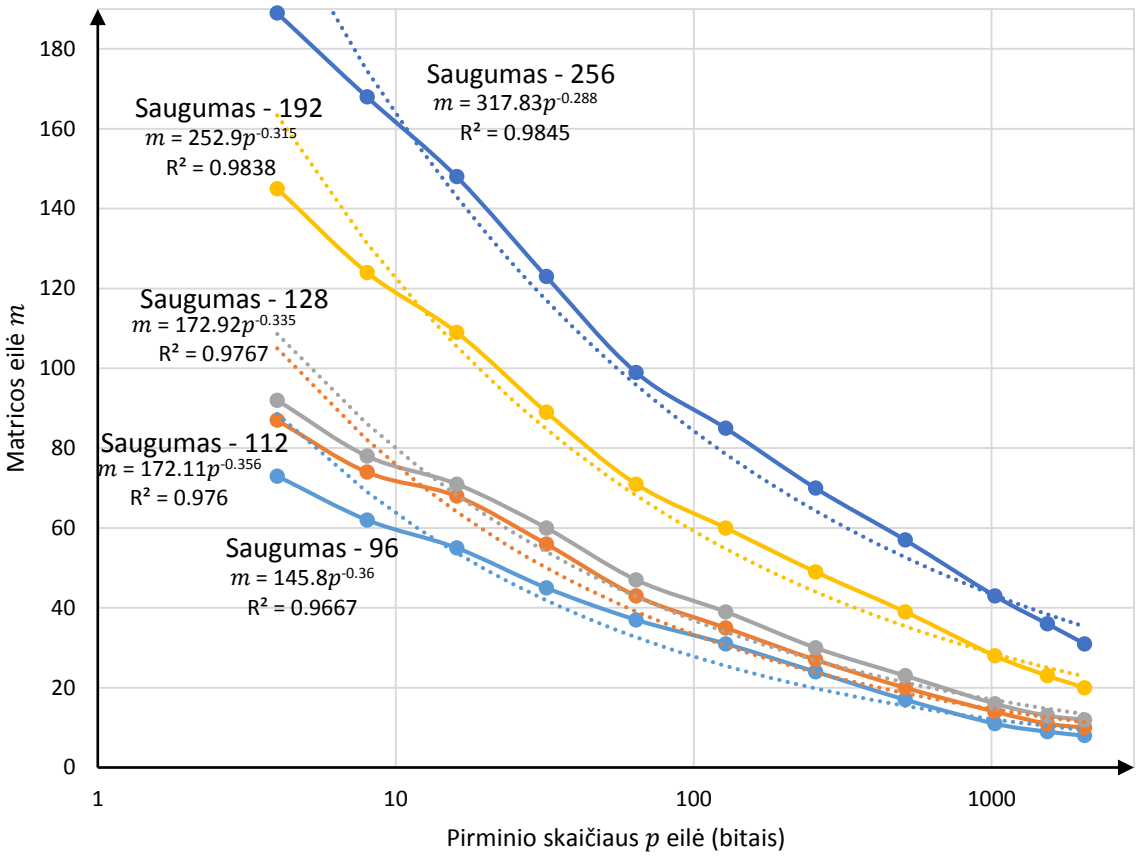




3.6 pav. Grafinis matricos eilės m sąryšio su pirminio skaičiaus p eile vaizdas

Šiek tiek patikslinsime **3.5 lentelės** sudedamasias dalis. Lentelėje „p eilė" – pirminio skaičiaus p eilė, matricų pusgrupėse, Vid. truk. – vidutinė DH algoritmo vykdymo trukmė sekundėmis. „m eilė" – kvadratinės matricos eilučių (stulpelių) skaičius. Skaičiai 96, 112, 128, 192, 256 parodo, prie kokio saugumo lygmens atliktas tyrimas. Su didesnėmis kaip 2048 bitų pirminio skaičiaus p eilėmis matricų pusgrupėse matricos eilių prie fiksuotos trukmės nebeieškojome, nes rezultatai tampa nekorektiški ir praranda prasmę. Kitaip tariant, tęsiant tokį procesą neberasime tokios matricos eilės, kad gautume vidutinę trukmę, artimą fiksuotajai elipsinėse kreivėse. **3.6 pav.** pateikiame visais penkiais saugumo stiprumo atvejais gautų rezultatų grafinius vaizdus. Tendencijų linijų ir aproksimuotų lygčių pagalba nustatėme, kad matricos ir pirminio skaičiaus eilės sąryšį apibūdina laipsninė priklausomybė.

Sekančiame paveiksle **3.7 pav.** visus saugumo lygmenis atvaizduojame viename grafike. Toks atvaizdavimo tikslas – palyginti matricos ir pirminio skaičiaus eilės sąryšio skirtumus, kuomet Difio ir Helmano apsikeitimo raktais algoritmas buvo vykdomas prie skirtingų saugumų. Šiame grafike pirminio skaičiaus eilė atvaizduota logaritminėje skalėje. Čia logaritminė skalė parenkama tam, kad ryškiau atsispindėtų visi taškai bei vaizdas būtų išsamesnis. Prie kiekvienos kreivės grafike nurodomas jo saugumo lygis bei kreivę nusakančios lygties aproksimuotos išraiškos.



3.7 pav. Bendras grafikas, kai p eilė logaritminėje skalėje

Svarbu paminėti, kad tyrimo rezultatams įtakos galėjo turėti ir kiti veiksniai. Visų pirma, DH algoritmas buvo tiriamas naudojant vieną personalinį kompiuterį, ir algoritmo vidutinės veikimo trukmės greičiai rasti jo galimybių ribose. DH algoritmą testuojant kitu personaliniu kompiuteriu tikėtina, kad trukmės bus labai panašios, o jų paklaidos – proporcingos. Nors kurdami algoritmą Visual Studio programinėje aplinkoje stengėmės kiek įmanoma optimizuoti skaičiavimus bei užtikrinti geriausią jų veikimo efektyvumą, reikia įvertinti, jog čia taip pat galimi nežymūs netikslumai. Vis dėlto rasti idealų ir visiškai optimizuotą tokios programos variantą – be galo sudėtingas uždavinys.

IŠVADOS IR REKOMENDACIJOS

Tiriant Difio ir Helmano apsikeitimo raktais algoritmą susipažinome su kirptografinių sistemų pagrindais. Apžvelgėme kitus kriptografiniame matematikos pasaulyje paplitusius apsikeitimo raktais algoritmus ir išsiaiškinome jų veikimo principą. Dirbant su DH algoritmu ir siekiant įgyvendinti jo realizaciją personaliniame kompiuteryje, nustatyti bendrąsias ir vidutines veikimo trukmes, sąryšius bei priklausomybes, susidūrėme su daug papildomų procedūrų bei metodų, kurie reikalavo atkaklumo ir pastangų. Teko susidurti ir su metodais, kurių galiausiai nepanaudojome – tokiais atvejais rasdavome

universalesnį atitikmenį. Pirmiausiai algoritmą vykdėme standartiškai, sveikųjų skaičių multiplikacinėse ir elipsinių kreivių plokštumos taškų grupėse. Modifikuotą DH algoritmą taip pat realizavome ir matricų pusgrupėse. Čia jo veikimo principas buvo paremtas ne kėlimo laipsniu operacija, o matricų daugyba. Tokiu atveju pats algoritmo saugumas buvo paremtas ne dikretinio logaritmo, o išsigimusios generatorių atitinkančios matricos savybėmis. Galiausiai pasiekėme užsibrėžtus tikslus, tuomet suformulavome išvadas.

- Pirminio skaičiaus eilę ir algoritmo vidutinę veikimo trukmę sveikųjų skaičių multiplikacinėse ir elipsinių kreivių plokštumos taškų grupėse sieja laipsninė priklausomybė, t. y. didinant pirminio skaičiaus eilę, vidutinė veikimo trukmė auga laipsniškai. Primename, kad didinant p eilę, su ja kartu didėja ir laipsnio rodikliai bei kiti sisteminiai parametrai.
- Palyginus Difio ir Helmano apsikeitimo raktais trukmes nustatėme, jog elipsinių kreivių plokštumos taškų grupėse šis algoritmas veikia greičiau negu sveikųjų skaičių multiplikacinėse grupėse. Pateikiame saugumo lygmenis bei kiek kartų prie šio lygmens algoritmas elipsinių kreivių atveju buvo vykdomas greičiau: 96 – 1,08 karto, 112 – 1,45 karto, 128 – 3,94 karto, 192 – 11,08 kartų, 256 – 25,79 kartų. Rezultatus sulyginti ir įrodyti, kurioje grupėje DH algoritmas vykdomas greičiau padėjo reikalavimai bei apibrėžti atitikmenys pirminio skaičiaus eilėms. Nustačius, kad elipsnių kreivių grupėse tiriamas algoritmas veikia žymiai efektyviau, darbą tęsėme būtent su šios rūšies grupėmis.
- Fiksavome vidutines Difio ir Helmano apsikeitimo raktais algoritmo veikimo trukmes prie penkių skirtingų saugumo lygmenų elipsinių kreivių plokštumos taškų grupėse. Tada prie šių laikų ištyrėme matricos eilės ir pirminio skaičiaus eilės sąryšį matricų pusgrupėse. Nustatėme dėsnį, kad matricos eilės priklausomybė nuo pirminio skaičiaus eilės yra laipsninė.

DH algoritmą vykdant nestandartiniu būdu – matricų pusgrupėse, skaičiavimus atlikome naudojant paprastą matricų daugybą. Siekdami dar efektyvesnio algoritmo veikimo, čia rekomenduotume naudoti Volker Strassen algoritmą, skirtą greitesnei matricų daugybai negu įprastinė [8]. Atlikinėjant Difio ir Helmano apsikeitimo raktais algoritmo vidutinės veikimo trukmės nustatymą, taip pat rekomenduojame atlikti kuo didesnį iteracijų skaičių, siekiant kaip įmanoma tikslesnių rezultatų. Jeigu norime tik patikrinti, kiek laiko užtruktų skaičiavimai prie atitinkamos pirminio skaičiaus eilės, galima atlikti mažiau testavimų, tačiau tokių atvejų negalėsime vadinti tiksliais ir naudoti tolimesnėje analizėje. Taip pat siekiant padaryti tikslesnes išvadas galima pritaikyti statistinius metodus, tikrinti hipotezes apie vidurkių lygybę. Jų pagalba galėtume užtikrinti tam tikrą DH algoritmo vidutinių veikimo trukmių patikimumą visais nagrinėtų grupių atvejais.

LITERATŪRA

1. Wikipedia, internetinė svetainė, *„Diffie-Hellman key exchange“*, [žiūrėta 2015-04-22], prieiga per internetą: <http://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange>.
2. Kriptografinės sistemos / Egidijus Sakalauskas, Narimantas Listopadsis, Gediminas Simonas Dosinas, Kęstutis Lukšys, Artūras Katvickis; Kauno technologijos univ. – Kaunas 2008.
3. Wikipedia, internetinė svetainė, *„Elliptic Curve Diffie-Hellman“*, [žiūrėta 2015-04-24], prieiga per internetą: <http://en.wikipedia.org/wiki/Elliptic_curve_Diffie%E2%80%93Hellman>.
4. Wikipedia, internetinė svetainė, *„MQV“*, [žiūrėta 2015-05-02], prieiga per internetą: <http://en.wikipedia.org/wiki/MQV>.
5. Wikipedia, internetinė svetainė, *„Discrete logarithm“*, [žiūrėta 2015-05-03], prieiga per internetą: <http://en.wikipedia.org/wiki/Discrete_logarithm>.
6. Daniel R. L. Brown, Standarts for Efficient Cryptography, *“Recommended Elliptic Curve Domain Parameters”*, Certicom Corp., 2010.
7. Algebrinių tiesinių operatorių struktūros / Gediminas Simonas Dosinas, Zenonas Navickas; Kauno technologijos univ. – Kaunas 2010.
8. Wikipedia, internetinė svetainė, *„Strassen algorithm“*, [žiūrėta 2015-05-13], prieiga per internetą: <http://en.wikipedia.org/wiki/Strassen_algorithm>.
9. Whitfield Diffie, Martin E. Hellman, Invited paper, *“New Directions in Cryptography”*, IEEE Transactions on Information Theory, 1976.
10. Andrew Byrne, Nicolas Meloni, Arnaud Tisserand, Emanuel Popovici, William Marnane, *“Comparison of Simple Power Analysis Attack Resistant Algorithms for an Elliptic Curve Cryptosystem”*, Journal of Computers, Academy Publisher, 2007.

1 priedas. Pagrindinių programos procedūrų išeities tekstai.

Čia pateikiamos Microsoft Visual Studio pagalba sukurtos programos išeities teksto pagrindinės dalys. Pilnas išeities tekstas nepateikiamas dėl jo didelės apimties. Dalis procedūrų buvo naudojamos tik individualios rūšies grupėse, dalis procedūrų – naudotos bendrai, visų rušių grupėse.

Bendrai naudotos procedūros:

(Millerio-Rabino pirmiškumo testas)

```
// Millerio-Rabino pirmiškumo testas
public static bool MillerRabin(BigInteger n, int k)
{
    BigInteger s = 0;
    BigInteger d = n - 1;
    while ((d & 1) == 0)
    {
        d >>= 1;
        s++;
    }

    if (s == 0)
        return true;             // sudėtinis skaičius

    Random rng = new Random();
    for (int i = 0; i < k; i++)
    {
        BigInteger a = NextRandomBigInt(rng, 2, n - 1);

        BigInteger x = BigInteger.ModPow(a, d, n);
        if (x == 1 || x == n - 1)
            continue;

        int j;
        for (j = 1; j < s; j++)
        {
            x = BigInteger.ModPow(x, 2, n);
            if (x == 1)
                return true;             // sudėtinis skaičius
            if (x == n - 1)
                break;
        }
        if (j == s)
            return true; // sudėtinis skaičius
    }
    return false;        // pirminis skaičius su didele tikimybe
}
```

(Atsitiktinio BigInteger generavimas)

```
// atsitiktinio BigInteger generavimas
public static BigInteger NextRandomBigInt(Random rng, BigInteger minValue, BigInteger maxValue)
{
    BigInteger n;

    var len = maxValue.ToByteArray().Length;
    var buffer = new byte[len];

    do
    {
        rng.NextBytes(buffer);
        n = new BigInteger(buffer);
    }
    while (n < minValue || n >= maxValue);
    return n;
}
```

(Apsauga nuo nelogiškų ženklų įvedimo)

```
// leidžiama įvesti tik skaičius
private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
{
    char ch = e.KeyChar;

    if(!char.IsDigit(ch) && ch != 8 && ch !=46)
    {
        e.Handled = true;
    }
}
```

(Saugaus pirminio skaičiaus generavimas)

```
// saugaus pirminio skaičiaus p generavimas
q = NextRandomBigInt(rng, biDidziausiaReiksme >> 1, biDidziausiaReiksme);

// generuojamas atsitiktinis BigInteger tipo tarpinis pirminis skaičius q
while (MillerRabin(q, 3) == true)
{
   q = NextRandomBigInt(rng, biDidziausiaReiksme >> 1, biDidziausiaReiksme);
}

// generuojamas atsitiktinis Biginteger tipo pirminis skaičius p
p = 2 * q + 1;
while (MillerRabin(p, 3) == true)
{
    q = NextRandomBigInt(rng, biDidziausiaReiksme >> 1, biDidziausiaReiksme);

    while (MillerRabin(q, 3) == true)
    {
        q = NextRandomBigInt(rng, biDidziausiaReiksme >> 1, biDidziausiaReiksme);
    }
    p = 2 * q + 1;
}
```

(Tikrinimas, ar užpildyti privalomi laukeliai)

```
// metodas, tikrinantis ar visur įvesti reikalingi duomenys
private void TikrintiDuomenis()
{
    // jei duomenų nėra, mygtukas apsaugotas nuo paspaudimo
    if (((comboBox1.Text != "") && (textBox11.Text != "")) ||
         (textBox8.Text != "") && (textBox1.Text != "") &&
         (textBox2.Text != "") && (textBox3.Text != ""))
    {
        if(!skaiciuoja)
            button1.Enabled = true;
    }

    // įvedus ir ištrynus duomenis, mygtukas vėl apsaugomas nuo paspaudimo
    else
    {
        button1.Enabled = false;
    }
}
```

Tik sveikųjų skaičių multiplikacinėse grupėse naudotos procedūros:

(Generatoriaus generavimas)

```
// generuojamas generatorius g
g = NextRandomBigInt(rng, 1, p);
while ((BigInteger.ModPow(g, q, p) == 1) || (BigInteger.ModPow(g, 2, p) == 1))
{
    g++;
    if (g == p)
    g = 2;
}
```

(Pirminio skaičiaus ir generatoriaus patikrinimas)

```
// metodas, tikrinantis ar parametrai p, g yra teisingi
private void Tikrinti_p_g(BigInteger p, BigInteger g)
{

    // tikriname ar buvo sėkmingai sugeneruotas pirminis skaičius p
    if (MillerRabin(p, 3) == false)
    {
        label14.ForeColor = Color.Green;
        label14.Text = "(Sėkmingai įvestas/sugeneruotas pirminis skaičius)";
    }
    else
    {
        label14.ForeColor = Color.Red;
        label14.Text = "(Nesėkmingai įvestas/sugeneruotas pirminis skaičius)";
    }


    // randame tarpinį pirminį skaičių q pagal įvestą p
    BigInteger q;
    q = p >> 1;

    // tikriname ar buvo sėkmingai sugeneruotas generatorius g
    if ((g < p) && (BigInteger.ModPow(g, q, p) != 1) && (BigInteger.ModPow(g, 2, p) != 1))
    {
        label15.ForeColor = Color.Green;
        label15.Text = "(Sėkmingai įvestas/sugeneruotas generatorius)";
    }
    else
    {
        label15.ForeColor = Color.Red;
        label15.Text = "(Nesėkmingai įvestas/sugeneruotas generatorius)";
    }
}
```

(Individualių slaptųjų raktų pasirinkimo patikrinimas)

```
// metodas, tikrinantis ar parametrai x, y yra teisingi
private void Tikrinti_x_y(BigInteger p, BigInteger x, BigInteger y)
{
    // tikriname ar buvo sėkmingai sugeneruotas slaptasis raktas x
    if (x < p)
    {
        label16.ForeColor = Color.Green;
        label16.Text = "(Sėkmingai įvestas/sugeneruotas x)";
    }
    else
    {
        label16.ForeColor = Color.Red;
        label16.Text = "(Nesėkmingai įvestas/sugeneruotas x)";
    }

    // tikriname ar buvo sėkmingai sugeneruotas slaptasis raktas y
    if (y < p)
    {
        label17.ForeColor = Color.Green;
        label17.Text = "(Sėkmingai įvestas/sugeneruotas x)";
    }
    else
    {
        label17.ForeColor = Color.Red;
        label17.Text = "(Nesėkmingai įvestas/sugeneruotas x)";
    }
}
```

Tik elipsinių kreivių plokštumos taškų grupėse naudotos procedūros:

(P ir Q sumavimą atliekantiprocedūra)

```
        // metodas skaičiuojantis P ir Q sumą grąžina porą (x3, y3)
        public static KeyValuePair<BigInteger, BigInteger> Suma_PQ(BigInteger x1, BigInteger y1, BigInteger x2,
BigInteger y2, BigInteger p, BigInteger pa, BigInteger pb)
        {
            // tarpiniai parametrai
            BigInteger lam, x3, y3;

            // kai P lygu Q
            if ((x1 == x2) && (y1 == y2))
            {
                lam = ((3 * BigInteger.Pow(x1, 2) + pa) % p) * BigInteger.ModPow(2 * y1, p - 2, p);
            }

            // kai P nelygu Q
            else
            {
                lam = ((y2 - y1) % p) * BigInteger.ModPow(x2 - x1, p - 2, p);
            }

            // x3 padarome teigiamu
            x3 = (BigInteger.Pow(lam, 2) - x1 - x2) % p;
            if (x3 < 0)
            {
                x3 = x3 + p;
            }

            // y3 padarome teigiamu
            y3 = ((lam * (x1 - x3)) - y1) % p;
            if (y3 < 0)
            {
                y3 = y3 + p;
            }
            return new KeyValuePair<BigInteger, BigInteger>(x3, y3);
        }
```

(nP sandaugą atliekanti procedūra)

```
        // metodas skaičiuojantis nP sandaugą ir grąžina porą (x3, y3)
        public static KeyValuePair<BigInteger, BigInteger> Sandauga_nP(BigInteger n, BigInteger x1, BigInteger y1,
BigInteger x2, BigInteger y2, BigInteger p, BigInteger pa, BigInteger pb)
        {
            BigInteger x3, y3; // rezultatai x3, y3
            BigInteger xd, yd; // daugikliai xd ir yd
            bool priskirta = false;

            // R = O
            x3 = y3 = 0;

            // D = P
            xd = x1;
            yd = y1;

            while (n > 0)
            {
                // jei nelyginis laipsnis
                if (n % 2 != 0)
                {
                    // sudetis R = R + D
                    if (priskirta)
                    {
                        var SumosPora1 = Suma_PQ(x3, y3, xd, yd, p, pa, pb);
                        x3 = SumosPora1.Key; // rezultatas x3
                        y3 = SumosPora1.Value; // rezultatas y3
                    }
                    else
                    {
                        x3 = xd;
                        y3 = yd;
                        priskirta = true;
                    }
                }
```

```
            n = n / 2;

            // dvigubinimas D = D + D
            var SumosPora2 = Suma_PQ(xd, yd, xd, yd, p, pa, pb);
            xd = SumosPora2.Key; // daugiklis xd
            yd = SumosPora2.Value; // daugiklis yd
        }
        return new KeyValuePair<BigInteger, BigInteger>(x3, y3);
    }
```

(Parametrų tikrinimo procedūra)

```
    // metodas, tikrinantis ar parametrai p, pa, pb, x, y yra teisingi
    private void TikrintiParametrus(BigInteger p, BigInteger pa, BigInteger pb,
        BigInteger x, BigInteger y)
    {

        // tikriname ar buvo sėkmingai sugeneruotas pirminis skaičius p
        if (MillerRabin(p, 3) == false)
        {
            label14.ForeColor = Color.Green;
            label14.Text = "(Sėkmingai įvestas/sugeneruotas pirminis skaičius)";
        }
        else
        {
            label14.ForeColor = Color.Red;
            label14.Text = "(Nesėkmingai įvestas/sugeneruotas pirminis skaičius)";
        }

        // tikriname ar buvo sėkmingai sugeneruoti parametrai pa ir pb
        if ((pa < p) && (pb < p) && (((4 * BigInteger.Pow(pa, 3) + 27 * BigInteger.Pow(pb, 2))) % p != 0))
         {
            label25.ForeColor = Color.Green;
            label25.Text = "(Sėkmingai įvesti/sugeneruoti pa ir pb)";
        }
        else
        {
            label25.ForeColor = Color.Red;
            label25.Text = "(Nesėkmingai įvesti/sugeneruoti pa ir pb)";
        }

        // tikriname ar buvo sėkmingai sugeneruotas slaptasis raktas x
        if (x < p)
        {
            label16.ForeColor = Color.Green;
            label16.Text = "(Sėkmingai įvestas/sugeneruotas x)";
        }
        else
        {
            label16.ForeColor = Color.Red;
            label16.Text = "(Nesėkmingai įvestas/sugeneruotas x)";
        }

        // tikriname ar buvo sėkmingai sugeneruotas slaptasis raktas y
        if (y < p)
        {
            label17.ForeColor = Color.Green;
            label17.Text = "(Sėkmingai įvestas/sugeneruotas x)";
        }
        else
        {
            label17.ForeColor = Color.Red;
            label17.Text = "(Nesėkmingai įvestas/sugeneruotas x)";
        }
    }
```

Tik matricų pusgrupėse naudotos procedūros:

(Atsitiktinės m-tos eilės matricos generavimas)

```
// atsitikrinės matricos generavimas
public static BigInteger[,] RandomMatrix(int m, BigInteger p, Random rng)
{
    BigInteger[,] no = new BigInteger[m, m];

    for (int i = 0; i < m; i++)
    {
        for (int j = 0; j < m; j++)
        {
            no[i, j] = NextRandomBigInt(rng, 0, p);
        }
    }
    return no;
}
```

(Atsitiktinės m-tos eilės išsigimusios matricos generavimas)

```
// atsitiktinės išsigimusios matricos (su proporcingomis eilutemis) generavimas
public static BigInteger[,] RandomSingular(int m, BigInteger p, Random rng)
{
    BigInteger[,] no = new BigInteger[m, m];
    int eilute1 = rng.Next(m);
    int eilute2 = rng.Next(m);

    while (eilute1 == eilute2)
    {
        eilute2 = rng.Next(m);
    }

    for (int i = 0; i < m; i++)
    {
        for (int j = 0; j < m; j++)
        {
            no[i, j] = NextRandomBigInt(rng, 0, p);
        }
    }

    for (int j = 0; j < m; j++)
    {
        no[eilute2, j] = no[eilute1, j];
    }
    return no;
}
```

(Atsitiktinės matricos vaizdavimo procedūra)

```
// atsitiktinės matricos išvedimas
public static string DisplayMatrix(int m, BigInteger[,] M)
{
    string matrixString = "";
    for (int i = 0; i < m; i++)
    {
        for (int j = 0; j < m; j++)
        {
            matrixString += M[i, j].ToString();
            matrixString += " ";
        }
        matrixString += Environment.NewLine;
    }
    return matrixString;
}
```

(Matricų daugybą atliekanti procedūra)

```
// matricų daugyba
public static BigInteger[,] MatrixMultiplication(int m, BigInteger p, BigInteger[,] M1, BigInteger[,] M2)
{
    BigInteger[,] R; // rezultatas

    R = new BigInteger[m, m];
    for (int i = 0; i < m; i++)
    {
        for (int j = 0; j < m; j++)
        {
            R[i, j] = 0;
            for (int k = 0; k < m; k++)
            {
                R[i, j] = (R[i, j] + M1[i, k] * M2[k, j]) % p;
            }
        }
    }
    return R;
}
```

(Determinantą skaičiuojanti procedūra)

```
// metodas skaičiuojantis determinantą
public static BigInteger Determinant(BigInteger[,] M)
{
    BigInteger det = 0;
    BigInteger[,] tempArr = new BigInteger[M.GetLength(0) - 1, M.GetLength(1) - 1];

    if (M.GetLength(0) == 2)
    {
        det = M[0, 0] * M[1, 1] - M[0, 1] * M[1, 0];
    }
    else
    {
        for (int i = 0; i < 1; i++)
        {
            for (int j = 0; j < M.GetLength(1); j++)
            {
                BigInteger subdet = Determinant(FillNewArray(M, i, j));
                if (j % 2 != 0) subdet *= -1;
                det += M[i, j] * subdet;
            }
        }
    }
    return det;
}
```

(Papildoma procedūra determinanto skaičiavimui)

```
// metodas masyvo užpildymui, naudojamas skaičiuojant determinantą
public static BigInteger[,] FillNewArray(BigInteger[,] originalArr, int row, int col)
{
    BigInteger[,] tempArray = new BigInteger[originalArr.GetLength(0) - 1, originalArr.GetLength(1) - 1];

    for (int i = 0, newRow = 0; i < originalArr.GetLength(0); i++)
    {
        if (i == row)
            continue;
        for (int j = 0, newCol = 0; j < originalArr.GetLength(1); j++)
        {
            if (j == col)
                continue;
            tempArray[newRow, newCol] = originalArr[i, j];
            newCol++;
        }
        newRow++;
    }
    return tempArray;
}
```

(Pirminio skaičiaus patikrinimas)

```
// metodas, tikrinantis ar p yra pirminis skaičius
private void TikrintiPirmini(BigInteger p)
{

    // tikriname ar buvo sėkmingai sugeneruotas pirminis skaičius p
    if (MillerRabin(p, 3) == false)
    {
        label14.ForeColor = Color.Green;
        label14.Text = "(Sėkmingai įvestas/sugeneruotas pirminis skaičius)";
    }
    else
    {
        label14.ForeColor = Color.Red;
        label14.Text = "(Nesėkmingai įvestas/sugeneruotas pirminis skaičius)";
    }
}
```

2 priedas. Pirminiai skaičiai ir jų eilės.

Pateikiami programos parametrų (saugių pirminių skaičių) pavyzdžiai sveikųjų skaičių, elipsinių kreivių plokštumos taškų grupių ir matricų pusgrupių atvejais. Nurodėme tik po vieną skaičių prie atitinkamo jo dydžio. Tai padeda suprasti programos veikimo efektyvumą atskirais atvejais, kuomet skaičiavimai atliekmai prie skirtingų saugaus pirminio skaičiaus eilių.

Pirminių skaičių eilių pavyzdžiai sveikųjų skaičių grupėse ir matricų pusgrupėse:

(Pirminio skaičiaus eilė **4** bitai)
11

(Pirminio skaičiaus eilė **8** bitai)
227

(Pirminio skaičiaus eilė **16** bitų)
65123

(Pirminio skaičiaus eilė **32** bitai)
3507680843

(Pirminio skaičiaus eilė **64** bitai)
13245904981395369359

(Pirminio skaičiaus eilė **128** bitai)
333518236514997499542541021375467189323

(Pirminio skaičiaus eilė **256** bitai)
72145874316064054800439589037980053503520805476979202368728023634612488056759

(Pirminio skaičiaus eilė **512** bitai)
10819133930664116898299879074924266924496811043413292152379860465553705401626875525928865354232560188701606536814836749916761410066948776531722427103211307

(Pirminio skaičiaus eilė **1024** bitai)
97897222807730140113475061814265290581279920253850140345947776632937862128483219113296899700521562124886446747966500453586795836854529888277069716607046843531325475871309361095573301131818170722268355907882282079578856682293134886632699024556639298480886131239503861964455805979341271031530663018453419829333871

(Pirminio skaičiaus eilė **1536** bitai)
18393427609999516489873695191636997066444273000958293843407243585795695061360358692416969868278237695596987223171124601432382958077771657741633075502402533377826437725678994754771395615699701548395321385376549351438501163988343696272344230008813489059341819080848634450583065044690325894902135811157621788930954101861627283740036936420008679382067126880588613492054175443828567216576273174887235597215330519407161425573436477510058515990573852304804904614314619431

(Pirminio skaičiaus eilė **2048** bitai)
25496322253902375560880394667118549214945470603906454678869110914325865510788415811616472026112687705341230458421421993177271579844862626453274398942623983601740447951318329137057565011095954000654218157868976326527752326510935780376095693802750592503884553297877684628651021048183687343502272583182411255934570390264901422219507230429053725312999197941949984559282305979947846719693613252066535990159750264070544231570637080401577549338667613321283013110918028825993108649297022889295138226202474948513058782100069512063960833913883206993702403432027510761648849067584123195828607670587000973357142152841256618771187

(Pirminio skaičiaus eilė **2560** bitai)
3129917083726072721366725182698186008596210630445635699178552687420035281315909663765803126258441588518309485208073330301582492451607454651333329852471364569792153976404142839093067389742643950426649395177629951505140868342709437167812320910935537991076263376913359788132113104730549679105519245699678092655488746567019736598029494257400719631848879166115012575160681119180000342183353130177451125109028114941098504715195535989724463166801158715295845647132851450493333657153457071387905745238979130377820399463095934645932704697572401722239775592518891813958391334143983349801304972969815740680040406135246372891352407998883053058496085754261839599720194490374762719712975467431686828484676491024703762624131567529632668943054638866360304521927278381684896642772146347031
(Pirminio skaičiaus eilė **3072** bitai)
5809605995369958062791915965639201402176612226902900533702900882779736177890990861472094774477339581147373410185646378328043729800750470098210924487866935059164371588168047540943981644516632755067501626434556398193186628990071248660819361205119793693985433297036118232914410171876807536457391277857011849897410207519105333355801121109356897459426271845471397952675959440793493071628394122780510124618488232602464649876850458861245784240929258426287699705312584509625419513463605155428017165714465363094021609290561084025893662561222573202082865797821865270991145082200656978177192827024538990239969175546190770645685893438011714430426409338676314743571154537142031573004276428701433036381801705308659830751190352946025482059931306571004727362479688415574702596946457770284148435989129632853913921179974726326930781131298864873993477969827727846158652326212896569442842168246113187097645351525073541163447037699985141483438071
(Pirminio skaičiaus eilė **3584** bitai)
4871788847392238247242063214456922134761186375362163357168272345716196398546227405594617229821317082065004583583348005562674813853162201278086649227727247235158196471094145190952421901886557104968150419168517958553740605271157998585830948121369385699046898811815642018178083617887245769486588739561916476848891647432745712483112682455004725302935101322091361144793469926237067475139230193684692901773257278052526404967683431833817622275179420635587800030094521785742341931076695579718311421977270458506746070868981802791117531771759121811470020101898664341141117669949412185065191573894328358147967735550239499383888792302793294009676548793072129895232778391858677806469555950325533753930176404448102779841575410972241236508267972868212115153231013966341528914868005526104815004411202944771486282365070316511818182429923261662723040703376582260513728036905944848051763178163292596492843479983824617275584370472842854370727209474270215126697345678965408861255247576949188957737317310569864317870929758557955371003062136950836692992605228686662741016769490182980993060821088444410923
(Pirminio skaičiaus eilė **4096** bitai)
7141372590471007986587346427749236129516962875459141718536001556767361277967978033955028750023179977265188553661273652858197892928147814207249301703058627796686991683124519164836421737906084273137759502198183898232726456510175896903758894993250293429036468498325171982404668870584840906877685244828888724585620057314637933453157548702873317213975241808166288643294799339135686161926304196823038783866480799096525781359568556718601997021106694283883683754846471495409751505659706473184533622045162698010565995857227932599672051858976709467600759148703806723870800826213898661328664282604782355953899583888792302793294009676548793072129895232778391858677806469555950325533753930176404448102779841575410972241236508267972868212115153231013966341528914868005526104815004411202944771486282365070316511818182429923261662723040703376582260513728036905944848051763178163292596492843479983824617275584370472842854370727209474270215126697345678965408861255247576949188957737317310569864317870929758557955371003062136950836692992605228686662741016769490182980993060821088444410923
(Pirminio skaičiaus eilė **6144** bitai)
3375152182143856118451852315996741233006489780574184654817389047442942990132667244520323510191916548396419435946099488106208938789376281404425743820443257394108301482700609025892587516101809632773233580059583191597601420882230400732784813273493329788580321367526156496260334045722077682632250005809131096725397661997398803366366638518815521265626807950172622336969342799980413446781012077235649859645532366527400517575471969335854905274504119509592366013711954148258884879224599915203456315881034776553083676995718335598586395591169999570824515035017543533352697525287753332500527176569576894926734950469293596134095086603716860086302051544539652689091299099784588919052383463057789440565460681441902442399956419060521629604697347879024654313800186078316526964529288062740879011035175920059192178561473199006205896719435014765345518490882366607110905303449152556221163232127426440691921134648766635695850239231304591744215610985029636895406718880766308249227315984267542266259489684372223916445411015900506239419267909716320331208988978180868987431623710347617992356201449023892203230133

094214639142912013460631252196369642616835915410143442392753407356909977322220697587739633908763605
465157552805170421605254873028981223116697996794475304536003993426970327144585495912859394539490349
812481143223223672386450425159844478907889178235763300191516965686543141530585475920913660145501438
196851700683437001046776090411663697600809334136054989623820777788455998349074759534307874462013845
673285306752757929623548837708069008271836857183534695747316805206219445409477346190351771800579730
226525710321965982292591948757099947097217931541586865157485072742241813169487971046010682120152329
216914824963468544136987197501906011027052744810505432398151306860736010763045122845492184598460460
822535967624338274190600890294170448712183160209231099889157071175 67

(Pirminio skaičiaus eilė **7680** bitai)
464926350015275954530119301251063924079223767371226854380562633246690839952029204822348763400472657
876029683151878550708180467073533182150052078798305063561883809052678859042841612723447979822325376
888475197532649792643824144129865298497483246112059015008839471282143672669664336037116995564399644
394443759726396473389727627398382309202929812298919047509229555398497308843257943232678476398931077
444339023624161631225365120954091601103378049468876318807618254876060336418250801158771011266894707
823428676217570880919632201738828270757124131438223292542153838242357956159842782379362252057092561
870872648539754532276190705266920446241004206854160063802517197738340293978142324303150894382773271
773274845610567917311771284310475093882128559205271334651201834278049067298277023238004618948005488
723224397899993655645567363212363043802274639614255257675586797284097482834118965509532305896251338
185766558748224051008980157552455943615122226694627085490821814283250364558095895601817845591092470
076529951200601702476502064500957458532567915561901686752996970383947548395155087533116191832224398
396951645858223266333019805826902055830400736289554774298304405311629837317118480204555362074451838
004392666678888302079638740123033038839162112096810239090659590783421913190795437451594292369973881
767116802613605647492318738906221713861847446796245758997891315218539198166139988773311769363387598
200290873251238839896822099500849205042508977579010454134573922599071194849338866302764031639758882
847160488543065633873675804946927278253202191212687136792049835220563211930606307463660572729083486
076170957421611160388870676676155361854731287509584465574675857580950482508735320534815036190849252
293134847205325708688973633887365610343565107913403385999432510784756498923690331250889729027510870
373792767906314638401145895401764011153015211500467788241490313285911946250085014059534454976601717
994787025400094404570593511918071478580136741705632132818510792379122371955300363710875818374589204
350952936796391088078270483925337751739696027737231527111417616528875725866928407872096827698557297
903088612936894116545972778805355630991637165248060026564500012481236230336793540726355996460919975
739751580034157725394120958880389037184974766859212431644306348172212946544421740965787975292956257
022556864604802353698891071348340 99

(Pirminio skaičiaus eilė **8192** bitai)
652213827904073750635424149547480531760656582977408203455616878788640380281182735113609219930795287
240923364789330654392814303805310584871481192962277862800235614553628791098101603299224132344179835
373893957421997088209246818980020952778524772489157108508644653190599366650945358592659588461335762
755336032218803848313646378927060015141967623900758934912782932479021427700303355027666305743996673
960230121307579023954918316450282681823144237467929105126905479432305261823369574659477936988387749
202756578063438577627205748741847260533121980265514472017194619508413883455873560053984406354980564
806605348006113277750025917627774324262661944919234537489511980789090857368666395018050414834403479
440416804865341379674689305113915728408440776663318590965330683330188151502560142394543653608927833
898076233411566237817090879999028279322996507915093761829212653925927074550100637404875707949876586
549923700599551630194249535004581558546631186134278854238022040320821245131658739743172994172383921
764428793350867334302496058182254053954205026392896412837006658713351406471544469041953311179875757
659714908373180333781267006781122401570924541143058537157613028519042769982123471117364823836331668
434611785992352216832604279597046143456957390692115654736048594212850789753681730765648404891591928
241170672427426260075750843638841063174946728616303834117624324365069454857619650725136760763292808
590321675975805137033698730963953921172089196995837922279391423893974318511757072515007780775621310
352500326089488484026255628020399563062259746803619432665274863062846783705469751746698858348868522
691849597741548669701092772524338416465271106982584756066730886764438347618000286582889950417819266
284092780668473253376449915772835725747909791672790721718762014439285101127692317689643419755229182
616864012077540580894167679668059148081516886200265447923350511381669100377908709281423030852331808
199276600371189189347090113556819596501902420678172718375721835182448793746026604427683777285653017
735483314700224908403327028298609714557728164966270403079212858550791002079188210855482909539985867
667355219610150789858049270300935817617317511140579486904461947045397057038182146853065395890177901
093501118440036103925470629102588928147268451536768259256450670395002828060625132741913612437308988
686513776391328751712877950510121577789452011972634715386290355838938016644846549996000379985856285
220257404489461687265179763284142302898467165536959321773050975758218755616844005485958483

(Pirminio skaičiaus eilė **15360** bitai)
24555139988895094145877470227010566763912026951019982074246819308393183771037848199572840734386475788776513118234661643494663664523926686409452366171524856515585564301725700723998377377463289280886019918442998999477534513127774063401860471479128561388104939823721877795232692664770605924903795372642906964559459551611899866652265926960990165009518340807933756954616025246112468862396888446734935912720591204128646889811607758154925256839104231894038837263394853238369811588310879637781723472587806917472675188424685579042521133852617188450548580695397904449452694839533434683295919333271795565280324389955603339613022119963653541222985680841637219640009649833332081830394710963570162166330828415003386103183638093953343081796833340458323449083890614138948844627564549450481252943448972753617790662596970136112110197290740332595946516595553154143654258232022894345759530008986667038341877492622964449990097823494940865439690007303239959837728316222200480939049206481147373283141641762616881674244789494681492499846540440154207171636127339342764615188115985381523416253859093633974299410575486805720012697148123341522848606507311630903313327769259361546418205639820371280531157226792932579813918588352093050944924443198486267640020389328429228295751312759513110598222287325289459425865926746634289596112233847598758006517767119622376272967301902512991688482822895034895536954696124540410453617275538231715273157610277421939273296475976315378159478473638960671157681696432820840018993241338738845844868127765457506575489995865577443259833222757792428042410149079033802850882464454283661246521839377553124029624811365899857988095642698648335824907095732808987404939569798217453519260905312637394240987935235713483861469807021840335159695484110346906957638059724629171951354226130222993825942117924055268747149086342181154155958549640294200110791972473508636281924159209294743355499950822399928158348722677810090724201578984082006885916410589970188190937010268807334166517895736780805774947187708128623144551282146100324654459340163243539639615559320619002587155122035950647087087200198148347926624981930305745642030812738013662053951705104111534211214941677718781937389679964915459867202367739543188880648739751397746584473521375292874551591445541664708173905228615127931681750112188220885554940941945823731793360336606424436183894598862705586030251357898272013953024252876468585812147182473503758550107834348428547622308490815791026833814232080638442222599166109249835399788126453575828279079905997912496310750832147358391845585907953233576342934563993060853324018451594567953882545810914919107559271337643219068545664611471380504603846583303559965782242567711711815388860046285401884806929237204995905282512556543610168897966278339390124501361863676269651279825464391775818126070471732198922469136886815854284155322353402353662309435381475676069018946861299971234451572589212558168258316382735812354885554903883789189021110311658483006766426929195751740405338711860082888624506348619460268604081260175119821523904672612564532460496583287691501586396337758392250863070415965147895293865166354484443993327076066101019308436183671353369047482017429313531558605295885296358696882864635527782109424576247425578722159889203111399427352448764584899972641589857365864325222576817451984517130641798258073746099126348511512422492250959780838650319654636743023892423070321072417467366649085355810264186705371916842911006812500699073346884192587453502978069290508412283666024021060381642197140591108812584667371121636684987459921680618114537343381660888722120358033031072821230187854926517797235420165574867366217548518721036617738563231188257121811295259219932573605021399315317041487271728740648266365343755642479152363251676239698808839290481183208624698449431001608018223452550126720619837233111573529365971815084140338914351365617228707998147823303104609489837805967536967629702765028171536560050227089543166497289232080391128439137458180390927811559964811639491292289692384550783332662830617179442685708984183563088355099171933945333621128896906450343416843045347763131164771452558506705664835115786621602151686112348603168576820626211360075490497776479836616127131962885384380722976973880535806532302361470359924070543020153392211609736272915024919012452900436429953770933965040467310272967954631270118334424592118101146104675898674482138597065737145947561073234678684703325307797565660478331880890722682872934890423280798315851756907568769391308566826655744428453077215500610607846424412846989987137205224826823576698229858487140571585503897573541629254887806677039009493060630545581204575688528281065667769188224759987077992826 11

(Pirminio skaičiaus eilė **16384** bitai)
91753702613364789822553466753978085672899679326063017201577326776776383819516034546529540580505482135709071802472126442326341653453248130584094712523991961141739574077894222445885107334906622290579885045955910936988841753692190660542341539915198779974333786290153250873259816529244470539485361218830498067365464759605646690115178155006332385220519847974482284776206570946522672586082700797361102670121860948552239584443663609951922436034690751326795190851958540263079252487160555323643791317255487473276211178296203072567075858598066830651713285829238630694436300196767563033489612417225825860715626044043168599365542486929152248251974480816830805502156339275092373167107454445031651797689420147786569144901890634926575466108359787273413386947336671064562651132438311974254586376658246205716356020217146712490385433977324888811598789166585708905319913709974181972692949206771742363995919828260933753436269305526651351634137497772093628066326292015890555752948455224472979493624813876115377911778592205490916871187034761223141182181057717992424685679984099691504973946714512266520504002983313169029419385203635902936054867120196268081525030632872127243611827381758342237351455242418 2

292184561357110198519707016427355474895435725919171080873272192919356930700422660639035794960707757
356075689061274313380030666142416528532154203103619026615191494975933921805444454268404134387528123
970625184392996995614093972390775694716718347961312023254641591049787748565335127309703481164960992
975612599439655497460597482436997000702967327568193689324182970846068327756899193191991414987736346
835721927237814450918988941267915576104394228364294262921733847383475884763194646775678955931487926
252891192353766958721747945568505094779144354980978439736558654691242790917446869726104022820685552
619874301068779937669555000667420555437371705013546463492582123732860937204936524398200025111879205
509841118068114020007708954483305668316070404950611753188861451267593286604377070837045120554725328
029465365057215778441955974573599970272984054757220630864521003650570345794484219200089220245161320
338844590551692346703874313583739786905554701756181764341481572850366160825888789434536584529139534
218156993644655072560738387788461845827889678160639537151684286123211793859471140965124970965472945
057858723060158654676919296565788256899835630791243936231428474655569392722356878993537618681324925
433831305730359645693346650299591975396763427410526610021464724691124734133830489703201050138329792
122995873829880059011869050630649823098359819845239226947110637421332254254672258482097717377677035
754970382651609934172325660186949382755140368725101554676059857195981706246089890733384166702511640
345894443595156216912099613753850829459569728428482592888326389014059146334398193349428990750304923
665707051271230187518907324016216832796513941222384430490044475837433201063969097770667860014241112
648237577551905065793279535143412803611980920755872539566870004102858587097840590132602838634033543
779794068101417131969176866647566702558440393449315035234513501677581534804421837818539526031061502
043677080714976607175825511646925976508699559214141140128768150042246690981973432145867539990309164
057826251175620416214458526065287492722837773830631974273543548040200855596436304709788300162177679
238742175809098324990666786094426142130494925547549607972578071572283076628722695520628638372491976
497107070584979540813522551050767891537275437804733343636853171805090424968858044912640076743528313
011359684035787091366740707717836367614981579346723894564521921558987391049923369750275182672433575
488296140268010137582548668296623490978327134505256125011704668317782225321229209446842353390560312
234052925435180702627817202823796695230425254751285276680618025301447127532284412253965581338425673
867613247946968516132783459443337919518333986347535043496469575748697477423071986932974748742546363
627195542289358029336197820851470857055745384953370529800715735949822010834176232161358240338790645
644114418694703719250701689846539331775622326482333854580373353586013523427953134594729468230717031
018589698300089108866671802574607846556021839960320279993764339331768885318978143713009161010122721
653380541704364231835511013560207507710312751704194765341610877823025599464599498158827280014438471
079364368030395688087265797223248551370061765826297598426650931200339138406144107775542382057218618
241013502878844214030782007577933258323450603936302166988429214408244952690957806562874657288853354
147134259502436600485371573304883940303009636231922118805238619271682239224366108892880305741110534
791794525469715536443717842158510570298440749415847839661505379659826996124935154490815571526945 63
734419941679805797021944505536736075843150895167560706708161496901017420466966876358648535382092 48
133995047352580762955151306362222789822878211316604600065282508004068980007056947670833610360793618
663252040225506115886019497732279118544961469958969012465533799578695909455416107

Pirminių skaičių eilių pavyzdžiai elipsinių kreivių plokštumos taškų grupėse:

(Pirminio skaičiaus eilė **192** bitai)
6277101735386680763835789423207666416083908700390324961279

(Pirminio skaičiaus eilė **224** bitai)
26959946667150639794667015087019630673557916260026308143510066298881

(Pirminio skaičiaus eilė **256** bitai)
115792089210356248762697446949407573530086143415290314195533631308867097853951

(Pirminio skaičiaus eilė **384** bitai)
394020061963944792122790401001436138050797392704654466679482934042457217714968703290472660882589380
01861606973112319

(Pirminio skaičiaus eilė **521** bitas)
686479766013060971498190079908139321726943530014330540939446345918554318339765605212255964066145455
49772963113914808580371219879997166438125740282911150571 51