KAUNO TECHNOLOGIJOS UNIVERSITETAS

FUNDAMENTALIŲJŲ MOKSLŲ FAKULTETAS

TAIKOMOSIOS MATEMATIKOS KATEDRA

Paulius Vitkus

VIENOS VIENKRYPTĖS FUNKCIJOS SAUGUMO IR EFEKTYVUMO ANALIZĖ

Magistro darbas

Vadovas
prof. dr. E.Sakalauskas

KAUNAS, 2009

KAUNO TECHNOLOGIJOS UNIVERSITETAS

FUNDAMENTALIŲJŲ MOKSLŲ FAKULTETAS

TAIKOMOSIOS MATEMATIKOS KATEDRA

TVIRTINU
Katedros vedėjas
doc. dr. N.Listopadskis
2009 06 04

VIENOS VIENKRYPTĖS FUNKCIJOS SAUGUMO IR EFEKTYVUMO ANALIZĖ

Matematikos magistro baigiamasis darbas

Vadovas
prof. dr. E.Sakalauskas
2009 05 22

Recenzentas
doc. dr. G.S.Dosinas
2009 06 03

Atliko
FMMM-7 gr. stud.
P.Vitkus
2009 05 22

KAUNAS, 2009

KVALIFIKACINĖ KOMISIJA

Pirmininkas: Leonas Saulis, profesorius (VGTU)

Sekretorius: Eimutis Valakevičius, docentas (KTU)

Nariai: Algimantas Jonas Aksomaitis, profesorius (KTU)

Vytautas Janilionis, docentas (KTU)

Vidmantas Povilas Pekarskas, profesorius (KTU)

Rimantas Rudzkis, valdybos pirmininko patarėjas (DnB NORD Bankas)

Zenonas Navickas, profesorius (KTU)

Arūnas Barauskas, vice-prezidentas projektams (UAB „Baltic Amadeus“)

Vitkus P. The security and efficiency investigation of certain on-way function: Master's work in applied mathematics / supervisor prof. E.Sakalauskas; Department of Applied mathematics, Faculty of Fundamental Sciences, Kaunas University of Technology. – Kaunas, 2009. – 43 p.

SUMMARY

For creating cryptographic protocols secure and efficient one-way function is needed. In this decade we saw several one-way functions based on braid groups being proposed. However recently those functions were shown to have security flaws and the use of braid groups in cryptography was set under suspicion. In this work we take a new look at braid group cryptography and propose a certain one-way function based in Burau representation level of braid groups. Function parameters are set with the respect to efficient computer memory usage and it can be compared with the security of AES encryption system with 128 bit cryptographic keys. The security of proposed one-way function is based on the complexity of solving underdefined systems of multivariate quadratic polynomial equations over finite fields. This problem is NP-complete even if all the equations are quadratic and the field is $GF(2)$. The classical algorithm for solving such systems is Buchberger's algorithm for constructing Groebner bases. It is shown that in our case Buchberger's algorithms running time is at least exponential.

TURINYS

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

ALGORITMŲ SĄRAŠAS

ĮVADAS

Nuo seniausių laikų žmonės bandė paslėpti informaciją nuo nepatikimų akių, pavyzdžiui, senovės Egipte žmonės naudojo nestandartinius hieroglifus. Per ilgus amžius susiformavo mokslas kaip šifruoti duomenis – kriptografija. Šiais laikais kriptografija yra paremta fundamentaliosiomis matematikos žiniomis. Be informacijos šifravimo jos funkcijos yra praplėstos ir apima informacijos integralumą bei siuntėjo/gavėjo autentifikaciją.

Svarbios informacijos šifravimas visą laiką buvo vienas iš svarbiausių kelių į sėkmę kare, finansuose ar asmeniniame gyvenime. Šiuo metu tai tapo masinėmis informacinės visuomenės technologijomis. Tobulėjant matematiniams metodams ir skaičiavimo technikai senesni šifrai tapo nesaugūs, todėl kuriami nauji šifravimo metodai.

Sudarant kriptografinius protokolus dažnai reikia duomenis užšifruoti negrįžtamai. Šiam tikslui pasiekti yra naudojamos vienkryptės funkcijos. Vienkryptė funkcija – tai tokia funkcija, kurią paprasta suskaičiuoti su visais pradiniais duomenimis, tačiau yra sudėtinga sudaryti jos atvirkštinę funkciją. Šiame kontekste „paprasta“ ir „sudėtinga“ yra nusakomi skaičiavimų sudėtingumo teorijos prasme.

Pastarajame dešimtmetyje pasirodė vienkryptės funkcijos paremtos braidų grupėmis (2, 16). Nepraėjo daug laiko ir šios funkcijos buvo sukritikuotos (18, 19), o braidų grupės kriptografinis panaudojimas tapo abejotinas. Šiame darbe bandoma naujai pasižiūrėti į braidų grupės pritaikymą kriptografijoje ir sudaryti vienkryptę funkciją Burau vaizdavimo lygmenyje.

Pirmoje darbo dalyje yra pateikiamas vienos vienkryptės funkcijos apibrėžimas bei teorija reikalinga sudaryti ir ištirti šią funkciją. Antroje darbo dalyje atliekamas jos saugumo ir efektyvumo tyrimas bei nustatomi funkcijos parametrai. Programinės įrangos realizacija ir vartotojo instrukcija yra pateikta trečioje darbo dalyje. Pabaigoje pateikiama rezultatų interpretacija ir išvados. Studento publikuoti pranešimai (24, 25) pateikti **4 Priede**.

1. TEORINĖ DALIS

1.1. VIENKRYPTĖS FUNKCIJOS SUDARYMAS

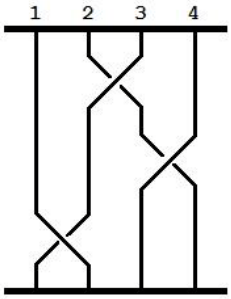
1.1.1. BRAIDŲ GRUPĖ

Pirmasis braidų (*angl*. braid – sruoga, kasa) grupes apibrėžė Artinas (4, 6, 14, 17). Nekomutatyvi grupė B_n vadinama n-braidų grupe, kai $n \geq 2$, kurią aprašo n-1 generatorius σ_i (i = 1, 2, ..., n-1) bei vienetinis elementas e ir yra tenkinami sąryšiai:

$$\sigma_i\sigma_j = \sigma_j\sigma_i, \qquad |i - j| > 1;$$

$$\sigma_i\sigma_j\sigma_i = \sigma_j\sigma_i\sigma_j, \qquad |i - j| = 1.$$

Artino generatoriai, dar vadinami elementariaisiais braidais, turi paprastą geometrinę interpretaciją. Įsivaizduokime, kad n-braidą sudaro n juostų, einančių iš viršaus į apačią, tada generatorius σ_i reiškia, kad susikerta i ir i+1 juostos. Sutarta, kad juosta i eina po juosta i+1, kai turime σ_i, ir juosta i eina virš juostos i+1, kai turime σ_i^{-1}. Braidų žodis $w \in B_n$, kitaip dar vadinamas braidu arba žodžiu, yra elementariųjų braidų ir jų atvirkštinių seka. Pavyzdžiui, braidas $w = \sigma_2\sigma_3\sigma_1^{-1}$ pavaizduotas **1.1. paveiksle**.



1.1. pav. Braidų žodis $w = \sigma_2\sigma_3\sigma_1^{-1}$.

Teigiamas braidas yra toks grupės B_n elementas, kurį galima užrašyti kaip žodį, sudarytą iš elementariųjų braidų nenaudojant jų atvirkštinių. Visų teigiamų braidų aibę grupėje B_n žymėsime B^+_n.

Braido $w \in B_n$ ilgis yra žymimas $|w|$ ir nusako elementariųjų braidų bei jų atvirkštinių, sudarančių braidą w, skaičių. Du braidai yra lygūs, jeigu jie yra identiškai atvaizduojami elementariaisiais braidais ir jų atvirkštinėmis. Du teigiami braidai yra lygūs, jeigu jie yra identiškai atvaizduojami elementariaisiais braidais arba jie gali būti transformuoti į identiškus atvaizdavimus grupės sąryšių pagalba.

Fundamentaliuoju braidu $\Delta_n \in B_n$ vadiname tokį braidą, kurį nusako sąryšis

$$\Delta_n = \sigma_1 \sigma_2 \ldots \sigma_{n-1} \Delta_{n-1}, \tag{1.1}$$

ir $\Delta_1 = \sigma_1$.

Su visais elementariaisiais braidais galioja savybė:

$$\sigma_i \Delta_n = \Delta_n \sigma_{n-i}. \tag{1.2}$$

Fundamentaliojo braido kvadratas $\Delta_n^2 = \Delta_n \Delta_n$ yra grupės B_n centro elementas, t. y.

$$w \Delta_n^2 = \Delta_n^2 w, \forall w \in B_n. \tag{1.3}$$

Viena iš matematinių problemų braidų grupėse yra jungtinio paieškos problema (*angl.* conjugator search problem). Ji apibrėžiama taip: turime du braidus $a, b \in B_n$ ir $b = xax^{-1}$, reikia rasti nežinomą braidą $x \in B_n$. Tačiau jungtinio paieškos problema kriptografiniuose protokoluose buvo supaprastinta (23): turime du braidus $a, b \in B_n$ ir $b = xay$, reikia rasti nežinomus braidus $x, y \in B_n$, kurie priklauso komutatyviems pogrupiams.

1.1.2. BURAU VAIZDAVIMAS

Aukščiau pateiktas braidų grupės apibrėžimas yra abstraktus ir vadinamas braid grupės apibrėžimu raiškos lygmenyje, t. y. grupė vaizduojama aibe generatorių bei aibe ryšių tarp jų. Kaip ir daugelis kitų grupė B_n gali būti atvaizduota matricų grupe M virš lauko F.

Braidų grupę B_n į matricų grupę M galima atvaizduoti Burau vaizdavimo (4) pagalba:

$$\rho_n : \sigma_i \rightarrow \beta_i, \sigma_i \in B_n, \beta_i \in M(n, F);$$

$$\beta_i = I_{i-1} \oplus \begin{pmatrix} 1-t & t \\ 1 & 0 \end{pmatrix} \oplus I_{n-i-1}, t \in F.$$

Lengva patikrinti, kad galioja braidų grupės ryšiai:

$$\beta_i \beta_j = \beta_j \beta_i, \qquad |i - j| > 1;$$

$$\beta_i \beta_j \beta_i = \beta_j \beta_i \beta_j, \quad |i - j| = 1.$$

Daug metų buvo manoma, kad šis vaizdavimas yra tikras. Pastaruoju metu buvo įrodyta (3, 10), kad Burau vaizdavimas yra nepatikimas, kai $n \geq 5$. Tačiau praktiškai šį vaizdavimą galima sėkmingai taikyti, nes jo branduolys yra pakankamai mažas.

Pabrėšime, kad Burau vaizdavimas taip pat yra tam tikra vienkryptė funkcija dėl to, kad kol kas nėra sugalvotas praktinis atvirkštinio vaizdavimo algoritmas. Vadinasi, tos matematinės problemos, kurios yra išsprendžiamos braidų grupės raiškos lygmenyje, gali būti praktiškai neišsprendžiamos vaizdų lygmenyje.

Šiame darbe siūlomai vienkryptei funkcijai sudaryti buvo nuspręsta modifikuoti aukščiau pateiktą Burau vaizdavimą. Braidų grupės elementai yra vaizduojami ne į matricų aibę virš lauko aibę, o į matricų aibę virš keleto kintamųjų daugianarių (*angl*. multivariate polynomial) žiedo. Pasirinktas žiedas yra $Z_p[x_1, x_2, \dots, x_m]$. Taigi žiedą sudaro daugianariai su m skirtingų kintamųjų, kurių koeficientai yra virš lauko Z_p, o p – pirminis skaičius.

Modifikuotą Burau vaizdavimą galima užrašyti:

$$\rho_n : \sigma_i \rightarrow \beta_i,\ \sigma_i \in B^+_n,\ \beta_i \in M(n, Z_p[x_1, x_2, \dots, x_m]);$$

$$\beta_i = I_{i-1} \oplus \begin{pmatrix} 1-t & t \\ 1 & 0 \end{pmatrix} \oplus I_{n-i-1},\ t \in Z_p[x_1, x_2, \dots, x_m].$$

Pasirinkus tam tikrą daugianarį $t \in Z_p[x_1, x_2, \dots, x_m]$ braidų grupė yra atvaizduojama į matricas ne virš viso žiedo $Z_p[x_1, x_2, \dots, x_m]$, o virš tam tikro požiedžio $T \subset Z_p[x_1, x_2, \dots, x_m]$. Požiedį T galima surasti žemiau pateikto **1.1. algoritmo** pagalba:

1.1. algoritmas. Požiedžio $T \subset Z_p[x_1, x_2, \dots, x_m]$ sudarymas.

$T = \{0, 1, t, 1\text{-}t\}$
Kartoti
    $U \neq \varnothing$
    $P = \{\{p, q\} \mid p, q \in T\}$
    Kartoti
        $\{p, q\}$ = pora iš aibės P
        $P = P - \{\{p, q\}\}$
        $u = p\cdot(1\text{-}t) + q$
        $v = p\cdot t$
        Jeigu $u \notin T$ ir $u \notin U$ tai $U = U \cup \{u\}$
        Jeigu $v \notin T$ ir $v \notin U$ tai $U = U \cup \{v\}$
    Kol $P \neq \varnothing$
    $T = T \cup U$
Kol $U \neq \varnothing$

Akivaizdu, kad šis algoritmas yra teisingas, nes sudarinėjant braidų žodį vaizdo lygmenyje, mes vienetinę matricą I_n dauginame su atitinkamomis matricomis β_i. Po pirmosios tokios daugybos žodyje yra elementai $T = \{0, 1, t, 1\text{-}t\}$. Nauji elementai žodyje gali atsirasti dauginant formuojamą žodį iš naujos matricos β_i. Kitaip tariant iš bet kokių dvieju aibės T elementų suformavus vektorių – eilutę ir jį padauginus iš matricos $\begin{pmatrix} 1-t & t \\ 1 & 0 \end{pmatrix}$ pirmojo bei antrojo stulpelio. Jeigu gaunamas naujas elementas, tai jis įtraukiamas į aibę T. Procesas yra tęsiamas, kol nebepavyksta gauti naujų elementų.

1.1.3. DAUGIANARIŲ KODAVIMAS

Kadangi visi skaičiavimai yra atliekami kompiuterio pagalba, daugianarius reikia efektyviai saugoti kompiuterio atmintyje. Efektyvus saugojimas reiškia, kad daugianariui saugoti reikalingos kompiuterio atminties būtų naudojama kaip galima mažiau, o veiksmus galima būtų atlikti greitai ir paprastai. Kad tai galėtume realizuoti pirmiausia aptarkime kokiomis savybėmis pasižymi daugianariai virš žiedo $Z_p[x_1, x_2, \ldots, x_m]$.

Kai daugianariai yra virš lauko Z_p, o p – pirminis skaičius, tai kiekvienam kintamajam galioja Ferma teorema (21): $x_i^{\,p} \bmod p = x_i^{\,0} \bmod p = 1$, $x_i \in Z_p$. Vadinasi, kiekvieno kintamojo laipsniai gali įgyti reikšmes nuo 0 iki p-1.

Kiekvieną vienanarį (*angl*. monomial) virš žiedo $Z_p[x_1, x_2, \ldots, x_m]$ nusako m skirtingų kintamųjų. Kiekvieną vienanarį taip pat galima įsivaizduoti kaip m-matį vektorių, kurio pirmoji komponentė nusako pirmojo vienanario kintamojo laipsnį, antroji – antrojo vienanario kintamojo laipsnį ir t. t. Visos šio vektoriaus komponentės įgyja reikšmes nuo 0 iki p-1. Taip sudarytą vektorių galima įsivaizduoti kaip skaičių, užrašytą p-tainėje skaičiavimo sistemoje, o jį pavertus į mums įprastą dešimtainę skaičiavimo sistemą mes gausime numerį, kuris vienareikšmiškai nusakys kiekvieną vienanarį.

Galėdami vienareikšmiškai sunumeruoti visus įmanomus vienanarius, mes kiekvienam vienanariui saugoti kompiuterio atmintyje galime skirti vieną bitą. Tokiu atveju bitas su reikšme „1“ reiškia, kad toks vienanaris yra daugianaryje, o bitas su reikšme „0“ reiškia, kad tokio vienanario daugianaryje nėra. Akivaizdu, kad skirtingų vienanarių daugianaryje gali būti p^m, tiek pat reikės ir bitų jiems išsaugoti. Tačiau kompiuterio atmintyje mažiausiais duomenų vienetas kurį galima išskirti yra baitas (8 bitai), o baito viduje galima pasiekti kiekvieną bitą. Vadinasi, norint išsaugoti visus daugianario vienanarius kompiuterio atmintyje mums reikes $\left\lceil \frac{p^m}{8} \right\rceil$ baitų.

Žinome, kad daugianario narį sudaro koeficientas padaugintas iš vienanario. Koeficientai, kaip ir daugianario kintamieji, virš lauko Z_p gali įgyti reikšmes nuo 0 iki p-1. Norint išsaugoti daugianarį kompiuterio atmintyje mums reikia išsaugoti tuos narius, kurių koeficientai yra nuo 1 iki p-1. Vadinasi, galime atskirai išsaugoti visus daugianario vienanarius, kurie yra padauginti iš „1“, „2“ ir t. t. Taip kompiuterio atmintyje užkuoduotam daugianariui išsaugoti reikės $(p-1)\times\left\lceil \frac{p^m}{8} \right\rceil$ baitų. Tada matricos iš grupės $M(n, Z_p[x_1, x_2, \dots, x_m])$ kompiuterio atmintyje užims:

$$a = n^2 \times (p-1)\times\left\lceil \frac{p^m}{8} \right\rceil \qquad \textbf{(1.4)}$$

baitų.

Kaip atliekami veiksmai virš lauko Z_p plačiau galima paskaityti literatūros šaltinyje (21). Trumpai paminėsime tiktai, kad norint kaip įmanoma greičiau atlikti daugybos ir sudėties veiksmus, reikia sudaryti Keilio lenteles šiems veiksmams virš lauko Z_p ir atlikti paiešką jose.

1.1.4. SIŪLOMA VIENKRYPTĖ FUNKCIJA

Turime dvi matricas $A, B \in M(n, Z_p[x_1, x_2, \dots, x_m])$ ir

$$B = XAY, \qquad \textbf{(1.5)}$$

reikia rasti nežinomas matricas $X, Y \in M(n, Z_p[x_1, x_2, \dots, x_m])$, $XY = YX$.

Pirmiausia, sugeneruojame braidų žodžius $a, x, y \in B_n$. Kiekvienas jų gaunamas sudauginus k atsitiktinių elementariųjų braidų $\sigma_i \in B_n$. Tada atliekamas aukščiau aptartas modifikuotas Burau vaizdavimas:

$$\rho_n : a \rightarrow A, a \in B_n, A \in M(n, Z_p[x_1, x_2, \dots, x_m]);$$

$$\rho_n : x \rightarrow X, x \in B_n, X \in M(n, Z_p[x_1, x_2, \dots, x_m]);$$

$$\rho_n : y \rightarrow Y, y \in B_n, Y \in M(n, Z_p[x_1, x_2, \dots, x_m]);$$

Norint užtikrinti, kad matricos X ir Y komutuotu, reikia, kad braidai x ir y raiškos lygmenyje priklausytu braidų grupės B_n komutatyviems pogrupiams (tenkintų lygybę $xy = yx$). Kaip konstruoti komutatyvius pogrupius galima rasti literatūros šaltiniuose (12, 13).

Kaip matome, siūloma vienkryptė funkcija $B = XAY$ yra aukščiau minėtos matematinės problemos $b = xay$ braidų grupėje išraiška matricų $M(n, Z_p[x_1, x_2, \ldots, x_m])$ grupėje. Matematinė problema $b = xay$ braidų grupėje kol kas nėra išspręsta, tačiau yra manoma, kad ši problema yra išsprendžiama. Todėl kriptografiniai protokolai paremti braidų grupėmis laikomi nepatikimi.

Vienkryptė funkcija tik pasinaudoja braidų grupių savybėmis, pavyzdžiui, komutatytviu pogrupiu sudarymu. Atlikus Burau vaizdavimą yra pereinama į matricų grupę, kurioje atliekami visi skaičiavimai. Kadangi nėra žinomas atvirkštinis Burau vaizdavimas, tai pereiti iš matricų grupės atgal į braidų grupę yra neįmanoma. Jeigu bus sugalvotas matematinės problemos $b = xay$ braidų grupėje sprendimas, tai vienkryptės funkcijos atvirkštinės radimo problema nebūtinai bus išspręsta.

Braidų atvaizdavimas į matricų grupę virš daugianarių žiedo $Z_p[x_1, x_2, \ldots, x_m]$ buvo pasirinktas todėl, kad sudėties ir daugybos veiksmus virš žiedo galima atlikti lengvai. Tuo tarpu sprendžiant matricines lygtis dažnai reikia dalybos veiksmo, kas atitinka daugybą iš atvirkštinio elemento. Jeigu atvirkštinis elementas neegzistuoja, reikia ieškoti bendrų daugiklių, kas padinina skaičiavimų apimtį.

Daugianariai virš lauko Z_p buvo pasirinkti todėl, kad tokiame lauke sudėties ir daugybos operacijos yra atliekamos efektyviai taikant Keilio lenteles (21). O kaip pamatysime 1.2.2. skyrelyje keleto kintamųjų daugianarių dėka gaunamos sudėtingesnės lygčių sistemos, kurių pagalba galima bandyti spręsti siūlomos vienkryptės funkcijos atvirkštinės radimo problemą.

Vienkryptės funkcijos atvirkštinės radimo problemą galima būtų spręsti taip: laisvai pasirenkama matrica Y, mums reikia rasti matricą X, išsprendus paprastą matricinę lygtį. Norint išvengti tokio scenarijaus į vienkryptės funkcijos apibrėžimą buvo įtrauktas reikalavimas, kad matricos X ir Y turi tarpusavyje komutuoti. Šiuo atveju laisvai pasirinkus matricą Y ir radus paprastos matricinės lygties sprendinį X mažai tikėtina, kad šios matricos komutuos tarpusavyje. Dar kartą priminsime, kad yra algoritmiškai sunku rasti paprastos matricinės lygties sprendinį, kai matricų elementai yra virš žiedo, nes praktiškai neegzistuoja dalybos operacija.

1.1.5. VIENKRYPTĖS FUNKCIJOS TAIKYMAS

Vienkryptė funkcija gali būti pritaikyta konstruojant kriptografinį raktų apsikeitimo protokolą. Tarkime turime du subjektus: Aldoną ir Bronių. Jie nori apsikeisti informacija taip, kad joks trečiasis subjektas negalėtų jos perskaityti. Šiam tikslui jiems reikia užšifruoti informaciją kriptografinių raktų pagalba. Šiuo atveju kriptografiniai raktai irgi yra tam tikra informacija, kurią gali teisingai susiskaičiuoti tik Aldona ir Bronius.

Tarkime, yra paskelbti ir visiems viešai žinomi parametrai: daugianaris $t \in Z_p[x_1, x_2, \ldots, x_m]$, matrica $A \in M(n, Z_p[x_1, x_2, \ldots, x_m])$ ir dviejų tarpusavyje komutatyvių braidų grupės B_n pogrupių C, $D \subset B_n$ generatoriai. Suprantama, šių pogrupių elementai turi tenkinti sąlygą:

$$c \cdot d = d \cdot c, \forall c \in C, \forall d \in D.$$

Aldona atlieka skaičiavimus: atsitiktinai parenka $x_a \in C$ ir $y_a \in D$, randa $X_a = \rho_n(x_a)$ ir $Y_a = \rho_n(y_a)$ bei suskaičiuoja $K_a = X_a A Y_a$.

Bronius atlieka skaičiavimus: atsitiktinai parenka $x_b \in D$ ir $y_b \in C$, randa $X_b = \rho_n(x_b)$ ir $Y_b = \rho_n(y_b)$ bei suskaičiuoja $K_b = X_b A Y_b$.

Aldona ir Bronius vienas kitam nusiunčia dydžius K_a ir K_b. Dabar jie gali suskaičiuoti bendrą kriptografinį raktą K:

$$K = X_a K_b Y_a = X_a X_b A Y_b Y_a = X_b X_a A Y_a Y_b = X_b K_a Y_b.$$

Kenkėjas žino viešai paskelbtus parametrus ir gali sužinoti dydžius K_a ir K_b, kai jais viešai apsikeičia Aldona ir Bronius. Jeigu pavyks surasti bet kokius dydžius X'_a ir Y'_a, kurie tenkintų komutatyvumo sąlygą ir lygtį $K_a = X'_a A Y'_a$, tai kenkėjas sužinos bendrą kriptografinį raktą K:

$$X'_a K_b Y'_a = X'_a X_b A Y_b Y'_a = X_b X'_a A Y'_a Y_b = X_b K_a Y_b = K.$$

Bendrą kriptografinį raktą K analogiškai galima gauti suradus bet kokius dydžius X'_b ir Y'_b, kurie tenkintų komutatyvumo sąlygą ir lygtį $K_b = X'_b A Y'_b$. Tačiau ieškant dydžius X'_a ir Y'_a arba X'_b ir Y'_b reikia išspręsti vienkryptės funkcijos atvirkštinės radimo problemą.

1.2. REIKALAVIMAI VIENKRYPTEI FUNKCIJAI

1.2.1. VIENKRYPTĖS FUNKCIJOS PARAMETRAI

Sudarius vienkryptę funkciją reikia parinkti tinkamus jos parametrus: n – braidų ir matricų grupės eilę, p – pirminį skaičių, nusakantį lauką Z_p, m – daugianarių virš lauko Z_p kintamųjų skaičių. Taip pat reikia parinkti daugianarį $t \in Z_p[x_1, x_2, \ldots, x_m]$ ir parametrą h, nusakantį iš kiek elementariųjų braidų turi būti generuojami braidų žodžiai $a, x, y \in B_n$.

Parinkus vienkryptės funkcijos parametrus gaunamos matricos virš žiedo $Z_p[x_1, x_2, \ldots, x_m]$ turi užimti kaip įmanoma mažiau kompiuterio atminties. O vienkryptės funkcijos saugumas turi bent prilygti pasaulyje plačiai naudojamai vienkryptei funkcijai AES (*angl*. Advanced Encryption Standard) su 128 bitų kriptografiniais raktais (8). Norint pilnai perrinkti visus įmanomus funkcijos AES raktus,

reikia atlikti 2^{128} patikrinimų. Mes reikalausime, kad perrinkant visas įmanomas matricas X arba Y, taip pat reikėtų atlikti nemažiau kaip 2^{128} patikrinimų.

Literatūros šaltinyje (5) nurodyta, kad atlikti 2^{128} elementų perrinkimą idealiomis sąlygomis prireiktų apie 10^{13} metų, kas yra praktiškai neįmanoma.

1.2.2. KELETO KINTAMŲJŲ KVADRATINIŲ DAUGIANARIŲ LYGČIŲ SISTEMA

Norint išspręsti vienkryptės funkcijos (1.5) atvirkštinės radimo problemą, reikia rasti bet kokias matricas X', $Y' \in M(n, Z_p[x_1, x_2, \ldots, x_m])$, tenkinančias funkciją ir jos sąlygas. Išanalizuokime, kaip kenkėjas galėtų bandyti spręsti šią problemą.

Matricą X sudaro $n \times n$ daugianarių, kurių kiekvienas gali turėti p^m skirtingų vienanarių. Kiekvienas vienanaris gali būti padaugintas iš nežinomo koeficiento. Gauname, kad matricoje X yra $p^m \cdot n^2$ nežinomų skirtingų kintamųjų. Analogiškai tiek pat nežinomų kintamųjų yra matricoje Y, o visoje matricinėje lygtyje $B = XAY$ yra

$$k = 2 \cdot p^m \cdot n^2 \qquad \textbf{(1.6)}$$

nežinomų skirtingų kintamųjų.

Matricinę lygtį $B = XAY$ galima perrašyti: $Q = XAY - B = 0$. Matricoje Q taip pat gali būti $p^m \cdot n^2$ skirtingų vienanarių, o prie kiekvieno vienanario turime lygtis su aukščiau minėtais $2 \cdot p^m \cdot n^2$ nežinomais skirtingais kintamaisiais. Vadinasi, turime

$$s = p^m \cdot n^2 \qquad \textbf{(1.7)}$$

skirtingų lygčių, kurių kiekviena turi būti lygi nuliui. Akivaizdu, kad visos šios lygtys bus bi-kvadratinės.

Kenkėjas gali supaprastinti šią lygčių sistemą atsižvelgęs į tai, kad atlikus Burau vaizdavimą matricos gaunamos ne virš viso žiedo $Z_p[x_1, x_2, \ldots, x_m]$, o virš tam tikro požiedžio $T \subset Z_p[x_1, x_2, \ldots, x_m]$. **1.1. algoritmo** pagalba galima susigeneruoti visą požiedį T. Tokiu atveju nežinomi skirtingi kintamieji yra ne prie visų matricos X ir Y vienanarių, o tik prie tų, kurie priklauso požiedžiui T.

Atsižvelgus į požiedį T galima gauti mažiau nei $2 \cdot p^m \cdot n^2$ kintamųjų ir mažiau nei $p^m \cdot n^2$ lygčių, bet visos šios lygtys vis tiek išlieka kvadratinės.

Paminėsime, kad vienkryptę funkciją AES su 128 bitų kriptografiniais raktais galima užrašyti keleto kintamųjų kvadratinių daugianarių lygčių sistema. Literatūros šaltinyje (8) nurodyta, kad tokią lygčių sistemą sudaro 8000 išretintų lygčių su 1600 nežinomųjų. Vienkryptės funkcijos atveju parenkant parametrų reikšmes taip pat bus siekiama, kad keleto kintamųjų kvadratinių daugianarių lygčių sistemoje būtų apie 1600 nežinomųjų.

Literatūros šaltinyje nurodyta (15), kad uždavinys – išspręsti keleto kintamųjų kvadratinių daugianarių lygčių sistemą virš bet kurio lauko – priklauso NP-complete uždavinių klasei.

1.2.3. GRIOBNERIO BAZĖS

Keleto kintamųjų kvadratinių daugianarių lygčių sistemas siūloma spręsti Griobnerio bazių (*angl*. Groebner bases) algoritmo pagalba. Šiame darbe bus pateikti algoritmai, reikalingi sudaryti Griobnerio bazes, plačiau apie jas literatūros šaltiniuose (1, 9). Paminėsime, kad Griobnerio bazės neišsprendžia lygčių sistemos, o ją supaprastina, kad būtų galima sėkmingai spręsti kitais metodais.

Kad galėtume sėkmingai taikyti Griobnerio bazių radimo algoritmą, pirmiausia reikia apsibrėžti tvarką, kaip galima rikiuoti keleto kintamųjų daugianarių narius. Praktikoje dažniausiai naudojamos trys daugianario narių rikiavimo tvarkos: leksikografinė tvarka (*angl*. lexicographic order), laipsninė ir leksikografinė tvarka (*angl*. graded lexicographic order), laipsninė ir atvirkštinė leksikografinė tvarka (*angl*. graded reverse lexicographic order).

Tarkime, kad turime daugianarį su kintamaisiais x_1, ... , x_k ir su koeficientais virš lauko. Šiuos daugianario koeficientus galima surikiuoti taip:

$$x_1 > x_2 > \ldots > x_k.$$

Kaip buvo aptarta 1.1.3. skyrelyje kiekvieną keleto kintamųjų daugianario vienanarį galima vienareikšmiškai atvaizduoti k-mačiu vektoriumi.

Tegul α ir β būna du vektoriais išreikšti vienanariai:

- Leksikografinė tvarka: $\alpha > \beta$ tada ir tik tada, jeigu kairiausia nelygi nuliui vektoriaus $\alpha - \beta$ reikšmė yra teigiama.
- Laipsninė ir leksikografinė tvarka: $\alpha > \beta$ tada ir tik tada, jeigu $|\alpha| > |\beta|$, arba $|\alpha| = |\beta|$ ir kairiausia nelygi nuliui vektoriaus $\alpha - \beta$ reikšmė yra teigiama.
- Laipsninė ir atvirkštinė leksikografinė tvarka: $\alpha > \beta$ tada ir tik tada, jeigu $|\alpha| > |\beta|$, arba $|\alpha| = |\beta|$ ir dešiniausia nelygi nuliui vektoriaus $\alpha - \beta$ reikšmė yra neigiama.

Pasirinkus tam tikrą keleto kintamųjų daugianario narių rikiavimo tvarką ir turint nelygų nuliui daugianarį $f \in Z_p[x_1, x_2, \ldots, x_k]$ galime apibrėžti tokias funkcijas:

- DV(f) – grąžina didžiausią daugianario f vienanarį, kuris padaugintas iš nelygaus nuliui koeficiento.
- DK(f) – grąžina nelygią nuliui koeficiento reikšmę, kuri yra padauginta iš didžiausio daugianario f vienanario.
- DN(f) – grąžina daugianario f narį, kurio koeficientas nelygus nuliui ir vienanaris yra didžiausias daugianario f vienanaris.

Tarkime, turime daugianarių aibę $F = \{f_1, f_2, \ldots, f_s\}$ ir bet kokį daugianarį f iš žiedo $Z_p[x_1, x_2, \ldots, x_k]$. Daugianarį f galima išskaidyti taip: $f = q_1 \cdot f_1 + \ldots + q_s \cdot f_s + r$. Koeficientus q_i galima interpretuoti kaip daugianario f dalybos iš daugianario f_i dalmenį, o r – kaip daugianario f dalybos iš aibės F liekaną. Dalybos liekana r gali būti lygi nuliui arba daugianariui, kuris nesidalina iš aibės F elementų. Žemiau pateiksime daugianario f dalybos iš aibės F liekanos radimo **1.2. algoritmą**.

1.2. algoritmas. Daugianario f dalybos iš aibės F liekanos radimas.

$r = 0$
$p = f$
Kartoti kol $p \neq 0$
    $i = 1$
    dalinti = taip
    Kartoti kol $i \leq s$ ir *dalinti* = taip
        Jei DN(f_i) dalina DN(p) Tai
            $p = p$ - (DN(p) / DN(f_i)) · f_i
            dalinti = ne
        Priešingu atveju
            $i = i + 1$
    Jei *dalinti* = ne Tai
        $r = r$ + DN(p)
        $p = p$ - DN(p)
Grąžinti r

Turėdami du daugianarius $f,\ g \in Z_p[x_1,\ x_2,\ \ldots\ ,\ x_k]$ bei J = MBK(DV(f), DV(g)) (MBK – mažiausias bendras kartotinis) galime apibrėžti labai svarbią funkciją Griobnerio bazių sudarymo teorijoje:

$$\mathrm{S}(f, g) = \frac{J}{\mathrm{DN}(f)} f - \frac{J}{\mathrm{DN}(g)} g\,. \qquad \textbf{(1.8)}$$

Žemiau pateiksime Buchbergerio **1.3. algoritmą**, kuris perskaičiuoja keleto kintamųjų kvadratinių daugianarių lygčių sistemą, užrašytą kaip daugianarių aibę $F = \{f_1, \ldots, f_s\}$ į Griobnerio bazių aibę $G = \{g_1, \ldots, g_t\}$.

1.3. algoritmas. Buchbergerio algoritmas.

$M = \{\{i, j\} \mid 1 \le i < j \le s\}$
$G = F$
$t = s$
Kartoti kol $M \neq \varnothing$
    Pasirinkti $\{i, j\} \in M$
    $M = M - \{\{i, j\}\}$
    Jei Kriterijus1(f_i, f_j) = Ne ir Kriterijus2(f_i, f_j, M) = Ne Tai
        s = S(f_i, f_j)
        r = dalybos s iš G liekana
        Jei $r \neq 0$ Tai
            $t = t + 1$
            $f_t = s$
            $G = G \cup \{f_t\}$
            $M = M \cup \{\{i, t\} \mid 1 \le i \le t\text{-}1\}$
Grąžinti G

Dalybos liekanos radimo **1.2. algoritmas** yra skaičiavimų atžvilgiu sudėtingiausia **1.3. algoritmo** dalis. Siekiant sumažinti dalybų skaičių buvo sugalvoti du kriterijai:

- Kriterijus1(f_i, f_j) grąžina „Taip“, jeigu DV(f_i) ir DV(f_j) neturi bendrų kintamųjų. Priešingu atveju šis kriterijus gražina reikšmę „Ne“.

- Kriterijus2(f_i, f_j, M) grąžina „Taip“, jeigu $\exists l \notin \{i, j\}$ toks, kad $\{i, l\} \notin M$, $\{l, i\} \notin M$, $\{j, l\} \notin M$, $\{l, j\} \notin M$ ir DN(f_l) dalina MBK(DV(f_i), DV(f_j)). Priešingu atveju šis kriterijus gražina reikšmę „Ne“.

Kaip matome, **1.2. algoritmas** ir **1.3. algoritmas** priklauso nuo pasirinktos daugianario narių rikiavimo tvarkos taikant funkcijas DV(f), DK(f) ir DN(f). Pasirinkus skirtingas rikiavimo tvarkas **1.3. algoritmo** pagalba galima gauti skirtingas Griobnerio bazes G.

Blogiausiu atveju yra žinoma, kad 1.3. algoritmo vykdymo trukmė yra dvigubos eksponentės eilės, o bendruoju atveju – eksponentės eilės (7). Geriausią mums žinomą šio algoritmo variantą sukūrė Jean-Charles Faugere (11). Kai lygčių skaičius s yra lygus kintamųjų skaičiui k, šio algoritmo sudėtingumas yra:

- Jeigu laukas, virš kurio yra daugianariai, yra didelis, tai teigiama, kad algoritmo teorinis sudėtingumas yra $O(2^{3k})$ ir praktiškai gaunamas $O(2^{2,7k})$.

- Kai daugianariai yra virš lauko Z_2, tai teigiama, kad algoritmo sudėtingumas yra apie $O(2^{2n})$. Tai yra sudėtingiau negu pilno perrinkimo sudėtingumas $O(2^n)$.

Geriausias žinomas metodas spręsti keleto kintamųjų kvadratinių daugianarių lygčių sistemas, neskaitant pilno visų galimų variantų perrinkimo, yra Griobnerio bazių radimo metodas. Kaip matome, sudaryti Griobnerio bazes yra sudėtingas uždavinys.

Šiame darbe taip pat bus bandoma įvertinti Griobnerio bazių sudarymo laiko funkciją, priklausančią nuo kintamųjų skaičiaus k. Panaudoję aproksimaciją mažiausių kvadratų metodu surasime nežinomus funkcijų parametrus. Plačiau apie tai yra literatūros šaltinyje (20).

2. TIRAIMOJI DALIS IR REZULTATAI

2.1. BURAU VAIZDAVIMO TYRIMAS

Tyrimo tikslas – nustatyti, ar Burau vaizdavimas yra teisingas.

Pirmiausia reikia susitarti, kaip bus vaizduojami daugianariai iš žiedo $Z_p[x_1, x_2, \dots, x_m]$.

1.1.3. skyrelyje buvo minėta, kad kiekvieną daugianario vienanarį galima vienareikšmiškai sunumeruoti ir kompiuterio atmintyje saugoti vieno bito pavidalu. Bitų seką, nusakančią visus vienanarius, galime įsivaizduoti kaip dvejetainį skaičių. Šį skaičių, nusakantį visus daugianario vienanarius, darbe pateiksime šešioliktainiu formatu.

Kompiuterio atmintyje daugianaris saugomas kaip nenulinių koeficientų ir juos atitinkančių vienanarių visuma. Daugianarį galima atvaizduoti kaip koeficientų sandaugų su vienanariais sumą. Daugybos operaciją žymėsime simboliu „*“. Pavyzdžiui, užrašas 1*4A+2*101 reiškia, kad daugianaryje iš vieneto yra padauginti vienanariai 4A ir iš dvejeto padauginti vienanariai 101. Čia vienanariai pateikti šešioliktainiu formatu.

Žinom, kad daugianaris iš žiedo $Z_2[x_1, x_2, \dots, x_m]$ turi vienintelį nenulinį koeficientą. Norint pavaizduoti tokį daugianarį, užtenka atvaizduoti tik vienanarį šešioliktainiu formatu, kuris yra padaugintas iš vieneto.

Tikrinant, ar Burau vaizdavimas yra teisingas, galima patikrinti, ar įvaizdžio lygmenyje yra tenkinamos B_n grupės (1.2) ir (1.3) savybės. Žemiau pateiktoje **2.1. lentelėje** pateikiami tokio tikrinimo rezultatai. Buvo pasirinkta braidų grupė B_4 ir tikrinama, ar programos atsitiktinai generuojamų braidų Burau įvaizdžiams X, A, Y ir $B = XAY$ galioja (1.3) lygybė, o tai pat tikrinama, ar elementariųjų braidų Burau įvaizdžiams galioja (1.2) lygybė. Kiekvienas iš įvaizdžių X, A ir Y buvo sudaromas iš dešimties atsitiktinai parinktų elementariųjų braidų įvaizdžių. Rezultatai pateikti su skirtingais daugianariais t bei skirtingomis žiedo $Z_p[x_1, x_2, \dots, x_m]$ parametrų p ir m reikšmėmis.

2.1. lentelė

Burau vaizdavimo tikrinimas

p	m	Daugianaris t	Tenkinamos B_n savybės
2	1	3	Taip
2	2	D	Taip
2	3	A5	Taip
2	4	3FC2	Taip
2	5	C4AD423C	Taip
2	6	3E3E5BD79628728D	Taip
3	1	1*4+2*3	Taip
3	2	1*4A+2*101	Taip
3	3	1*2088C0+2*31C0437	Taip
3	4	1*1567320D06C00C41CB032 +2*A9840E0502013AA04A84	Taip
3	5	1*224036E4C340110CA4322EB1 1841000BC8C4003632E1E0028170402649D13+2*1CAB48113898 A622524CC1442218DC44301BD600411004D4100E9CC8B00E4	Taip
5	1	1*6+2*8+4*1	Taip
5	2	1*130000+2*88488+3*C03910+4*200261	Taip
5	3	1*4001611000C8609000040420886205 8+2*100A08000012080082 8020101410A02+3*10B3090680600820020205008020 5400+4*A48 004060020044B50488B006088025	Taip

Iš **2.1. lentelės** matome, kad visais tikrinimo atvejais (1.2) ir (1.3) savybės buvo tenkinamos. Šios sakybės buvo tenkinamos ir visais kitais stebėtais atvejais, kurie nebuvo įtraukti į lentelę. Daugianario t = 1*2088C0+2*31C0437 analizės pavyzdys pateiktas **1 Priede**.

2.2. DAUGIANARIŲ TYRIMAS

Tyrimo tikslas – nustatyti kriterijus, kaip parinkti daugianarį t.

Pasirinkus konkretų daugianarį $t \in Z_p[x_1, x_2, \dots, x_m]$ yra gaunamas Burau vaizdavimas virš tam tikro požiedžio $T \subset Z_p[x_1, x_2, \dots, x_m]$ (žiūrėti 1.1.2. skyrelį). 1.2.2. skyrelyje buvo parodyta, kad atsižvelgiant į požiedį T vienkryptę funkciją nusakančioje keleto kintamųjų kvadratinių daugianarių lygčių sistemoje gali būti mažiau kintamųjų negu nurodyta (1.6) formulėje. Taip yra todėl, kad požiedyje T gali nebūti visų vienanarių, kurie galimi žiede $Z_p[x_1, x_2, \dots, x_m]$. Kadangi daugianaris t

priklauso požiedžiui *T*, tai norint užtikrinti, kad požiedyje *T* būtų visi įmanomi vienanariai, reikia, kad daugianaryje *t* visi vienanariai būtų padauginti iš nenulinių koeficientų. Akivaizdu, kad žiede $Z_p[x_1, x_2, \dots, x_m]$ iš viso yra p^m vienanarių, daugianarių – p^{p^m}, o daugianarių be nulinių koeficientų – $(p-1)^{p^m}$. Žemiau pateikta **2.2. lentelė** su tikslesnėmis šių daugianarių kiekio reikšmėmis.

2.2. lentelė

Žiedo $Z_p[x_1, \dots, x_m]$ elementai

p	m	Visų daugianarių skaičius žiede	Daugianarių be nulinių koeficientų skaičius žiede
2	1	$2^2 = 4$	$1^2 = 1$
2	2	$2^4 = 16$	$1^4 = 1$
2	3	$2^8 = 256$	$1^8 = 1$
2	4	$2^{16} \approx 6{,}554\times10^4$	$1^{16} = 1$
2	5	$2^{32} \approx 4{,}295\times10^9$	$1^{32} = 1$
2	6	$2^{64} \approx 1{,}845\times10^{19}$	$1^{64} = 1$
3	1	$3^3 = 27$	$2^3 = 27$
3	2	$3^9 \approx 1{,}968\times10^4$	$2^9 = 512$
3	3	$3^{27} \approx 7{,}626\times10^{12}$	$2^{27} \approx 1{,}342\times10^8$
3	4	$3^{81} \approx 4{,}434\times10^{38}$	$2^{81} \approx 2{,}418\times10^{24}$
3	5	$3^{243} \approx 8{,}719\times10^{115}$	$2^{243} \approx 1{,}413\times10^{73}$
5	1	$5^5 = 3125$	$4^5 = 1024$
5	2	$5^{25} \approx 2{,}98\times10^{17}$	$4^{25} \approx 1{,}126\times10^{15}$
5	3	$5^{125} \approx 2{,}351\times10^{87}$	$4^{125} \approx 1{,}809\times10^{75}$

Kai *p* = 2, **2.2. lentelėje** matome, kad visame žiede yra tik vienas elementas, su kuriuo tikrai būtų tenkinama (1.6) formulė. Visais kitais atvejais daugianarį *t* galima rinktis iš platesnės aibės.

1.1. algoritmo pagalba galime įvertinti, kiek elementų yra požiedyje *T*. Požiedžio elementų skaičius su skirtingomis parametro *p* reikšmėmis pateiktas **2.3. lentelėje**.

2.3. lentelė

Požiedžio T $\subset Z_p[x_1, \dots, x_m]$ maksimalus elementų skaičius

p	Elementų skaičius
2	4
3	18
5	1250

2.3. lentelėje matome, kad požiedžio *T* elementų skaičius nepriklauso nuo parametro *m*. Šiuos požiedžius sudarantys elementai pateikti **2 Priede**.

Pateiksime kelis Burau vaizdavimo pavyzdžius su skirtingomis daugianario *t* reikšmėmis, kai šis daugianaris neturi nulinių koeficientų. Bus atliekamas B_{10} grupės braidų, sudarytų iš 100 atsitiktinai parinktų elementariųjų braidų, Burau vaizdavimas į matricą *X*.

1. Pavyzdys: *p* = 2, *m* = 3, *n* = 10, *t* = FF.

Su šiuo daugianarių požiedis *T* įgyja 4 skirtingas reikšmes, o matrica *X* atrodo taip:

```
FE 0 FF 0 0 0 0 0 0 0
FE 0 0 0 0 FF 0 0 0 0
1  0 0 0 0 0  0 0 0 0
FE FF 0 0 0 0 0 0 0 0
FE 0 0 FF 0 0 0 0 0 0
FE 0 0 0 0 0 FF 0 0 0
FE 0 0 0 0 0 0 0 0 FF
FE 0 0 0 0 0 0 FF 0 0
FE 0 0 0 0 0 0 0 FF 0
FE 0 0 0 FF 0 0 0 0 0
```

2. Pavyzdys: *p* = 3, *m* = 2, *n* = 10, *t* = 1*1FF.

Su šiuo daugianarių požiedis *T* įgyja 6 skirtingas reikšmes, o matrica *X* atrodo taip:

```
2*1FE 1*1FF 1*0 1*0 1*0 1*0 1*0 1*0 1*0 1*0
2*1FE 1*0 1*1FF 1*0 1*0 1*0 1*0 1*0 1*0 1*0
2*1FE 1*0 1*0 1*0 1*1FF 1*0 1*0 1*0 1*0 1*0
1*1   1*0 1*0 1*0 1*0   1*0 1*0 1*0 1*0 1*0
2*1FE 1*0 1*0 1*0 1*0 1*0 1*0 1*0 1*0 1*1FF
2*1FE 1*0 1*0 1*0 1*0 1*1FF 1*0 1*0 1*0 1*0
2*1FE 1*0 1*0 1*1FF 1*0 1*0 1*0 1*0 1*0 1*0
2*1FE 1*0 1*0 1*0 1*0 1*0 1*0 1*0 1*1FF 1*0
2*1FE 1*0 1*0 1*0 1*0 1*0 1*1FF 1*0 1*0 1*0
2*1FE 1*0 1*0 1*0 1*0 1*0 1*0 1*1FF 1*0 1*0
```

3. Pavyzdys: $p = 3$, $m = 2$, $n = 10$, t = 1*1F0+2*F.

Su šiuo daugianarių požiedis T įgyja 18 skirtingų reikšmių, o matrica X atrodo taip:

```
1*1 1*0 1*188+2*77 1*188+2*77 1*77+2*188 1*188+2*77 1*77+2*188 1*77+2*188 1*0 1*0
1*E+2*1F1 1*0 1*0 1*77+2*188 1*0 1*77+2*188 1*77+2*188 1*7+2*180 1*0 1*77+2*188
1*E+2*1F1 1*77+2*188 1*1F0+2*F 1*188+2*77 1*77+2*188 1*77+2*188 1*77+2*188 1*188+2*77 1*0 1*77+2*188
1*71+2*8 1*0 1*188+2*77 1*77+2*188 1*8+2*70 1*188+2*77 1*77+2*188 1*0 1*0 1*0
1*71+2*8 1*77+2*188 1*188+2*77 1*77+2*188 1*188+2*77 1*0 1*1F0+2*F 1*0 1*0 1*77+2*188
1*180+2*6 1*0 1*0 1*77+2*188 1*77+2*188 1*0 1*188+2*77 1*0 1*0 1*8+2*70
1*E+2*1F1 1*0 1*0 1*0 1*188+2*77 1*7+2*180 1*0 1*77+2*188 1*0 1*77+2*188
1*180+2*6 1*8+2*70 1*0 1*0 1*0 1*0 1*0 1*0 1*188+2*77 1*188+2*77
1*E+2*1F1 1*0 1*0 1*1F0+2*F 1*77+2*188 1*0 1*188+2*77 1*77+2*188 1*77+2*188 1*77+2*188
1*71+2*8 1*0 1*0 1*188+2*77 1*188+2*77 1*188+2*77 1*77+2*188 1*188+2*77 1*7+2*180 1*188+2*77
```

Pirmame pavyzdyje gautoje matricoje galima įžvelgti aiškią jos struktūrą: pirmajame stulpelyje yra daugianariai „FE“ ir vienas daugianaris „1“, likusi matricos dalis yra užpildyta daugianariais „0“ bei ties matricos įstrižaine išsibarsčiusiais daugianariais „FF“. Kai $p = 2$, analogiška struktūra buvo randama visose matricose, gautose Burau vaizdavimo pagalba, nepriklausomai nuo parametrų m ir n.

Antrame pavyzdyje gautoje matricoje taip pat galima įžvelgti aiškią struktūrą, analogišką pirmo pavyzdžio matricai. Pastebėsime, kad šiuo atveju požiedžis T įgyja tik 6 reikšmes iš 18 maksimaliai galimų. Tačiau trečiame pavyzdyje, kai požiedis įgyja visas galimas 18 reikšmių, jau nebepavyksta įžvelgti aiškios matricos struktūros.

2.3. KELETO KINTAMŲJŲ KVADRATINIŲ DAUGIANARIŲ LYGČIŲ SISTEMOS

Tyrimo tikslas – nustatyti vienkryptės funkcijos parametrų p, n ir m reikšmes.

Nustatant parametrų p, n ir m reikšmes sieksime, kad vienkryptę funkciją nusakančios keleto kintamųjų kvadratinių daugianarių lygčių sistemos (žiūrėti 1.2.2. skyrelį) kintamųjų skaičius būtų nemažesnis už 1600. Taip pat bus siekiama užtikrinti, kad Burau vaizdavimo pagalba gaunamų matricų užimama kompiuterio atmintis būtų kuo mažesnė.

Atsižvelgus į (1.6) formulę galime užrašyti lygybę $1600 = 2 \cdot p^m \cdot n^2$. Iš šio s lygybės išreiškus parametrą n, gauname formulę:

$$n = \left\lceil \sqrt{\frac{1600}{2p^m}} \right\rceil . \qquad \textbf{(2.1)}$$

Turėdami (2.1) formulę ir pasirinkę parametrus *p* ir *m*, galėsime pasakyti, kokia turi būti minimali parametro *n* reikšmė, kad gautume bent 1600 kintamųjų lygčių sistemą. Gavę parametro *n* reikšmę (1.6) formulės pagalba rasime tikslų lygčių sistemos kintamųjų skaičių *k*, o (1.4) formulės pagalba galėsime pasakyti, kiek kompiuterio atminties užims Burau vaizdavimo pagalba gautos matricos. Šie rezultatai su skirtingomis parametrų *p* ir *m* reikšmėmis pateikti **2.4. lentelėje** ir **2.5.lentelėje**.

2.4. lentelė

Burau vaizdavimo matricų virš $Z_3[x_1, \dots, x_m]$ užimama kompiuterio atmintis

m	*n*	*k*	Matricų užimama atmintis (baitai / B)
1	17	1734	578
2	10	1800	400
3	6	1944	288
4	4	2592	352
5	2	1944	248
6	2	5832	736

2.5. lentelė

Burau vaizdavimo matricų virš $Z_5[x_1, \dots, x_m]$ užimama kompiuterio atmintis

m	*n*	*k*	Matricų užimama atmintis (baitai / B)
1	13	1690	676
2	6	1800	576
3	3	2250	576
4	2	5000	1264
5	1	6250	1564
6	1	31250	7816

2.4. lentelėje ir **2.5 lentelėje** iš karto atmetame tuos rezultatus, kur n = 1 ir n = 2. Taip daroma todėl, kad braidų grupė B_1 neegzistuoja pagal apibrėžimą, o braidų grupėje B_2 yra tik vienas elementarusis braidas ir generuojamas žodis priklauso tik nuo savo paties ilgio. Iš likusių rezultatų išsirenkame tą, kur naudojama mažiausiai kompiuterio atminties.

Su sąlyga $n > 2$ mažiausiai kompiuterio atminties (288 baitus) naudojama, kai vienkryptės funkcijos parametrai yra: $p = 3$, $m = 3$ ir $n = 6$.

2.4. GRIOBNERIO BAZIŲ TYRIMAS

Tyrimo tikslas – suprastinti gaunamas keleto kintamųjų kvadratinių daugianarių lygčių sistemas Griobnerio bazių pagalba ir įvertinti šių skaičiavimų trukmę. Griobnerio bazės yra ieškomos **1.3. algoritmo** pagalba.

Pakankamai paprastų, su nedideliu kintamųjų kiekiu, lygčių sistemų pavyko gauti tik tokiu atveju, kai kintamieji yra iš lauko Z_2. Skaičiavimai buvo atliekami su dviem skirtingais kompiuteriais:

- Komp1: dviejų branduolių 2,13 GHz Intel procesorius, 4 GB operatyviosios atminties, Windows Vista operacinė sistema.
- Komp2: dviejų branduolių 1,81 GHz AMD procesorius, 1,4 GB operatyviosios atminties, Windows XP operacinė sistema.

Aptarkime žymėjimus pateiktose Griobnerio bazių skaičiavimo **2.6. lentelėje**, **2.7. lentelėje** ir **2.8. lentelėje**. Parametras *t* reiškia daugianarį, kurio pagal buvo sudaromos lygčių sistemos. Parametras *m* nusako daugianario *t* kintamųjų skaičių. Parametras *s* nusako lygčių skaičių gautoje lygčių sistemoje. Parametras *k* nusako kintamųjų skaičių gautuose lygčių sistemose. Skaičiavimų trukmės *L* reikšmės yra vidutinės atlikus 10 bandymų ir pateikiamos milisekundėmis. Gautos lygčių sistemos ir jų Griobnerio bazės pateiktos **3 Priede**.

2.6. lentelė

Griobnerio bazių skaičiavimas, kai tvarka leksikografinė

t	*m*	*k*	*s*	Skaičiavimų trukmė *L* su Komp1 (ms)	Skaičiavimų trukmė su *L* Komp2 (ms)
1	1	8	4	12	15
3	1	14	8	2232	1093
2	1	16	8	1480	5000
A	2	22	12	–	–

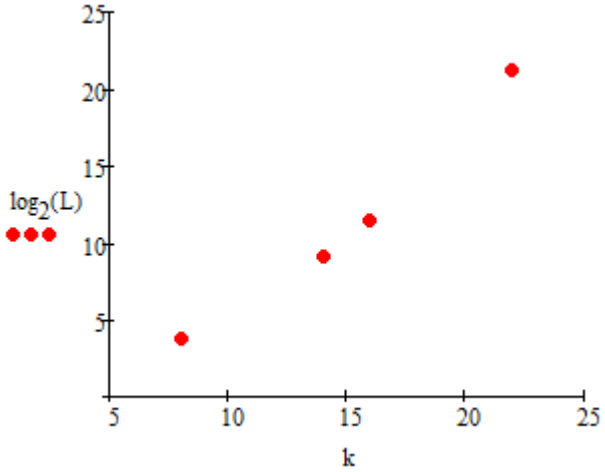
Iš **2.6. lentelėje** gautų rezultatų matome, kad lygčių sistemoms nepavyko sudaryti Griobnerio bazių, kai t = A, nes skaičiavimai truko ilgiau nei 24 valandas.

2.7. lentelė

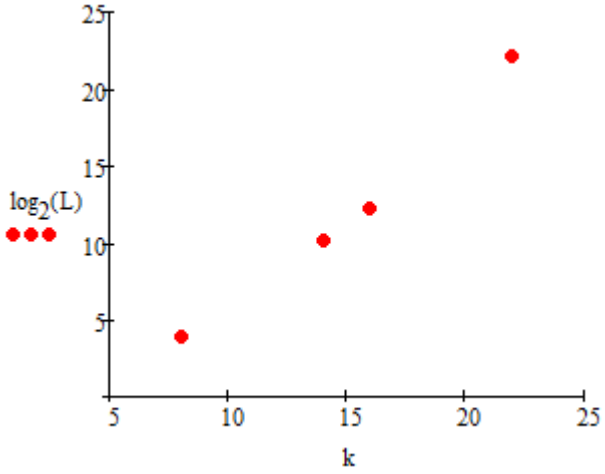
Griobnerio bazių skaičiavimas, kai tvarka laipsninė ir leksikografinė

t	*m*	*k*	*s*	Skaičiavimų trukmė *L* su Komp1 (ms)	Skaičiavimų trukmė su *L* Komp2 (ms)
1	1	8	4	14	15
3	1	14	8	561	1093
2	1	16	8	2706	5000
A	2	22	12	2397624	4528984

Iš **2.7. lentelėje** gautų rezultatų akivaizdžiai matome, kad skaičiavimų trukmės *L* ir kintamųjų skaičius *k* nėra susiję pagal tiesinį dėsnį. Pabandome šiuos dydžius atvaizduoti pusiau logaritminėje skalėje, kai naudojamas dvejetainis logaritmas.



2.1. pav. Skaičiavimų trukmė su Komp1 pusiau logaritminėje skalėje.



2.2. pav. Skaičiavimų trukmė su Komp2 pusiau logaritminėje skalėje.

Iš **2.1. paveikslo** ir **2.2. paveikslo** matome, kad abejais atvejais skaičiavimų trukmė *L* pusiau logaritminėje skalėje išsidėsto tiesės pavidalu. Vadinasi, duomenis nusako $L = v \cdot 2^{u \cdot k}$ funkcinė priklausomybė, kur *u* ir *v* yra nežinomi parametrai. Šiuos parametrus galima rasti mažiausių kvadratų metodo pagalba.

Duomenims, gautiems su kompiuteriu Komp1, suradus parametrų *u* ir *v* reikšmes galima užrašyti tokią *L* funkcinę priklausomybę nuo *k*:

$$L_1(k) = 3{,}934 \cdot 10^{-5} \cdot 2^{1{,}628 \cdot k} \quad \textbf{(2.2)}$$

Duomenims, gautiems su kompiuteriu Komp2, suradus parametrų *u* ir *v* reikšmes galima užrašyti tokią *L* funkcinę priklausomybę nuo *k*:

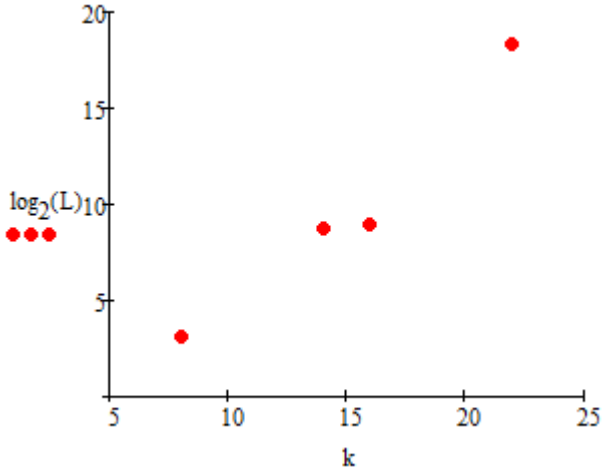
$$L_2(k) = 6{,}891 \cdot 10^{-5} \cdot 2^{1{,}633 \cdot k} \quad \textbf{(2.3)}$$

2.8. lentelė

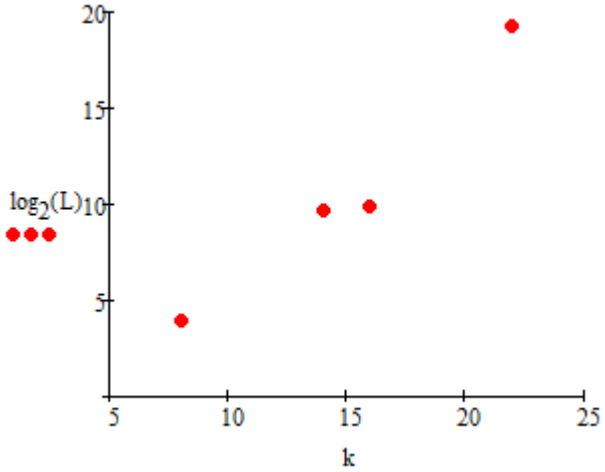
Griobnerio bazių skaičiavimas, kai tvarka laipsninė ir atvirkštinė leksikografinė

t	*m*	*k*	*s*	Skaičiavimų trukmė *L* su Komp1 (ms)	Skaičiavimų trukmė su *L* Komp2 (ms)
1	1	8	4	9	15
3	1	14	8	431	796
2	1	16	8	481	921
A	2	22	12	334317	615296

Iš **2.8. lentelėje** gautų rezultatų akivaizdžiai matome, kad skaičiavimų trukmės *L* ir kintamųjų skaičius *k* nėra susiję pagal tiesinį dėsnį. Pabandome šiuos dydžius atvaizduoti pusiau logaritminėje skalėje, kai naudojamas dvejetainis logaritmas.



2.3. pav. Skaičiavimų trukmė su Komp1 pusiau logaritminėje skalėje.



2.4. pav. Skaičiavimų trukmė su Komp2 pusiau logaritminėje skalėje.

Iš **2.3. paveikslo** ir **2.4. paveikslo** matome, kad abejais atvejais skaičiavimų trukmė L pusiau logaritminėje skalėje išsidėsto tiesės pavidalu. Vadinasi, duomenis nusako $L = v \cdot 2^{u \cdot k}$ funkcinė priklausomybė, kur u ir v yra nežinomi parametrai. Šiuos parametrus galima rasti mažiausių kvadratų metodo pagalba.

Duomenims, gautiems su kompiuteriu Komp1, suradus parametrų u ir v reikšmes galima užrašyti tokią L funkcinę priklausomybę nuo k:

$$L_3(k) = 1{,}92 \cdot 10^{-5} \cdot 2^{1{,}546 \cdot k} \qquad \textbf{(2.4)}$$

Duomenims, gautiems su kompiuteriu Komp2, suradus parametrų u ir v reikšmes galima užrašyti tokią L funkcinę priklausomybę nuo k:

$$L_4(k) = 4{,}035 \cdot 10^{-5} \cdot 2^{1{,}538 \cdot k} \qquad \textbf{(2.5)}$$

Pateiksime kelis vienkryptė funkciją nusakančių lygčių pavyzdžius. Lygčių sistemoje simbolis „*“ žymi daugybos operaciją, o simbolis „^“ – kėlimą laipsnių. Kai naudojamas kėlimo laipsnių simbolis „^“, tai šio simbolio kairėje pusėje esantis skaičius nusako kintamojo indeksą, o dešinėje – laipsnio rodiklį. Pavyzdžiui, užrašas „4^2“ reiškia – x_4^2.

4. Pavyzdys: $p = 2$, $m = 1$, $n = 2$, $t = 1$. Naudojama laipsninė ir atvirkštinė leksikografinė rikiavimo tvarka.

Gauname $s = 4$ lygčių sistemą su $k = 8$ kintamaisiais:

```
1*1^1*5^1+1*2^1*7^1+1 = 0
1*1^1*6^1+1*2^1*8^1 = 0
1*3^1*5^1+1*4^1*7^1 = 0
1*3^1*6^1+1*4^1*8^1+1 = 0
```

Šią lygčių sistemą suprastinę Griobnerio bazių pagalba gauname lygčių sistemą su 8 kintamaisiais ir 15 lygčių:

```
1*1^1*5^1+1*2^1*7^1+1 = 0
1*1^1*6^1+1*2^1*8^1 = 0
1*3^1*5^1+1*4^1*7^1 = 0
1*3^1*6^1+1*4^1*8^1+1 = 0
1*2^1*7^1+1*4^1*8^1+1 = 0
1*4^1*5^1*8^1+1*4^1*6^1*8^1+1*6^1*7^1+1*5^1*8^1+1*6^1 = 0
1*1^1*4^1*8^1+1*3^1*4^1*8^1+1*2^1*3^1+1*1^1*4^1+1*3^1 = 0
1*2^1*8^1+1*2^1+1*8^1+1 = 0
1*3^1*4^1*8^1+1*1^1*7^1*8^1+1*2^1*3^1+1*1^1*7^1+1*1^1*8^1+1*1^1+1*3^1 = 0
1*2^1*4^1+1*2^1+1*4^1+1 = 0
1*4^1*7^1+1*4^1+1*7^1+1 = 0
1*7^1*8^1+1*7^1+1*8^1+1 = 0
1*4^1*6^1*8^1+1*6^1*7^1+1*6^1 = 0
1*1^1*2^1*3^1+1*1^1*3^1*4^1+1*1^1*3^1 = 0
1*5^1*6^1*7^1+1*5^1*6^1*8^1+1*5^1*6^1 = 0
```

Tikrasis lygčių sistemos sprendinys tenkina suprastintą lygčių sistemą.

Pavyzdys: $p = 2$, $m = 2$, $n = 2$, $t = \mathrm{D}$. Naudojama laipsninė ir atvirkštinė leksikografinė rikiavimo tvarka.

Gauname $s = 12$ lygčių sistemą su $k = 24$ kintamaisiais:

```
    1*1^1*13^1+1*4^1*19^1+1 = 0
    1*2^1*13^1+1*4^1*13^1+1*5^1*13^1+1*1^1*14^1+1*2^1*14^1+1*4^1*14^1+1*5^1*14^1+
+1*4^1*19^1 = 0
    1*3^1*13^1+1*4^1*13^1+1*5^1*13^1+1*3^1*14^1+1*4^1*14^1+1*5^1*14^1+1*1^1*15^1+
+1*2^1*15^1+1*3^1*15^1+1*4^1*19^1+1*5^1*19^1+1*6^1*19^1+1*4^1*20^1+1*5^1*20^1+
+1*6^1*20^1+1*4^1*21^1+1*5^1*21^1+1*6^1*21^1 = 0
    1*1^1*16^1+1*4^1*22^1 = 0
    1*2^1*16^1+1*4^1*16^1+1*5^1*16^1+1*1^1*17^1+1*2^1*17^1+1*4^1*17^1+1*5^1*17^1+
+1*4^1*22^1 = 0
    1*3^1*16^1+1*4^1*16^1+1*5^1*16^1+1*3^1*17^1+1*4^1*17^1+1*5^1*17^1+1*1^1*18^1+
+1*2^1*18^1+1*3^1*18^1+1*4^1*22^1+1*5^1*22^1+1*6^1*22^1+1*4^1*23^1+1*5^1*23^1+
+1*6^1*23^1+1*4^1*24^1+1*5^1*24^1+1*6^1*24^1 = 0
    1*7^1*13^1+1*10^1*19^1 = 0
    1*8^1*13^1+1*10^1*13^1+1*11^1*13^1+1*7^1*14^1+1*8^1*14^1+1*10^1*14^1+
+1*11^1*14^1+1*10^1*19^1+1 = 0
    1*9^1*13^1+1*10^1*13^1+1*11^1*13^1+1*9^1*14^1+1*10^1*14^1+1*11^1*14^1+
+1*7^1*15^1+1*8^1*15^1+1*9^1*15^1+1*10^1*19^1+1*11^1*19^1+1*12^1*19^1+1*10^1*20^1+
+1*11^1*20^1+1*12^1*20^1+1*10^1*21^1+1*11^1*21^1+1*12^1*21^1+1 = 0
    1*7^1*16^1+1*10^1*22^1+1 = 0
    1*8^1*16^1+1*10^1*16^1+1*11^1*16^1+1*7^1*17^1+1*8^1*17^1+1*10^1*17^1+
+1*11^1*17^1+1*10^1*22^1+1 = 0
    1*9^1*16^1+1*10^1*16^1+1*11^1*16^1+1*9^1*17^1+1*10^1*17^1+1*11^1*17^1+
+1*7^1*18^1+1*8^1*18^1+1*9^1*18^1+1*10^1*22^1+1*11^1*22^1+1*12^1*22^1+1*10^1*23^1+
+1*11^1*23^1+1*12^1*23^1+1*10^1*24^1+1*11^1*24^1+1*12^1*24^1+1 = 0
```

Lygčių sistemos nepavyko suprastinti Griobnerio bazių pagalba, nes skaičiavimai trūko ilgiau nei 24 valandas.

3. PROGRAMINĖ REALIZACIJA IR INSTRUKCIJOS VARTOTOJUI

3.1. PROGRAMINĖS ĮRANGOS PASIRINKIMAS

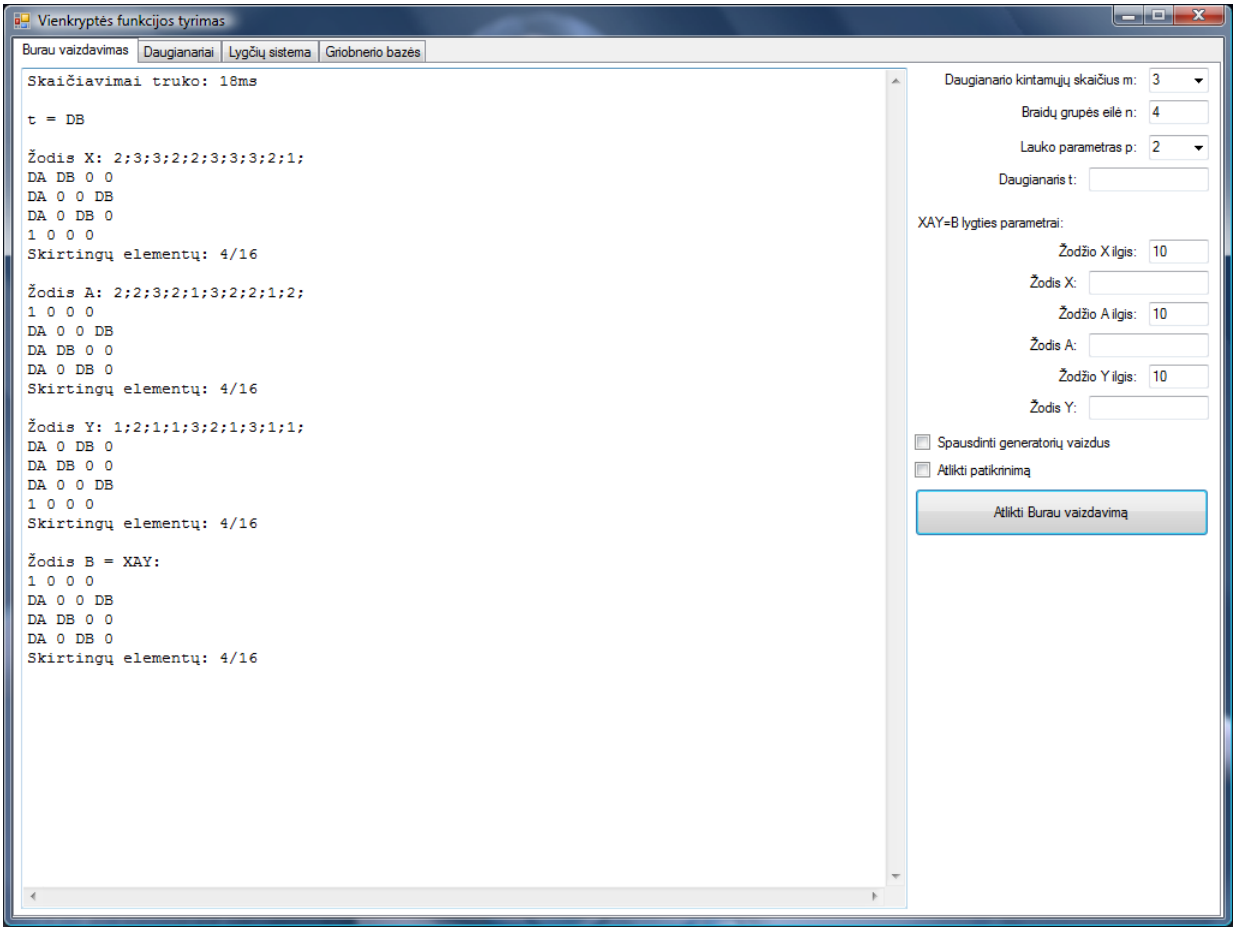
Šiame darbe aprašytiems skaičiavimas atlikti buvo sudaryta programa C# programavimo kalba. Ši programavimo kalba pasirinkta dėl plačių galimybių susikurti bet kokias duomenų struktūras ir lengvai įgyvendinamų bitų apdorojimo operacijų. Gebant paprastai atlikti bitų operacijas galima atlikti greitesnius skaičiavimus, o duomenų struktūros kompiuterio atmintyje užima mažiau vietos.

Norint naudotis programa reikia tūrėti Windows operacinę sistemą su .NET Framework v3.5 paketu.

Sudarytos programos grafinė vartotojo sąsaja susideda iš penkių pagrindinių dalių:

- Burau vaizdavimas,
- Daugianariai,
- Lygčių sistema,
- Griobnerio bazės.

3.2. DALIES „BURAU VAIZDAVIMAS“ APRAŠYMAS



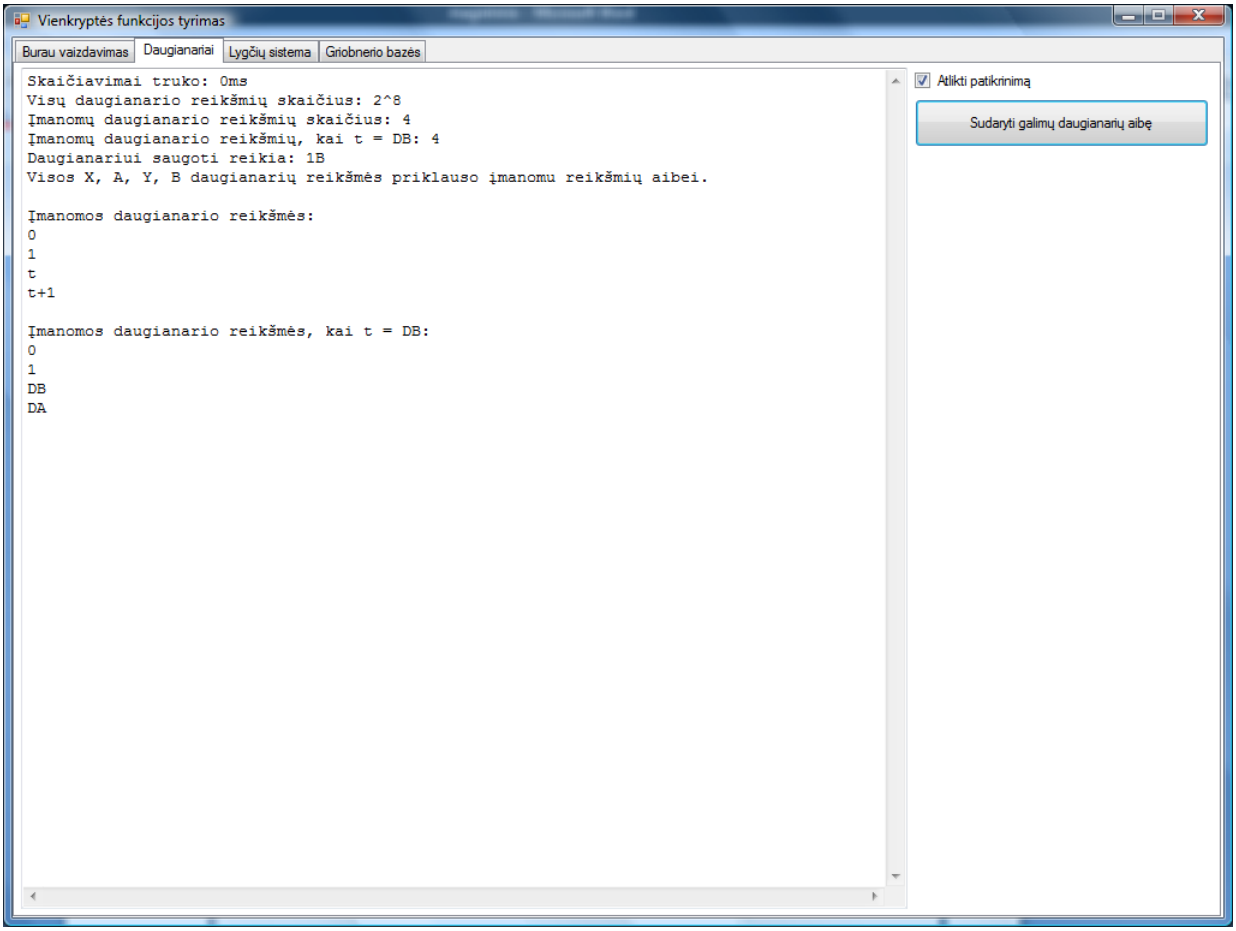
3.1. pav. Programos dalies „Burau vaizdavimas“ vaizdas.

Šioje programos dalyje yra pasirenkami vienkryptės funkcijos (1.5) parametrai ir atliekamas sudaromų braidų žodžių Burau vaizdavimas. Programos dalyje pasirenkami duomenys:

- Daugianario kintamųjų skaičius *m*: galimos reikšmės nuo 0 iki 6.
- Braidų grupės eilė *n*: parametrui taikomas apribojimas $n \geq 2$.
- Lauko parametras *p*: lauko Z_p parametras gali įgyti reikšmes 2, 3, 5 ir 7.
- Daugianaris *t*: pasirenkamas Burau atvaizdavimo daugianaris pagal formatą aptartą 2.1. skyrelyje. Jeigu įvedimo laukas paliekamas tuščias, tai parenkamas atsitiktinis daugianaris.
- Žodžio *X* ilgis: nurodoma, iš kelių atsitiktinių elementariųjų braidų turi būti sugeneruotas žodžio *X* pirmavaizdis.
- Žodis *X*: pasirenkama, iš kokių elementarių braidų turi būti sugeneruotas žodžio *X* pirmavaizdis. Elementarieji braidai pasirenkami nurodant jų indeksus, atskirtus kabliataškiais. Jeigu įvedimo laukas paliekamas tuščias, tai parenkami atsitiktiniai elementarieji braidą atsižvelgiant į lauką „Žodžio *X* ilgis“.
- Žodžio *A* ilgis: nurodoma, iš kelių atsitiktinių elementariųjų braidų turi būti sugeneruotas žodžio *A* pirmavaizdis.
- Žodis *A*: pasirenkama, iš kokių elementarių braidų turi būti sugeneruotas žodžio *A* pirmavaizdis. Elementarieji braidai pasirenkami nurodant jų indeksus atskirtus kabliataškiais. Jeigu įvedimo laukas paliekamas tuščias, tai parenkami atsitiktiniai elementarieji braidą atsižvelgiant į lauką „Žodžio *A* ilgis“.
- Žodžio *B* ilgis: nurodoma, iš kelių atsitiktinių elementariųjų braidų turi būti sugeneruotas žodžio *B* pirmavaizdis.
- Žodis *B*: pasirenkama, iš kokių elementarių braidų turi būti sugeneruotas žodžio *B* pirmavaizdis. Elementarieji braidai pasirenkami nurodant jų indeksus atskirtus kabliataškiais. Jeigu įvedimo laukas paliekamas tuščias, tai parenkami atsitiktiniai elementarieji braidą atsižvelgiant į lauką „Žodžio *B* ilgis“.

Paspaudus mygtuką „Atlikti Burau vaizdavimą“ programa rezultatų lange išveda skaičiavimų trukmę, daugianarį *t* bei matricas *X*, *A*, *Y* ir $B = XAY$. Po kiekvienos matricos pateikiama, kiek skirtingų daugianarių ją sudaro ir kiek joje yra elementų. Prie *X*, *A* ir *Y* matricų išvedami jų pirmavaizdžius sudarančių elementariųjų braidų indeksai. Papildomai pasirinkus „Spausdinti generatorių vaizdus“ yra atspausdinami elementariųjų braidų vaizdai. Papildomai pasirinkus „Atlikti patikrinimą“ yra patikrinama, ar matricos *X*, *A*, *Y*, *B* ir elementariųjų braidų vaizdai tenkina (1.2) ir (1.3) sąlygas.

3.3. DALIES „DAUGIANARIAI“ APRAŠYMAS

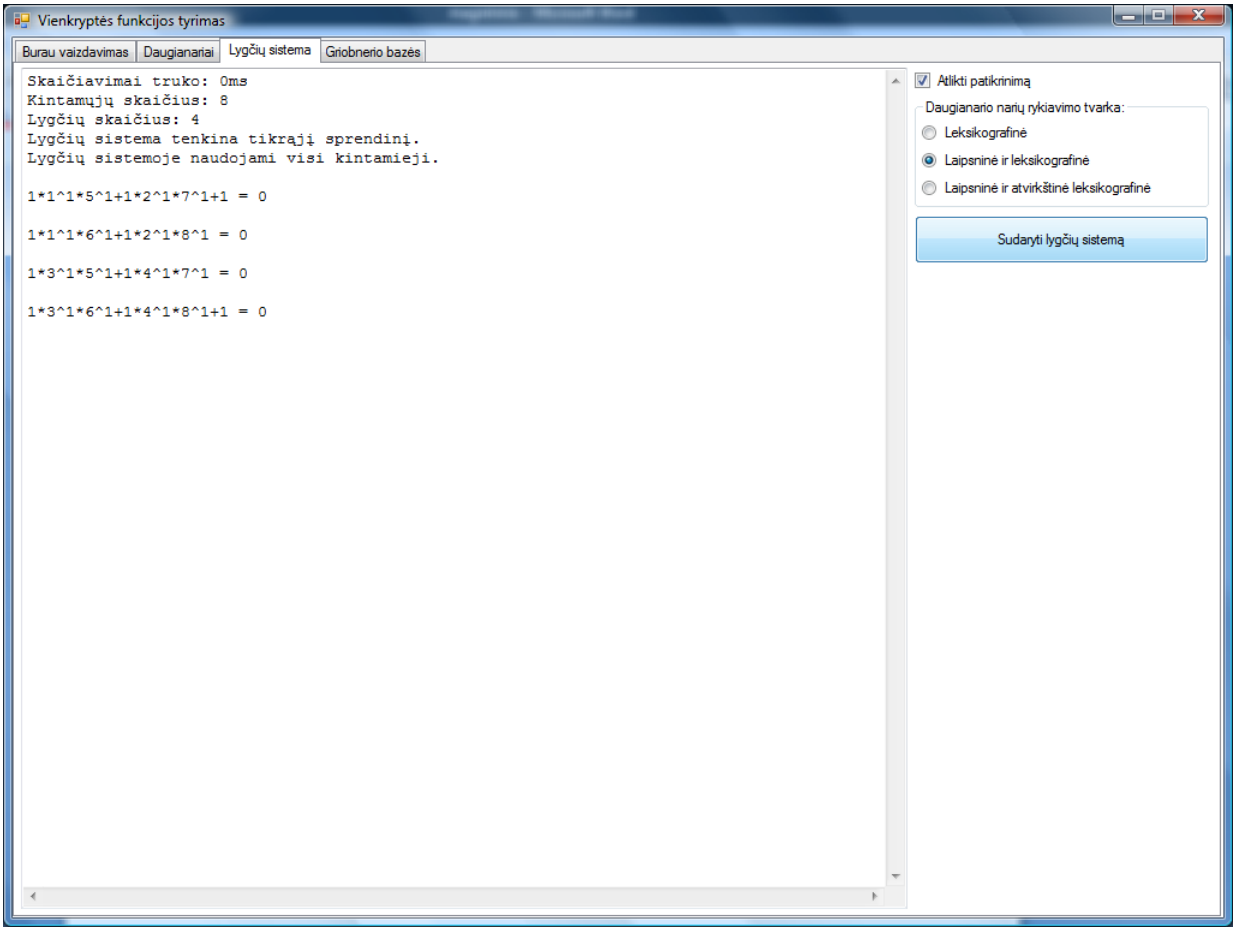


3.2. pav. Programos dalies „Daugianariai“ vaizdas.

Šioje programos dalyje yra nustatoma virš kokio daugianarių požiedžio T yra atliekamas Burau vaizdavimas. Čia skaičiavimus atlikti galima pirma atlikus skaičiavimus programos dalyje „Burau vaizdavimas“.

Paspaudus „Sudaryti galimų daugianarių aibę“ yra naudojami parametrai, kurie yra įvesti ir suskaičiuoti dalyje „Burau vaizdavimas“. Rezultatų lange yra pateikiama skaičiavimų trukmė, visų daugianarių skaičius žiede $Z_p[x_1, x_2, \dots, x_m]$, įmanomų daugianarių skaičių požiedyje T, požiedžio T elementų skaičius su konkrečia t reikšme ir daugianarių užimama kompiuterio atmintis. Taip pat visi galimi požiedžio T elementai bei elementai, gaunami įstačius t reikšmę. Papildomai pasirinkus „Atlikti patikrinimą“ patikrinama, ar visi matricų X, A, Y ir B elementai priklauso požiedžiui T.

3.4. DALIES „LYGČIŲ SISTEMA“ APRAŠYMAS

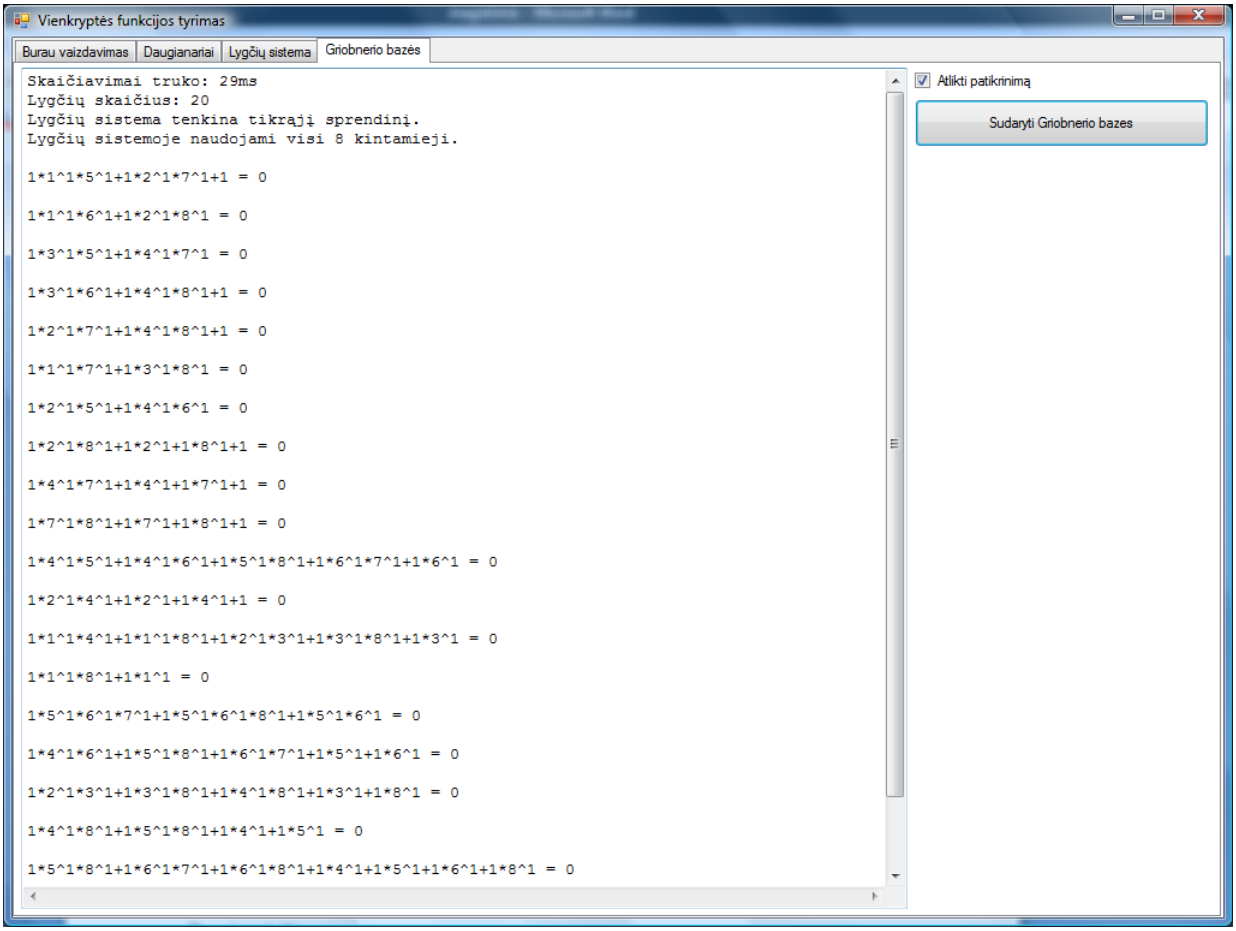


3.3. pav. Programos dalies „Lygčių sistema“ vaizdas.

Šioje programos dalyje yra sudaromos keleto kintamųjų kvadratinių daugianarių lygčių sistemos aprašytos 1.2.2. skyrelyje. Čia skaičiavimus atlikti galima pirma atlikus skaičiavimus programos dalyje „Daugianariai“.

Pasirinkus vieną iš galimų trijų daugianarių narių rikiavimo tvarkų ir paspaudus mygtuką „Sudaryti lygčių sistemą“ rezultatų lange pateikiama skaičiavimų trukmė, lygčių sistemos kintamųjų skaičius, lygčių skaičius lygčių sistemoje ir lygčių sistema 2.4. skyrelyje aprašytu formatu. Papildomai pasirinkus „Atlikti patikrinimą“ patikrinama, ar lygčių sistema tenkina tikrąjį sprendinį ir ar visi kintamieji yra naudojami lygčių sistemoje.

3.5. DALIES „GRIOBNERIO BAZĖS“ APRAŠYMAS



3.4. pav. Programos dalies „Griobnerio bazės“ vaizdas.

Šioje programos dalyje bandoma suprastinti keleto kintamųjų kvadratinės daugianarių lygčių sistemas taikant **1.3. algoritmą**. Čia skaičiavimus atlikti galima pirma atlikus skaičiavimus programos dalyje „Lygčių sistema“.

Paspaudus mygtuką „Sudaryti Griobnerio bazes“ rezultatų lange pateikiama skaičiavimų trukmė, suprastintos lygčių sistemos kintamųjų skaičius ir suprastinta lygčių sistema 2.4. skyrelyje aprašytu formatu. Papildomai pasirinkus „Atlikti patikrinimą“ patikrinama, ar lygčių sistema tenkina tikrąjį sprendinį ir ar visi kintamieji yra naudojami lygčių sistemoje.

4. DISKUSIJOS

Atsižvelgę į rezultatus, gautus 2.1. skyrelyje, galime teigti, kad braidų grupės B_n elementai yra teisingai atvaizduojami į matricų grupę $M(n, Z_p[x_1, x_2, \ldots, x_m])$, išsaugant visas braidų grupės savybes.

2.2. skyrelio pirmame pavyzdyje, kai lauko parametras $p = 2$, buvo pastebima aiški Burau vaizdavimo matricų struktūra. Dabar nežinome, kaip galima būtų pasinaudoti šia struktūra, bet tikėtina, kad ateityje kenkėjas galės pasinaudoti ja siekdamas išspręsti vienkryptės funkcijos atvirkštinės radimo problemą. Ši struktūra randama visose matricose, gautose atlikus Burau vaizdavimą, kai $p = 2$. Dėl šios priežasties sudarant vienkryptę funkciją, negalima naudoti Burau vaizdavimo į matricų grupę virš daugianarių žiedo $Z_2[x_1, x_2, \ldots, x_m]$.

Antrame pavyzdyje, kai lauko parametras $p = 3$ ir požiedžis T įgyja tik 6 reikšmes iš 18 maksimaliai galimų, buvo gaunama analogiška Burau vaizdavimo matricų struktūra. Trečiame pavyzdyje, kai lauko parametras $p = 3$ ir požiedis T įgyja visas galimas 18 reikšmių, jau nebepavyksta įžvelgti aiškios vaizdavimo matricos struktūros. Todėl reikalausime, kad daugianaris t ne tik neturėtų nulinių narių, bet ir su juo gaunamas požiedis T turėtų maksimalų galimą elementų skaičių.

Atlikus tyrimą aprašytą 2.3. skyrelyje nustatėme, kad vienkryptę funkciją nusakinti keleto kintamųjų kvadratinių daugianarių lygčių sistema turės nemažiau kaip 1600 kintamųjų ir Burau vaizdavimo matricos užims mažiausiai atminties (288 baitus), kai parenkamos parametrų reikšmės $p = 3$, $m = 3$ ir $n = 6$.

Vaizdavimą braidų grupės B_6 elementus į matricų grupę $M(6, Z_3[x_1, x_2, x_3])$ iš tikro yra atliekamas vaizdavimas į požiedį T. Šiuo atveju požiedį T sudaro 18 skirtingų elementų, o virš jo galima sudaryti 18^{36} skirtingų matricų, t. y. apie 2^{150} skirtingų matricų. Vadinasi, atliekant pilną vienkryptėje funkcijoje galimų matricų X arba Y perrinkimą, reikia atlikti apie 2^{150} patikrinimų. Tai yra apie $4{,}5\times10^6$ kartų daugiau negu reikia atlikti patikrinimų perrenkant visus galimus funkcijos AES kriptografinius 128 bitų raktus. Priminsime, kad perrinkti visus galimus funkcijos AES raktus, reikės atlikti 2^{128} patikrinimų. Galime teigti, kad šiuo požiūriu siūloma vienkryptė funkcija yra pranašesnė už AES funkciją.

Grupė B_6 turi 5 elementariuosius braidus ir galima sudaryti 5^h skirtingus braidų žodžius, kai h yra sudaromų žodžių ilgis. Kad galimų žodžių skaičius nebūtų mažesnis už 2^{128}, parametras h turi būti lygus $\left|\log_5 2^{128}\right| = 56$.

Kadangi nustatėme, kad $p = 3$, tai atsižvelgiant į **2.3. lentelės** duomenis požiedis T turi būtinai būti sudarytas iš 18 elementų.

Atlikus tyrimą aprašytą 2.4. skyrelyje **2.6. lentelėje**, **2.7. lentelėje** ir **2.8. lentelėje** gauti rezultatai patvirtina teiginį, kad Griobnerio bazių skaičiavimas priklauso nuo pasirinktos daugianario narių rikiavimo tvarkos. Kai naudojama laipsninė ir atvirkštinė leksikografinė rikiavimo tvarka, tai rezultatai gaunami greičiausiai, o kai naudojama leksikografinė rikiavimo tvarka, tai skaičiavimai trunka ilgiausiai.

Iš (2.2), (2.3), (2.4) ir (2.5) formulių matome, kad **1.3. algoritmo** vykdymo trukmė yra eksponentės eilės. 5. Pavyzdyje pateiktai lygčių sistemai sudaryti Griobnerio bazes (2.4) ir (2.5) formulės prognozuoja, kad atitinkamai reikės 2851996 ms ≈ 48 min. ir 5185987 ms ≈ 1,4 val., bet rezultatų nepavyko gauti net per 24 val. 5. Pavyzdyje pateiktai lygčių sistemai, kai naudojama laipsninė ir leksikografinė daugianario narių rikiavimo tvarka, sudaryti Griobnerio bazes (2.2) ir (2.3) formulės prognozuoja, kad atitinkamai reikės 22921106 ms ≈ 6,4 val. ir 43594083 ms ≈ 12 val., bet rezultatų taip pat nepavyko gauti net per 24 val. Vadinasi, šios prognozės formulės nėra tikslios ir tikroji Griobnerio bazių sudarymo trukmė yra didesnė. Tai tik patvirtina, kad **1.3. algoritmo** vykdymo trukmė yra bent eksponentės eilės.

Pasižiūrėjus į ketvirtame pavyzdyje Griobnerio bazių pagalba suprastintą lygčių sistemą matome, kad ji išliko kvadratinė. Visais kitais atvejais, kai pavyko lygčių sistemas suprastinti Griobnerio bazių pagalba, supaprastintos lygčių sistemos taip pat išliko kvadratinės. Galima teigti, kad vienkryptę funkciją nusakančioms keleto kintamųjų kvadratinių daugianarių lygčių sistemoms (net ir sėkmingai pritaikius Griobnerio bazių radimo algoritmą) bus sunku rasti teisinga sprendinį, nes lygčių sistemos išliks kvadratinės.

Šifravimo sistemą AES nusakančioje lygčių sistemoje lygčių yra daugiau nei kintamųjų (kintamųjų – 1600, o lygčių – 8000). Literatūros šaltinyje (22) nurodyta, kad lygčių sistemoje turint daugiau lygčių nei kintamųjų sudaryti Griobnerio bazes yra paprasčiausia. Iš formulių (1.6) ir (1.7) matome, kad siūlomą vienkryptę funkciją nusakanti lygčių sistema turi daugiau kintamųjų negu lygčių. Vadinasi, šiuo požiūriu siūloma vienkryptė funkcija yra pranašesnė už AES funkciją.

IŠVADOS

1. 1.1.2 skyrelyje pateiktas Burau vaizdavimas yra teisingas.

2. Burau vaizdavimas yra atliekamas ne į visą matricų grupę $M(n, Z_p[x_1, \dots, x_m])$, o į pogrupį $M(n, T)$, kur $T \subset M(n, Z_p[x_1, \dots, x_m])$.

3. Siūlomos vienkryptės funkcijos parametrų reikšmės turi būti: $p = 3$, $m = 3$, $n = 6$ ir $h = 56$.

4. Vienkryptės funkcijos parametras, daugianaris t, turi būti sudarytas iš visų įmanomų vienanarių žiede $Z_3[x_1, x_2, x_3]$, padaugintų iš nenulinių koeficientų, kad vienkryptę funkciją nusakanti keleto kintamųjų kvadratinių daugianarių lygčių sistema turėtų maksimalų kintamųjų skaičių.

5. Požiedyje T, gaunamame atliekant Burau vaizdavimą, būtinai turi būti 18 elementų.

6. Keleto kintamųjų kvadratinių daugianarių lygčių sistemai, nusakančiai vienkryptę funkciją, Griobnerio bazių sudarymas trunka ne mažiau kaip eksponentinį laiką.

ŠALTINIAI IR LITERATŪRA

1. Ajwa, I.A., Liu, Z., Wang, P.S. Gröbner Bases Algorithm, 2003, 14 p.
2. Anshel, I., Anshel, M., Goldfeld, D. An algebraic method for public-key cryptography / *Mathematical Research Letters 6*, 1999, p. 287-291.
3. Bigelow, S. The Burau representation is not faithful for $n = 5$ / *Geometry & Topology*, Vol. 3, 1999, p. 397-4004.
4. Birman, J.S., Brendle T.E. Braids: A Survay, 2004, 91 p.
5. Brute force attack. Prieiga per internetą: http://en.wikipedia.org/wiki/Brute_force_attack.
6. Campagna, M.J. Algorithms in Braid Groups, 2003, 18 p.
7. Courtois, N., Klimov, A., Patarin, J., Shamir, A. Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations, 2000, 18 p.
8. Courtois, N., Pieprzyk, J. Cryptanalysis of Block Ciphers with Overdefined Systems of Equations, 2002, 16 p.
9. Cox, D., Little, J., O'Shea, D. Ideals, Varieties, and Algorithms, Third Edition, 2007, 560 p.
10. Dehornoy, P. Weak Faithfulness Properties for Burau Representation, 1995, 29 p.
11. Faug ère, J.-C. A new efficient algorithm for computing Gröbner bases (F_4) / *Journal of Pure and Applied Algebra*, Vol. 139, 1999, p. 61-88.
12. Franco, N., González-Meneses, J. Computation of Centralizers in Braid groups and Garside Groups, 2002, 13 p.
13. Franco, N., González-Meneses, J. Conjugacy problem for braid groups and Garside groups, 2002, 21 p.
14. Garber, D. Braid Group Cryptography, 2008, 79 p.
15. Garey, M.R., Johnson, D.S. Computers and Intractability: A Guide to the Theory of NP-Completeness, 1979, 340 p.
16. Ko, K.H., Lee, S.J., Cheon, J.H., Han, J.W., Kang, J.S., Park, C. New Public key Cryptosystem Using Braid Groups / *Lecture Notes in Computer Science*, Vol. 1880 , 2000, p. 166-183.
17. Mahlburg, K. An Overview of Braid Group Cryptography, 2004, 13 p.
18. Myasnikov, A., Shpilrain, V., Ushakov, A. A practical attack on a braid group based cryptographic protocol, 2005, 12 p.
19. Myasnikov, A., Shpilrain, V., Ushakov, A. Random Subgroups of Braid Groups: An Approach to Cryptanalysis of Braid Group based Cryptographic Protocols, 2006, 13 p.
20. Plukas, K. Skaitiniai metodai ir algoritmai, 2001, 548 p.
21. Sakalauskas, E., Listopaskis, N., Dosinas, G.S., Lukšys, K., Katvickis, A. Kriptografinės sistemos, Technologija, 2008, 166 p.

22. Shamir, A., Kipnis, A. Cryptanalysis of the HFE Public Key Cryptosystem, 1999, 15 p.
23. Shpilrain, V., Ushakov, A. The conjungacy search problem in public key cryptography: unnecessary and insufficient, 2004, 5 p.
24. Vitkus, P., Katvickis, A. Key aggreement protocol using elliptic curve matrix power function / *Advanced Studies in Software and Knowledge Engineering. No. 4, Information Science & Computing : suppl. to "Informations Technologies and Knowledge"*, Vol. 2, Sofia, ITHEA, 2008, p. 103-106.
25. Vitkus, P., Tvarijonas, P. Vienos vienkryptės funkcijos apibrėžimas panaudojant braid grupę / *Taikomoji matematika VII studentų konferencijos pranešimų medžiaga*, Kauno Technologijos Universitetas, 2008, p. 29-30.

1 PRIEDAS. BURAU VAIZDAVIMO TIKRINIMO PAVYZDYS

Pasirenkame vienkryptės funkcijos parametrus: $n = 4$, $p = 3$, $m = 3$, t = 1*2088C0+2*31C0437.

Programos pagalba gauname rezultatus:

```
Skaičiavimai truko: 39ms
t = 1*2088C0+2*31C0437

Generatoriai:

Gen[0]:
1*1 1*0 1*0 1*0
1*0 1*1 1*0 1*0
1*0 1*0 1*1 1*0
1*0 1*0 1*0 1*1

Gen[1]:
1*31C0436+2*2088C1 1*2088C0+2*31C0437 1*0 1*0
1*1 1*0 1*0 1*0
1*0 1*0 1*1 1*0
1*0 1*0 1*0 1*1

Gen[2]:
1*1 1*0 1*0 1*0
1*0 1*31C0436+2*2088C1 1*2088C0+2*31C0437 1*0
1*0 1*1 1*0 1*0
1*0 1*0 1*0 1*1

Gen[3]:
1*1 1*0 1*0 1*0
1*0 1*1 1*0 1*0
1*0 1*0 1*31C0436+2*2088C1 1*2088C0+2*31C0437
1*0 1*0 1*1 1*0

Žodis X: 2;1;2;3;2;1;1;3;3;1;
1*31C0436+2*2088C1 1*B3507+2*3248840 1*B3507+2*3248840 1*3073180+2*100030
1*31C0436+2*2088C1 1*B3507+2*3248840 1*3073180+2*100030 1*B3507+2*3248840
1*100031+2*3073180 1*3073180+2*100030 1*0 1*0
1*3248840+2*B3506 1*B3507+2*3248840 1*0 1*0
Skirtingų elementų: 6/16

Žodis A: 2;3;3;3;3;3;2;1;1;2;
1*3248840+2*B3506 1*3248840+2*B3507 1*3248840+2*B3507 1*0
1*31C0436+2*2088C1 1*3248840+2*B3507 1*B3507+2*3248840 1*2088C0+2*31C0437
1*100031+2*3073180 1*80487+2*33B970 1*3248840+2*B3507 1*0
1*100031+2*3073180 1*3248840+2*B3507 1*2088C0+2*31C0437 1*3248840+2*B3507
Skirtingų elementų: 8/16

Žodis Y: 1;1;3;1;3;1;2;3;1;3;
1*31C0436+2*2088C1 1*80487+2*33B970 1*0 1*B3507+2*3248840
1*31C0436+2*2088C1 1*3248840+2*B3507 1*3073180+2*100030 1*0
1*3248840+2*B3506 1*0 1*3248840+2*B3507 1*3248840+2*B3507
1*31C0436+2*2088C1 1*0 1*B3507+2*3248840 1*80487+2*33B970
Skirtingų elementų: 7/16

Žodis B = XAY:
1*3248840+2*B3506 1*3248840+2*B3507 1*B3507+2*3248840 1*B3507+2*3248840
1*33B970+2*80486 1*B3507+2*3248840 1*3073180+2*100030 1*0
1*100031+2*3073180 1*B3507+2*3248840 1*B3507+2*3248840 1*80487+2*33B970
1*33B970+2*80486 1*3073180+2*100030 1*B3507+2*3248840 1*0
Skirtingų elementų: 8/16
```

Patikrinimas:
Fundamentalusis braidas D:
1*31C0436+2*2088C1 1*B3507+2*3248840 1*3248840+2*B3507 1*2088C0+2*31C0437
1*31C0436+2*2088C1 1*B3507+2*3248840 1*80487+2*33B970 1*0
1*31C0436+2*2088C1 1*2088C0+2*31C0437 1*0 1*0
1*1 1*0 1*0 1*0
Skirtingų elementų: 7/16

Centro generatorius C:
1*3248840+2*B3506 1*B3507+2*3248840 1*3248840+2*B3507 1*B3507+2*3248840
1*31C0436+2*2088C1 1*2088C0+2*31C0437 1*3248840+2*B3507 1*B3507+2*3248840
1*31C0436+2*2088C1 1*B3507+2*3248840 1*3073180+2*100030 1*B3507+2*3248840
1*31C0436+2*2088C1 1*B3507+2*3248840 1*3248840+2*B3507 1*2088C0+2*31C0437
Skirtingų elementų: 6/16

Lygybė G[1] * D = D * G[n-1] **teisinga**:
1*31C0436+2*2088C1 1*B3507+2*3248840 1*3073180+2*100030 1*B3507+2*3248840
1*31C0436+2*2088C1 1*B3507+2*3248840 1*3248840+2*B3507 1*2088C0+2*31C0437
1*31C0436+2*2088C1 1*2088C0+2*31C0437 1*0 1*0
1*1 1*0 1*0 1*0
Skirtingų elementų: 7/16

Lygybė G[2] * D = D * G[n-2] **teisinga**:
1*31C0436+2*2088C1 1*B3507+2*3248840 1*3248840+2*B3507 1*2088C0+2*31C0437
1*31C0436+2*2088C1 1*3073180+2*100030 1*3248840+2*B3507 1*0
1*31C0436+2*2088C1 1*B3507+2*3248840 1*80487+2*33B970 1*0
1*1 1*0 1*0 1*0
Skirtingų elementų: 8/16

Lygybė G[3] * D = D * G[n-3] **teisinga**:
1*31C0436+2*2088C1 1*B3507+2*3248840 1*3248840+2*B3507 1*2088C0+2*31C0437
1*31C0436+2*2088C1 1*B3507+2*3248840 1*80487+2*33B970 1*0
1*3248840+2*B3506 1*B3507+2*3248840 1*0 1*0
1*31C0436+2*2088C1 1*2088C0+2*31C0437 1*0 1*0
Skirtingų elementų: 7/16

Lygybė X * C = C * X **teisinga**:
1*100031+2*3073180 1*3248840+2*B3507 1*0 1*80487+2*33B970
1*100031+2*3073180 1*3248840+2*B3507 1*2088C0+2*31C0437 1*3248840+2*B3507
1*33B970+2*80486 1*80487+2*33B970 1*3248840+2*B3507 1*B3507+2*3248840
1*B3506+2*3248841 1*3248840+2*B3507 1*3248840+2*B3507 1*B3507+2*3248840
Skirtingų elementų: 8/16

Lygybė A * C = C * A **teisinga**:
1*B3506+2*3248841 1*0 1*B3507+2*3248840 1*B3507+2*3248840
1*100031+2*3073180 1*0 1*0 1*3073180+2*100030
1*33B970+2*80486 1*2088C0+2*31C0437 1*B3507+2*3248840 1*B3507+2*3248840
1*33B970+2*80486 1*0 1*80487+2*33B970 1*0
Skirtingų elementų: 8/16

Lygybė Y * C = C * Y **teisinga**:
1*100031+2*3073180 1*2088C0+2*31C0437 1*3248840+2*B3507 1*3248840+2*B3507
1*100031+2*3073180 1*0 1*2088C0+2*31C0437 1*B3507+2*3248840
1*B3506+2*3248841 1*B3507+2*3248840 1*B3507+2*3248840 1*0
1*100031+2*3073180 1*B3507+2*3248840 1*0 1*2088C0+2*31C0437
Skirtingų elementų: 6/16

Lygybė B * C = C * B **teisinga**:
1*B3506+2*3248841 1*0 1*0 1*3248840+2*B3507
1*31C0436+2*2088C1 1*3248840+2*B3507 1*2088C0+2*31C0437 1*B3507+2*3248840
1*33B970+2*80486 1*3248840+2*B3507 1*0 1*2088C0+2*31C0437
1*31C0436+2*2088C1 1*80487+2*33B970 1*0 1*B3507+2*3248840
Skirtingų elementų: 8/16

2 PRIEDAS. POŽIEDŽIO *T* ELEMENTAI

1. Kai $p = 2$, tai požiedį T sudaro daugianariai:

```
0
1
t
t+1
```

2. Kai $p = 3$, tai požiedį T sudaro:

```
0
1
t
2*t+1
2*t+2
t+2
t^2
2*t^2+t
2*t^2+t+1
2*t^2+2*t
2*t^2+1
t^2+t+1
t^2+t+2
t^2+2*t+1
t^2+2
2*t^2+2
t^2+2*t
2*t^2+2*t+2
```

3. Kai $p = 5$, tai požiedį T sudaro:

```
0
1
t
4*t+1
4*t+2
3*t+2
t^2
4*t^2+t
4*t^2+t+1
4*t^2+2*t
4*t^2+1
t^2+3*t+1
t^2+3*t+2
t^2+4*t+1
t^2+2*t+2
t^2+2*t+3
3*t+3
4*t^2+2
t^2+t+3
t^2+t+4
t^2+4
2*t^2+2*t+2
2*t+3
2*t^2+3
2*t^2+4
2*t^2+t+3
2*t^2+4*t+4
3*t^2+2*t
2*t^2+t+2
4*t^2+4*t+2
3*t^2+t+2
3*t^2+4*t+3
3*t^2+4*t+4
3*t^2+3
3*t^2+3*t+4
t^3
4*t^3+t^2
4*t^3+t^2+1
4*t^3+t^2+t
4*t^3+t^2+4*t+1
2*t^3+4*t^2+4*t+1
2*t^3+4*t^2+3*t+1
2*t^3+4
2*t^3+3*t
2*t^3+3*t+1
2*t^3+4*t
2*t^3+2*t+1
t^3+t^2+3
t^3+2*t^2+3*t+4
3*t^3+2*t^2+2*t+3
2*t^3+4*t+4
2*t^3+t^2+2*t
2*t^3+t^2+2*t+1
2*t^3+t^2+3*t
2*t^3+t^2+t+1
2*t^3+2*t^2+1
2*t^3+2*t^2+t+1
2*t^3+4*t^2+2*t+2
2*t^3+3*t^2+3*t+2
2*t^3+4*t^2+t+3
t^3+t^2+t+2
t^4
4*t^4+t^3
4*t^4+t^3+1
4*t^4+t^3+t
4*t^4+t^3+4*t+1
4*t^4+t^3+4*t+2
4*t^4+t^3+3*t+2
4*t^4+t^3+t^2
4*t^4+t^3+4*t^2+t
4*t^4+t^3+4*t^2+t+1
4*t^4+t^3+4*t^2+2*t
4*t^4+t^3+4*t^2+1
4*t^4+t^3+t^2+3*t+1
4*t^4+t^3+t^2+3*t+2
4*t^4+t^3+t^2+4*t+1
4*t^4+t^3+t^2+2*t+2
4*t^4+t^3+t^2+2*t+3
4*t^4+t^3+3*t+3
4*t^4+t^3+4*t^2+2
t^4+2*t^3+4*t+4
t^4+2*t^3+3*t+4
t^4+2*t^3+t^2+2
t^4+2*t^3+t^2+3*t+3
t^4+2*t^3+t^2+3*t+4
t^4+2*t^3+t^2+4*t+3
t^4+2*t^3+t^2+2*t+4
t^4+3*t^3+t^2+2*t+4
t^4+3*t^3+4*t+3
t^4+3*t^3+2*t^2+4*t
t^4+3*t^3+t^2+t+4
t^4+3*t^3+3*t^2+4*t
t^4+3*t^3+3*t^2+3*t
t^4+3*t^3+3*t+3
t^4+2*t^3+2*t^2+3*t+3
t^4+2*t^3+2*t^2+2*t+3
t^4+2*t^3+t^2+4*t+2
t^4+4*t^3+4*t^2+3*t+4
t^4+4*t^3+4*t^2+2*t+4
t^4+4*t^3+3*t^2+3
t^4+4*t^3+3*t^2+4*t+3
t^4+4*t^3+2*t+4
t^4+4*t^3+t+4
t^4+4*t^3+2*t^2+2*t+2
t^4+4*t^3+2*t^2+t+2
t^4+3*t^3+4*t^2+2
t^4+4*t^3+t^2+t+4
t^4+4*t^3+t^2+4
t^4+2*t^3+4*t^2+4*t
t^4+2*t^3+4*t^2+3*t
t^4+2*t^3+3*t^2+t+4
t^4+2*t^3+3*t^2+4
t^4+2*t^3+3*t
t^4+2*t^3+2*t
t^4+2*t^3+t+1
t^4+4*t^3+t^2+4*t
t^4+4*t^3+t^2+3*t+1
t^4+3*t^3+3*t^2+2*t+1
t^4+3*t^3+2*t+4
t^4+3*t^3+3*t+4
```

```
2*t^2+3*t+1
4*t^3+t^2+4*t+2
4*t^3+t^2+3*t+2
4*t^3+2*t^2
4*t^3+t
4*t^3+t+1
4*t^3+2*t
4*t^3+1
4*t^3+2*t^2+3*t+1
4*t^3+2*t^2+3*t+2
4*t^3+2*t^2+4*t+1
4*t^3+2*t^2+2*t+2
t^3+3*t^2+t
t^3+3*t^2+t+1
t^3+3*t^2+2*t
t^3+3*t^2+1
t^3+3*t^2+2
t^3+3*t^2+4*t+2
t^3+4*t^2+t
t^3+2*t^2+2*t
t^3+2*t^2+2*t+1
t^3+2*t^2+3*t
t^3+2*t^2+t+1
t^3+4*t^2+4*t+1
t^3+4*t^2+4*t+2
t^3+4*t^2+1
t^3+4*t^2+3*t+2
3*t^2+2*t+1
t^3+3*t^2+4*t+3
t^3+3*t^2+3*t+3
t^3+2*t^2+t+2
t^3+2*t^2+2
t^3+4*t^2+3*t+3
t^3+4*t^2+2*t+3
3*t^2+3*t
t^3+t^2+3*t
t^3+t^2+3*t+1
t^3+t^2+4*t
t^3+t^2+2*t+1
3*t^2+t+1
t^3+4*t+1
t^3+2*t+2
t^3+2*t+3
t^3+3*t+2
t^3+t+3
4*t^3+3*t^2+2*t+1
4*t^3+3*t^2+2*t+2
4*t^3+3*t^2+3*t+1
4*t^3+3*t^2+t+2
4*t^3+3*t^2+t+3
4*t^3+3*t^2+3
4*t^3+4*t^2+2*t+1
4*t^3+4*t^2+3*t+2
4*t^3+3*t^2+4
4*t^3+3*t^2+4*t+4
4*t^3+2*t^2+2*t+3
4*t^3+2*t^2+t+3
4*t^3+4*t^2+4*t+4
4*t^3+4*t^2+3*t+4
4*t^3+2
4*t^4+t^3+t^2+t+3
4*t^4+t^3+t^2+t+4
4*t^4+t^3+t^2+4
4*t^4+t^3+2*t^2+2*t+2
4*t^4+t^3+2*t+3
4*t^4+t^3+2*t^2+3
4*t^4+t^3+2*t^2+4
4*t^4+t^3+2*t^2+t+3
4*t^4+t^3+2*t^2+4*t+4
4*t^4+t^3+3*t^2+2*t
4*t^4+t^3+2*t^2+t+2
4*t^4+t^3+4*t^2+4*t+2
4*t^4+t^3+3*t^2+t+2
4*t^4+t^3+3*t^2+4*t+3
4*t^4+t^3+3*t^2+4*t+4
4*t^4+t^3+3*t^2+3
4*t^4+t^3+3*t^2+3*t+4
4*t^4+2*t^3
4*t^4+t^2
4*t^4+t^2+1
4*t^4+t^2+t
4*t^4+t^2+4*t+1
4*t^4+t^3+2*t^2+3*t+1
4*t^4+t^2+4*t+2
4*t^4+t^2+3*t+2
4*t^4+2*t^2
4*t^4+t
4*t^4+t+1
4*t^4+2*t
4*t^4+1
4*t^4+2*t^2+3*t+1
4*t^4+2*t^2+3*t+2
4*t^4+2*t^2+4*t+1
4*t^4+2*t^2+2*t+2
4*t^4+2*t^3+3*t^2+t
4*t^4+2*t^3+3*t^2+t+1
4*t^4+2*t^3+3*t^2+2*t
4*t^4+2*t^3+3*t^2+1
4*t^4+2*t^3+3*t^2+2
4*t^4+2*t^3+3*t^2+4*t+2
4*t^4+2*t^3+4*t^2+t
4*t^4+2*t^3+2*t^2+2*t
4*t^4+2*t^3+2*t^2+2*t+1
4*t^4+2*t^3+2*t^2+3*t
4*t^4+2*t^3+2*t^2+t+1
4*t^4+2*t^3+4*t^2+4*t+1
4*t^4+2*t^3+4*t^2+4*t+2
4*t^4+2*t^3+4*t^2+1
4*t^4+2*t^3+4*t^2+3*t+2
4*t^4+t^3+3*t^2+2*t+1
4*t^4+2*t^3+3*t^2+4*t+3
4*t^4+2*t^3+3*t^2+3*t+3
4*t^4+2*t^3+2*t^2+t+2
4*t^4+2*t^3+2*t^2+2
4*t^4+2*t^3+4*t^2+3*t+3
4*t^4+2*t^3+4*t^2+2*t+3
4*t^4+t^3+3*t^2+3*t
4*t^4+2*t^3+t^2+3*t
4*t^4+2*t^3+t^2+3*t+1
4*t^4+2*t^3+t^2+4*t
4*t^4+2*t^3+t^2+2*t+1
4*t^4+t^3+3*t^2+t+1
4*t^4+2*t^3+4*t+1
t^4+4*t^3+2*t^2+3
t^4+4*t^3+2*t^2+4
t^4+4*t^3+2*t^2+t+3
t^4+4*t^3+2*t^2+4*t+4
t^4+2*t^3+2*t^2+t+4
t^4+2*t^3+2*t^2+2*t+4
t^4+4*t^3
t^4+4*t^3+3*t^2+3*t+4
t^4+4*t^3+3*t^2+4*t+4
t^4+2*t^3+3*t^2+4*t
t^4+t^3+3*t^2
t^4+t^3+3*t^2+1
t^4+t^3+3*t^2+t
t^4+t^3+3*t^2+4*t+1
t^4+t^3+3*t^2+4*t+2
t^4+t^3+3*t^2+3*t+2
t^4+t^3+4*t^2
t^4+t^3+2*t^2+t
t^4+t^3+2*t^2+t+1
t^4+t^3+2*t^2+2*t
t^4+t^3+2*t^2+1
t^4+t^3+4*t^2+3*t+1
t^4+t^3+4*t^2+3*t+2
t^4+t^3+4*t^2+4*t+1
t^4+t^3+4*t^2+2*t+2
t^4+t^3+2*t+1
t^4+t^3+2*t+2
t^4+t^3+3*t+1
t^4+t^3+t+2
t^4+t^3+t+3
t^4+t^3+3
t^4+t^3+t^2+2*t+1
t^4+t^3+t^2+2
t^4+t^3+t^2+3
t^4+t^3+t^2+t+2
t^4+t^3+t^2+4*t+3
t^4+t^3+4
t^4+t^3+4*t+4
t^4+t^3+4*t^2+2*t+3
t^4+t^3+4*t^2+t+3
t^4+t^3+t^2+4*t+4
t^4+t^3+t^2+3*t+4
t^4+t^3+2*t^2+2
t^4+t^3+2*t^2+3*t+3
t^4+t^3+2*t^2+3*t+4
t^4+t^3+2*t^2+4*t+3
t^4+t^3+2*t^2+2*t+4
t^4+4*t^2+t
t^4+2*t^2+2*t
t^4+2*t^2+2*t+1
t^4+2*t^2+3*t
t^4+2*t^2+t+1
t^4+2*t^2+t+2
t^4+2*t^2+2
t^4+3*t^2+2*t
t^4+t^2+3*t
t^4+t^2+3*t+1
t^4+t^2+4*t
t^4+t^2+2*t+1
t^4+3*t^2+1
t^4+3*t^2+2
t^4+3*t^2+t+1
t^4+3*t^2+4*t+2
```

```
4*t^3+3*t+3
4*t^3+3*t+4
4*t^3+4*t+3
4*t^3+2*t+4
4*t^3+4*t^2+3*t
2*t^2+4*t
4*t^3+4*t^2+2*t
t^3+4*t^2+2*t+4
t^3+t+4
4*t^3+3*t^2+4*t
4*t^3+4*t+2
4*t^3+4*t^2+t+1
4*t^3+3*t^2+3*t
4*t^2+4*t+3
3*t^3+4*t+3
3*t^3+4*t+4
3*t^3+3
3*t^3+3*t+4
4*t^3+t^2+2*t+4
3*t^3+3*t
3*t^3+2*t
3*t^3+t^2+4*t+3
3*t^3+4*t^2+3
3*t^3+4*t^2+4
3*t^3+4*t^2+t+3
3*t^3+4*t^2+4*t+4
3*t^3+t^2+2*t+4
3*t^3+t^2+2*t
3*t^3+t^2+3*t+4
3*t^3+t^2+t
2*t^2+3*t
4*t^3+2*t^2+t+4
2*t^2+2*t+1
t^2+4*t
2*t+4
t+4
4*t^3+2*t^2+4
4*t^2+3*t+4
3*t^3+2*t^2+t+4
3*t^3+2*t^2+t
3*t^3+2*t^2+2*t+4
3*t^3+2*t^2
3*t^3+2*t^2+1
3*t^3+2*t^2+4*t+1
3*t^3+3*t^2+t+4
3*t^3+3*t^2+4*t
3*t^3+3*t^2+4*t+1
3*t^3+3*t^2
3*t^3+3*t^2+3*t+1
3*t^3+2*t^2+4*t+2
3*t^3+2*t^2+3*t+2
3*t^3+t^2+t+1
3*t^3+t^2+1
3*t^3+3*t^2+3*t+2
3*t^3+3*t^2+2*t+2
3*t^3+4*t^2+4*t
3*t^3+4*t^2+2*t+1
3*t^3+4*t^2+2*t+2
3*t^3+4*t^2+3*t+1
3*t^3+4*t^2+t+2
4*t^3+t^2+3*t+3
t^3+3*t^2+3*t+4
t^3+2*t^2+3
4*t^4+2*t^3+2*t+2
4*t^4+2*t^3+2*t+3
4*t^4+2*t^3+3*t+2
4*t^4+2*t^3+t+3
4*t^4+3*t^2+2*t+1
4*t^4+3*t^2+2*t+2
4*t^4+3*t^2+3*t+1
4*t^4+3*t^2+t+2
4*t^4+3*t^2+t+3
4*t^4+3*t^2+3
4*t^4+4*t^2+2*t+1
4*t^4+4*t^2+2
4*t^4+4*t^2+3
4*t^4+4*t^2+t+2
4*t^4+4*t^2+4*t+3
4*t^4+3*t^2+4
4*t^4+3*t^2+4*t+4
4*t^4+2*t^2+2*t+3
4*t^4+2*t^2+t+3
4*t^4+4*t^2+4*t+4
4*t^4+4*t^2+3*t+4
4*t^4+2
4*t^4+3*t+3
4*t^4+3*t+4
4*t^4+4*t+3
4*t^4+2*t+4
t^4+3*t^3+t^2
t^4+3*t^3+t^2+1
t^4+3*t^3+t^2+t
t^4+3*t^3+t^2+4*t+1
t^4+3*t^3+t^2+4*t+2
t^4+3*t^3+t^2+3*t+2
t^4+3*t^3+2*t^2
t^4+3*t^3+t
t^4+3*t^3+t+1
t^4+3*t^3+2*t
t^4+3*t^3+1
t^4+3*t^3+2*t^2+3*t+1
t^4+3*t^3+2*t^2+3*t+2
t^4+3*t^3+2*t^2+4*t+1
t^4+3*t^3+2*t^2+2*t+2
t^4+3*t^3+2*t^2+2*t+3
t^4+3*t^3+t^2+3*t+3
t^4+3*t^3+2
t^4+3*t^3+2*t^2+t+3
t^4+3*t^3+2*t^2+t+4
t^4+3*t^3+3*t^2+2*t+2
t^4+3*t^3+t^2+2*t+3
t^4+3*t^3+3*t^2+3
t^4+3*t^3+3*t^2+4
t^4+3*t^3+3*t^2+t+3
t^4+3*t^3+4*t^2+2*t
t^4+3*t^3+4*t^2+3
t^4+3*t^3+4*t^2+4
t^4+3*t^3+4*t^2+t+3
t^4+3*t^3+4*t^2+4*t+4
t^4+3*t^3+4*t^2+3*t+4
t^4+4*t^3+t^2
t^4+2*t^3+2*t^2
t^4+2*t^2+3
t^4+2*t^2+4*t+3
t^4+t^2+2*t+2
t^4+t^2+t+2
t^4+3*t^2+4*t+3
t^4+3*t^2+3*t+3
t^4+4*t
t^4+4*t+1
t^4+3*t+1
t^4+4*t^2+1
t^4+4*t^2+3*t+2
t^4+4*t^2+3*t+3
t^4+4*t^2+4*t+2
t^4+4*t^2+2*t+3
t^4+2*t^2+4*t+4
t^4+2*t^2+3*t+4
t^4+t^2+t+3
t^4+t^2+3
t^4+3*t^2+3*t+4
t^4+3*t^2+2*t+4
t^4+3*t+2
t^4+2*t+2
t^4+4*t^2+2*t+4
t^4+4*t^2+t+4
t^4+4*t^2
t^4+4*t^2+4*t+1
t^4+t+3
t^4+t+4
t^4+2*t+3
t^4+4
t^4+t^3+2*t^2+4*t+2
t^4+t^3+3*t^2+2*t+3
t^4+t^3+3*t^2+2*t+4
t^4+t^3+3*t^2+3*t+3
t^4+t^3+3*t^2+t+4
t^4+t^3+4*t^2+4
t^4+t^3+4*t^2+t+4
t^4+t^3+4*t^2+4*t
t^4+t^3+t^2+3*t
t^4+t^3+t^2+2*t
t^4+t^3+3*t
t^4+t^3+4*t
4*t^4+3*t^3+2*t^2+t
4*t^4+3*t^3+2*t^2+t+1
4*t^4+3*t^3+2*t^2+2*t
4*t^4+3*t^3+2*t^2+1
4*t^4+3*t^3+2*t^2+2
4*t^4+3*t^3+2*t^2+4*t+2
4*t^4+3*t^3+3*t^2+t
4*t^4+3*t^3+t^2+2*t
4*t^4+3*t^3+t^2+2*t+1
4*t^4+3*t^3+t^2+3*t
4*t^4+3*t^3+t^2+t+1
4*t^4+3*t^3+3*t^2+4*t+1
4*t^4+3*t^3+3*t^2+4*t+2
4*t^4+3*t^3+3*t^2+1
4*t^4+3*t^3+3*t^2+3*t+2
4*t^4+3*t^3+3*t^2+3*t+3
4*t^4+3*t^3+2*t^2+4*t+3
4*t^4+3*t^3+t^2+t+2
4*t^4+3*t^3+3*t^2+2*t+3
4*t^4+3*t^3+3*t^2+2*t+4
4*t^4+3*t^3+3*t^2+t+4
```

```
3*t^2+2
t^3+4
t^3+4*t
t^3+t^2+t+3
t^3+4*t^2+t+4
t^3+t^2+4*t+4
t^3+t^2+4
t^3+3*t+1
t^3+4*t^2
t^3+3*t^2+2*t+4
t^3+2*t^2+4*t+4
t^3+t^2+2*t+2
2*t^3+4*t^2+3*t+2
4*t^2+3*t+3
2*t^3+2*t^2+4*t+2
2*t^3+2*t^2+4*t+3
2*t^3+2*t^2+2
2*t^3+2*t^2+3*t+3
2*t^3+2*t^2+3*t+4
2*t^3+2*t^2+2*t+4
2*t^3+3*t^2+4*t+2
2*t^3+t^2+2
2*t^3+t^2+3
2*t^3+t^2+t+2
2*t^3+t^2+4*t+3
2*t^3+3*t^2+2*t+3
2*t^3+3*t^2+2*t+4
2*t^3+3*t^2+3*t+3
2*t^3+3*t^2+t+4
2*t^3+2*t^2+2*t
2*t^3+2*t^2+t
2*t^3+t^2+4*t+4
2*t^3+t^2+3*t+4
2*t^3+3*t^2+t
2*t^3+3*t^2
2*t^3+t+2
2*t^3+t+3
2*t^3+2*t+2
2*t^3+3
t^3+2*t^2+4*t+3
2*t^3+4*t^2+2*t+3
2*t^3+4*t^2+4
2*t^3+4*t^2
2*t^3+4*t^2+t+4
2*t^3+4*t^2+4*t
4*t^3+t^2+t+4
4*t^3+2*t^2+4*t
3*t^3+t^2+3*t+3
3*t^3+2*t^2+3*t+3
4*t^2+2*t+4
3*t^3+3*t^2+4
3*t^3+4*t^2+3*t
3*t^3+t+1
3*t^3+t+2
3*t^3+2*t+1
3*t^3+2
4*t^3+t^2+2*t+3
3*t^3+3*t^2+2*t+3
3*t^3+3*t^2+t+3
3*t^3+t^2+4*t+2
3*t^3+t^2+2
2*t^3+3*t^2+1
2*t^3+3*t^2+4*t+1
t^4+2*t^3+2*t^2+1
t^4+2*t^3+2*t^2+t
t^4+2*t^3+2*t^2+4*t+1
t^4+3*t^3+3*t^2+3*t+1
t^4+2*t^3+2*t^2+4*t+2
t^4+2*t^3+2*t^2+3*t+2
t^4+2*t^3+3*t^2
t^4+2*t^3+t^2+t
t^4+2*t^3+t^2+t+1
t^4+2*t^3+t^2+2*t
t^4+2*t^3+t^2+1
t^4+2*t^3+3*t^2+3*t+1
t^4+2*t^3+3*t^2+3*t+2
t^4+2*t^3+3*t^2+4*t+1
t^4+2*t^3+3*t^2+2*t+2
t^4+4*t^3+4*t^2+t
t^4+4*t^3+4*t^2+t+1
t^4+4*t^3+4*t^2+2*t
t^4+4*t^3+4*t^2+1
t^4+4*t^3+4*t^2+2
t^4+4*t^3+4*t^2+4*t+2
t^4+4*t^3+t
t^4+4*t^3+3*t^2+2*t
t^4+4*t^3+3*t^2+2*t+1
t^4+4*t^3+3*t^2+3*t
t^4+4*t^3+3*t^2+t+1
t^4+4*t^3+4*t+1
t^4+4*t^3+4*t+2
t^4+4*t^3+1
t^4+4*t^3+3*t+2
t^4+3*t^3+4*t^2+2*t+1
t^4+4*t^3+4*t^2+4*t+3
t^4+4*t^3+4*t^2+3*t+3
t^4+4*t^3+3*t^2+t+2
t^4+4*t^3+3*t^2+2
t^4+4*t^3+3*t+3
t^4+4*t^3+2*t+3
t^4+3*t^3+4*t^2+3*t
t^4+4*t^3+2*t^2+3*t
t^4+4*t^3+2*t^2+3*t+1
t^4+4*t^3+2*t^2+4*t
t^4+4*t^3+2*t^2+2*t+1
t^4+3*t^3+4*t^2+t+1
t^4+4*t^3+t^2+4*t+1
t^4+4*t^3+t^2+2*t+2
t^4+4*t^3+t^2+2*t+3
t^4+4*t^3+t^2+3*t+2
t^4+4*t^3+t^2+t+3
t^4+2*t^3+4*t^2+2*t+1
t^4+2*t^3+4*t^2+2*t+2
t^4+2*t^3+4*t^2+3*t+1
t^4+2*t^3+4*t^2+t+2
t^4+2*t^3+4*t^2+t+3
t^4+2*t^3+4*t^2+3
t^4+2*t^3+2*t+1
t^4+2*t^3+2
t^4+2*t^3+3
t^4+2*t^3+t+2
t^4+2*t^3+4*t+3
t^4+2*t^3+4*t^2+4
t^4+2*t^3+4*t^2+4*t+4
t^4+2*t^3+3*t^2+2*t+3
t^4+2*t^3+3*t^2+t+3
4*t^4+3*t^3+4*t^2+3*t+2
4*t^4+3*t^3+2*t^2+3*t+3
4*t^4+3*t^3+4*t^2+t+3
4*t^4+3*t^3+4*t^2+t+4
4*t^4+3*t^3+4*t^2+2*t+3
4*t^4+3*t^3+4*t^2+4
4*t^4+3*t^3+3*t
4*t^4+3*t^3+4*t^2+2*t+2
4*t^4+3*t^3+t^2+2
4*t^4+3*t^3+2*t+2
4*t^4+3*t^3+3
4*t^4+3*t^3+4
4*t^4+3*t^3+t+3
4*t^4+3*t^3+4*t+4
4*t^4+4*t^3+2*t^2+t
4*t^4+3*t^3+4*t^2+4*t+1
4*t^4+4*t^3+2*t
4*t^4+4*t^3+2*t+1
4*t^4+4*t^3+3*t
4*t^4+4*t^3+t+1
4*t^4+4*t^3+t+2
4*t^4+4*t^3+2
4*t^4+4*t^3+t^2+2*t
4*t^4+4*t^3+4*t^2+3*t
4*t^4+4*t^3+4*t^2+3*t+1
4*t^4+4*t^3+4*t^2+4*t
4*t^4+3*t^3+3*t+1
4*t^4+4*t^3+3
4*t^4+4*t^3+4*t+3
4*t^4+4*t^3+4*t^2+2*t+2
4*t^4+4*t^3+4*t^2+t+2
4*t^4+4*t^3+t^2+4*t+3
4*t^4+4*t^3+t^2+3*t+3
4*t^4+3*t^3+4*t
4*t^4+4*t^3+3*t^2+4*t
4*t^4+4*t^3+3*t^2+4*t+1
4*t^4+4*t^3+3*t^2
4*t^4+4*t^3+3*t^2+3*t+1
4*t^4+3*t^3+2*t+1
4*t^4+4*t^3+2*t^2+1
4*t^4+4*t^3+2*t^2+3*t+2
4*t^4+4*t^3+2*t^2+3*t+3
4*t^4+4*t^3+2*t^2+4*t+2
4*t^4+4*t^3+2*t^2+2*t+3
4*t^4+2*t^3+3*t+1
4*t^4+2*t^3+t^2+t+2
4*t^4+2*t^3+t^2+t+3
4*t^4+2*t^3+t^2+2*t+2
4*t^4+2*t^3+t^2+3
4*t^4+2*t^3+t+4
4*t^4+2*t^3+4
4*t^4+2*t^3+t^2+4
4*t^4+2*t^3+t^2+4*t+4
4*t^4+2*t^3+2*t^2+4*t+3
4*t^4+2*t^3+2*t^2+4*t+4
4*t^4+2*t^3+2*t^2+3
4*t^4+2*t^3+2*t^2+3*t+4
4*t^4+3*t^3+2*t^2+3*t+4
```

```
2*t^4+3*t^3+3*t+2
2*t^4+4*t^3+2*t^2+4*t+4
2*t^4+4*t^3+2*t^2+3*t+4
2*t^4+2*t^3+2*t
2*t^4+2*t^3+t
2*t^4+2*t^3+4*t^2+4*t+4
2*t^4+2*t^3+4*t^2+3*t+4
2*t^4+2*t^3+t^2+t
2*t^4+2*t^3+t^2
2*t^4+2*t^3+t^2+4*t+1
2*t^4+4*t^3+2*t^2+2*t
2*t^4+4*t^3+2*t^2+t+1
2*t^4+3*t^3+4*t^2+1
2*t^4+3*t^3+t^2+4
2*t^4+3*t^3+t^2+t+4
2*t^4+4*t^3+3*t^2+3*t+3
2*t^4+4*t^3+3*t^2+3*t+4
2*t^4+4*t^3+3*t^2+4*t+3
2*t^4+4*t^3+3*t^2+2*t+4
2*t^4+2*t^3+3*t^2+4*t+4
2*t^4+2*t^3+3*t^2+4
2*t^4+4*t^3+t^2+3*t
2*t^4+4*t^3+4*t^2+t+4
2*t^4+4*t^3+4*t^2+2*t+4
2*t^4+2*t^3+4*t^2+2*t
2*t^4+t^3+4*t^2+3*t
2*t^4+t^3+4*t^2+3*t+1
2*t^4+t^3+4*t^2+4*t
2*t^4+t^3+4*t^2+2*t+1
2*t^4+t^3+4*t^2+2*t+2
2*t^4+t^3+4*t^2+t+2
2*t^4+t^3+3*t
2*t^4+t^3+3*t^2+4*t
2*t^4+t^3+3*t^2+4*t+1
2*t^4+t^3+3*t^2
2*t^4+t^3+3*t^2+3*t+1
2*t^4+t^3+t+1
2*t^4+t^3+t+2
2*t^4+t^3+2*t+1
2*t^4+t^3+2
2*t^4+t^3+t^2+1
2*t^4+t^3+t^2+2
2*t^4+t^3+t^2+t+1
2*t^4+t^3+t^2+4*t+2
2*t^4+t^3+t^2+4*t+3
2*t^4+t^3+t^2+3*t+3
2*t^4+t^3+2*t^2+1
2*t^4+t^3+2*t^2+3*t+2
2*t^4+t^3+2*t^2+3*t+3
2*t^4+t^3+2*t^2+4*t+2
2*t^4+t^3+2*t^2+2*t+3
2*t^4+t^3+t^2+3*t+4
2*t^4+t^3+t^2+2*t+4
2*t^4+t^3+3
2*t^4+t^3+4*t+3
2*t^4+t^3+2*t^2+2*t+4
2*t^4+t^3+2*t^2+t+4
2*t^4+t^3+3*t^2+3*t+2
2*t^4+t^3+3*t^2+t+3
2*t^4+t^3+3*t^2+t+4
2*t^4+t^3+3*t^2+2*t+3
2*t^4+t^3+3*t^2+4
2*t^4+4*t
3*t^4+4*t^3+t^2+3*t
3*t^4+4*t^3+t^2+t+1
3*t^4+3*t^3
3*t^4+4*t^3+2*t+2
3*t^4+4*t^3+t+2
3*t^4+4*t^3+4*t^2+4*t+1
3*t^4+4*t^3+4*t^2+3*t+1
3*t^4+4*t^3+t^2+t+2
3*t^4+4*t^3+t^2+2
3*t^4+3*t^3+t+4
3*t^4+4*t^3+3*t^2+t+4
3*t^4+4*t^3+3*t^2+t
3*t^4+4*t^3+3*t^2+2*t+4
3*t^4+4*t^3+3*t^2
3*t^4+3*t^3+4*t
3*t^4+4*t^3+2*t^2+2*t
3*t^4+4*t^3+2*t^2+1
3*t^4+4*t^3+2*t^2+2
3*t^4+4*t^3+2*t^2+t+1
3*t^4+4*t^3+2*t^2+4*t+2
3*t^4+2*t^3
3*t^4+2*t^3+1
3*t^4+2*t^3+t
3*t^4+2*t^3+4*t+1
3*t^4+2*t^3+4*t+2
3*t^4+2*t^3+3*t+2
3*t^4+2*t^3+t^2
3*t^4+2*t^3+t^2+3*t+1
3*t^4+2*t^3+t^2+3*t+2
3*t^4+2*t^3+t^2+4*t+1
3*t^4+2*t^3+t^2+2*t+2
3*t^4+2*t^3+3*t+3
3*t^4+2*t^3+2*t+3
3*t^4+2*t^3+4*t^2+2
3*t^4+2*t^3+4*t^2+4*t+2
3*t^4+2*t^3+t^2+2*t+3
3*t^4+2*t^3+t^2+t+3
3*t^4+2*t^3+2*t^2+3*t+1
3*t^4+2*t^3+2*t^2+t+2
3*t^4+2*t^3+2*t^2+t+3
3*t^4+2*t^3+2*t^2+2*t+2
3*t^4+2*t^3+2*t^2+3
3*t^4+4*t^2+4*t+4
3*t^4+4*t^2+4*t
3*t^4+4*t^2+4
3*t^4+4*t^2+3*t
3*t^4+4*t^2+3*t+1
3*t^4+4*t^2+2*t+1
3*t^4+4*t+4
3*t^4+3*t^2+4
3*t^4+3*t^2
3*t^4+3*t^2+t+4
3*t^4+3*t^2+4*t
3*t^4+2*t
3*t^4+2*t+1
3*t^4+3*t
3*t^4+t+1
3*t^4+t+2
3*t^4+4*t^2+2*t+2
3*t^4+3*t^2+4*t+1
3*t^4+2
3*t^4+3
3*t^4+4*t+3
4*t^4+3*t^3+t^2+3
4*t^4+3*t^3+3*t^2
4*t^4+3*t^3+2*t^2+2*t+4
4*t^4+3*t^3+4*t^2
4*t^4+3*t^3+4*t^2+4*t
4*t^4+3*t^3+t^2+4*t+3
4*t^4+4*t^3+4*t+4
4*t^4+4*t^3+3*t+4
4*t^4+4*t^3+4*t^2+t+3
4*t^4+4*t^3+4*t^2+3
4*t^4+4*t^3+t^2+3*t+4
4*t^4+4*t^3+t^2+2*t+4
4*t^4+4*t^3+3*t^2+3*t+2
4*t^4+4*t^3+3*t^2+2*t+2
4*t^4+3*t^3+t+2
4*t^4+4*t^3+2*t^2+2*t+4
4*t^4+4*t^3+2*t^2+t+4
4*t^4+2*t^3+4*t
4*t^4+2*t^3+4*t^2+2*t+4
4*t^4+2*t^3+4*t^2+t+4
4*t^4+4*t^3+2*t^2
4*t^4+4*t^3+2*t^2+4*t+1
4*t^4+3*t^3+4*t^2+3*t+1
4*t^4+3*t^3+t^2+3*t+4
4*t^4+3*t^3+t^2+4*t+4
4*t^4+4*t^3+3*t^2+t+3
4*t^4+4*t^3+3*t^2+t+4
4*t^4+4*t^3+3*t^2+2*t+3
4*t^4+4*t^3+3*t^2+4
4*t^4+2*t^3+3*t^2+2*t+4
4*t^4+2*t^3+3*t^2+3*t+4
4*t^4+4*t^3+t^2+t
4*t^4+4*t^3+4*t^2+4*t+4
4*t^4+4*t^3+4*t^2+4
4*t^4+2*t^3+4*t^2
4*t^4+3*t^2+3*t
4*t^4+3*t^2+4*t
4*t^4+4*t^2+3*t
4*t^4+2*t^2+4*t
4*t^4+4*t^2+t+1
4*t^4+4*t+2
4*t^4+t^2+2*t+3
4*t^4+t^2+2*t+4
4*t^4+t^2+3*t+3
4*t^4+t^2+t+4
4*t^4+t^3+3*t^2+2
4*t^4+t^3+4*t^2+3*t+3
4*t^4+t^3+4*t^2+3*t+4
4*t^4+t^3+4*t^2+4*t+3
4*t^4+t^3+4*t^2+2*t+4
4*t^4+t^3+t+4
4*t^4+t^3+2*t+4
4*t^4+t^3+2*t^2+4*t
4*t^4+t^3+2*t^2+3*t
4*t^4+t^3+t^2+4*t
4*t^4+4*t^2+2*t
4*t^4+2*t^2+4
4*t^4+2*t^2+t+4
4*t^4+t^3+2*t^2+2*t+1
t^4+t^3+t^2+t+1
t^4+t^2+4*t+4
t^4+t^2+4
t^4+3*t^2+t
```

```
2*t^4+3*t^2
2*t^4+3*t^2+1
2*t^4+3*t^2+t
2*t^4+3*t^2+4*t+1
2*t^4+3*t^2+4*t+2
2*t^4+3*t^2+3*t+2
2*t^4+4*t^2
2*t^4+2*t^2+t
2*t^4+2*t^2+t+1
2*t^4+2*t^2+2*t
2*t^4+2*t^2+1
2*t^4+4*t^2+3*t+1
2*t^4+4*t^2+3*t+2
2*t^4+4*t^2+4*t+1
2*t^4+4*t^2+2*t+2
2*t^4+3*t^2+3*t+3
2*t^4+3*t^2+2*t+3
2*t^4+2*t^2+2
2*t^4+2*t^2+4*t+2
2*t^4+4*t^2+2*t+3
2*t^4+4*t^2+t+3
2*t^4+t^2+2*t
2*t^4+t^2+2*t+1
2*t^4+t^2+t+1
2*t^4+3*t+1
2*t^4+t+2
2*t^4+t+3
2*t^4+2*t+2
2*t^4+3
2*t^4+3*t^2+2*t+4
2*t^4+3*t^2+t+4
2*t^4+2*t^2+4*t+3
2*t^4+2*t^2+3*t+3
2*t^4+4*t^2+t+4
2*t^4+4*t^2+4
2*t^4+t^2+t+2
2*t^4+t^2+2
2*t^4+4
2*t^4+4*t+4
2*t^4+3*t
2*t^4+2*t+1
2*t^4+t^2+4*t+3
2*t^4+t^2+4*t+4
2*t^4+t^2+3
2*t^4+t^2+3*t+4
2*t^4+t^3+3*t^2+2*t+2
2*t^4+t^3+4*t^2+3*t
2*t^4+t^3+4*t^2+4
2*t^4+t^3+3*t+4
2*t^4+t^3+3*t^2+t+4
2*t^4+t^3+4*t^2+t+4
2*t^4+t^3+3*t+4
2*t^4+t^3+3*t^2+2*t+3
2*t^4+t^3+4*t^2+2*t+4
3*t^4+3*t^3+2*t^2+3*t+4
3*t^4+3*t^3+2*t^2+3*t
3*t^4+t^2+t+1
3*t^4+4*t^2+t+2
3*t^4+t^2+4*t+2
3*t^4+t^2+4*t+3
3*t^4+t^2+2
3*t^4+t^2+3*t+3
3*t^4+2*t^2+t+4
3*t^4+t^2+1
3*t^4+3*t^2+3*t+1
3*t^4+2*t^2+1
3*t^4+2*t^2+3*t+2
3*t^4+2*t^2+3*t+3
3*t^4+2*t^2+4*t+2
3*t^4+2*t^2+2*t+3
3*t^4+t^3+4*t^2+4*t+4
3*t^4+t^2+2*t
3*t^4+t^3+2*t^2+4
3*t^4+t^3+2*t^2
3*t^4+t^3+2*t^2+t+4
3*t^4+t^3+2*t^2+4*t
3*t^4+t^3+2*t^2+4*t+1
3*t^4+t^3+2*t^2+3*t+1
3*t^4+t^3+3*t^2+4
3*t^4+t^3+t^2+t+4
3*t^4+t^3+t^2+t
3*t^4+t^3+t^2+2*t+4
3*t^4+t^3+t^2
3*t^4+t^3+3*t^2+3*t
3*t^4+t^3+3*t^2+3*t+1
3*t^4+t^3+3*t^2+4*t
3*t^4+t^3+3*t^2+2*t+1
3*t^4+2*t^2+t
3*t^4+t^3+2*t^2+3*t+2
3*t^4+t^3+2*t^2+2*t+2
3*t^4+t^3+t^2+1
3*t^4+t^3+t^2+4*t+1
3*t^4+t^3+3*t^2+2*t+2
3*t^4+t^3+3*t^2+t+2
3*t^4+2*t^2+2*t+4
3*t^4+t^3+2*t+4
3*t^4+t^3+2*t
3*t^4+t^3+3*t+4
3*t^4+t^3+t
3*t^4+2*t^2
3*t^4+t^3+4*t^2+3*t
3*t^4+t^3+4*t^2+t+1
3*t^4+t^3+4*t^2+t+2
3*t^4+t^3+4*t^2+2*t+1
3*t^4+t^3+4*t^2+2
3*t^4+4*t^3+2*t^2+t
3*t^4+4*t^3+3*t^2+4*t+1
3*t^4+4*t^3+3*t^2+4*t+2
3*t^4+4*t^3+3*t^2+1
3*t^4+4*t^3+3*t^2+3*t+2
3*t^4+4*t^3+2*t^2+4*t+3
3*t^4+4*t^3+2*t^2+3*t+3
3*t^4+4*t^3+3*t^2+3*t+3
3*t^4+4*t^3+3*t^2+2*t+3
3*t^4+4*t^3+4*t^2+2*t+2
3*t^4+4*t^3+4*t^2+2*t+3
3*t^4+4*t^3+4*t^2+3*t+2
3*t^4+4*t^3+4*t^2+t+3
3*t^4+4*t^2+t+3
2*t^4+t^2+3*t
2*t^4+3*t^3+2*t^2+3*t
2*t^4+3*t^3+2*t^2+3*t+1
2*t^4+3*t^3+2*t^2+4*t
2*t^4+3*t^3+2*t^2+2*t+1
2*t^4+3*t^3+2*t^2+2*t+2
2*t^4+3*t^3+2*t^2+t+2
2*t^4+3*t^3+3*t^2+3*t
2*t^4+3*t^3+t^2+4*t
2*t^4+3*t^3+t^2+4*t+1
2*t^4+3*t^3+t^2
2*t^4+3*t^3+t^2+3*t+1
2*t^4+3*t^3+3*t^2+t+1
2*t^4+3*t^3+3*t^2+t+2
2*t^4+3*t^3+3*t^2+2*t+1
2*t^4+3*t^3+3*t^2+2
2*t^4+3*t^3+3*t^2+3
2*t^4+3*t^3+2*t^2+t+3
2*t^4+3*t^3+t^2+3*t+2
2*t^4+3*t^3+3*t^2+4*t+3
2*t^4+3*t^3+3*t^2+4*t+4
2*t^4+3*t^3+3*t^2+3*t+4
2*t^4+3*t^3+4*t^2+2
2*t^4+3*t^3+2*t^2+3
2*t^4+3*t^3+4*t^2+3*t+3
2*t^4+3*t^3+4*t^2+3*t+4
2*t^4+3*t^3+4*t^2+4*t+3
2*t^4+3*t^3+4*t^2+2*t+4
2*t^4+3*t^3
2*t^4+3*t^3+4*t^2+4*t+2
2*t^4+3*t^3+t^2+2*t+2
2*t^4+3*t^3+4*t+2
2*t^4+3*t^3+2*t+3
2*t^4+3*t^3+2*t+4
2*t^4+3*t^3+3*t+3
2*t^4+3*t^3+t+4
2*t^4+4*t^3+2*t^2+3*t
2*t^4+2*t^3+3*t^2+3*t
2*t^4+2*t^3+3*t^2+3*t+1
2*t^4+2*t^3+3*t^2+4*t
2*t^4+2*t^3+3*t^2+2*t+1
2*t^4+3*t^3+4*t^2+t+1
2*t^4+2*t^3+3*t^2+2*t+2
2*t^4+2*t^3+3*t^2+t+2
2*t^4+2*t^3+4*t^2+3*t
2*t^4+2*t^3+2*t^2+4*t
2*t^4+2*t^3+2*t^2+4*t+1
2*t^4+2*t^3+2*t^2
2*t^4+2*t^3+2*t^2+3*t+1
2*t^4+2*t^3+4*t^2+t+1
2*t^4+2*t^3+4*t^2+t+2
2*t^4+2*t^3+4*t^2+2*t+1
2*t^4+2*t^3+4*t^2+2
2*t^4+4*t^3+4*t
2*t^4+4*t^3+4*t+1
2*t^4+4*t^3
2*t^4+4*t^3+3*t+1
2*t^4+4*t^3+3*t+2
2*t^4+4*t^3+2*t+2
2*t^4+4*t^3+t^2+4*t
2*t^4+4*t^3+4*t^2
2*t^4+4*t^3+4*t^2+1
2*t^4+4*t^3+4*t^2+t
```

```
3*t^4+3*t^3+2*t^2+4*t+4
3*t^4+3*t^3+2*t^2+2*t
3*t^4+3*t^3+2*t^2+2*t+1
3*t^4+3*t^3+2*t^2+t+1
3*t^4+3*t^3+3*t^2+3*t+4
3*t^4+3*t^3+t^2+4*t+4
3*t^4+3*t^3+t^2+4*t
3*t^4+3*t^3+t^2+4
3*t^4+3*t^3+t^2+3*t
3*t^4+3*t^3+3*t^2+t
3*t^4+3*t^3+3*t^2+t+1
3*t^4+3*t^3+3*t^2+2*t
3*t^4+3*t^3+3*t^2+1
3*t^4+3*t^3+3*t^2+2
3*t^4+3*t^3+2*t^2+t+2
3*t^4+3*t^3+t^2+3*t+1
3*t^4+3*t^3+3*t^2+4*t+2
3*t^4+3*t^3+3*t^2+4*t+3
3*t^4+3*t^3+3*t^2+3*t+3
3*t^4+3*t^3+4*t^2+1
3*t^4+3*t^3+2*t^2+2
3*t^4+3*t^3+4*t^2+3*t+2
3*t^4+3*t^3+4*t^2+3*t+3
3*t^4+3*t^3+4*t^2+4*t+2
3*t^4+3*t^3+4*t^2+2*t+3
3*t^4+3*t^3+4
3*t^4+3*t^3+4*t^2+4*t+1
3*t^4+3*t^3+t^2+2*t+1
3*t^4+3*t^3+4*t+1
3*t^4+3*t^3+2*t+2
3*t^4+3*t^3+2*t+3
3*t^4+3*t^3+3*t+2
3*t^4+3*t^3+t+3
3*t^4+4*t^3+2*t^2+3*t+4
3*t^4+2*t^3+3*t^2+3*t+4
3*t^4+2*t^3+3*t^2+3*t
3*t^4+2*t^3+3*t^2+4*t+4
3*t^4+2*t^3+3*t^2+2*t
3*t^4+3*t^3+4*t^2+t
3*t^4+2*t^3+3*t^2+2*t+1
3*t^4+2*t^3+3*t^2+t+1
3*t^4+2*t^3+4*t^2+3*t+4
3*t^4+2*t^3+2*t^2+4*t+4
3*t^4+2*t^3+2*t^2+4*t
3*t^4+2*t^3+2*t^2+4
3*t^4+2*t^3+2*t^2+3*t
3*t^4+2*t^3+4*t^2+t
3*t^4+2*t^3+4*t^2+t+1
3*t^4+2*t^3+4*t^2+2*t
3*t^4+2*t^3+4*t^2+1
3*t^4+4*t^3+4*t+4
3*t^4+4*t^3+4*t
3*t^4+4*t^3+4
3*t^4+4*t^3+3*t
3*t^4+4*t^3+3*t+1
3*t^4+4*t^3+2*t+1
3*t^4+4*t^3+t^2+4*t+4
3*t^4+4*t^3+4*t^2+4
3*t^4+4*t^3+4*t^2
3*t^4+4*t^3+4*t^2+t+4
3*t^4+4*t^3+4*t^2+4*t
3*t^4+4*t^3+t^2+2*t
3*t^4+4*t^3+t^2+2*t+1
3*t^4+3*t^2+3*t+2
3*t^4+3*t+4
3*t^4+4*t^2+3
3*t^4+t^2+3*t+4
3*t^4+t^2+2*t+4
3*t^4+3*t^2+2*t+2
3*t^4+t^3+2*t^2+2*t+3
3*t^4+t^3+2*t^2+t+3
3*t^4+t^3+t^2+4*t+2
3*t^4+t^3+t^2+3*t+2
3*t^4+t^3+3*t^2+t+3
3*t^4+t^3+3*t^2+3
3*t^4+t^3+t+1
3*t^4+t^3+1
3*t^4+2*t^2+4*t+1
3*t^4+t^3+4*t^2+3
3*t^4+t^3+4*t^2+4*t+3
3*t^4+4*t^3+2*t^2+2*t+4
3*t^4+4*t^3+t^2+3
3*t^4+4*t^3+t^2+4*t+3
3*t^4+4*t^3+t^2+3*t+4
3*t^4+4*t^3+t^2+2*t
3*t^4+t^2+t
3*t^4+3*t^2+t+3
3*t^4+3*t^2+2*t+3
3*t^4+t^3+4*t+2
3*t^4+t^3+4*t+3
3*t^4+t^3+2
3*t^4+t^3+3*t+3
3*t^4+4*t^3+3
3*t^4+4*t^3+t+3
3*t^4+t^3+3*t^2+4*t+4
3*t^4+t^3+t^2+2*t+3
3*t^4+t^3+t^2+3*t+3
3*t^4+4*t^3+t^2+3*t+4
3*t^4+2*t^3+t+4
3*t^4+2*t^3+2*t+4
3*t^4+2*t^3+t^2+t+4
3*t^4+2*t^3+4*t^2+2*t+4
3*t^4+2*t^3+t^2+4*t
3*t^4+2*t^3+2*t^2+2*t+1
3*t^4+2*t^3+3*t^2+2
3*t^4+2*t^3+3*t^2+3
3*t^4+2*t^3+3*t^2+t+2
3*t^4+2*t^3+3*t^2+4*t+3
3*t^4+3*t^3+3*t+1
3*t^4+3*t^3+t^2+t+2
3*t^4+3*t^3+t^2+t+3
3*t^4+3*t^3+t^2+2*t+2
3*t^4+3*t^3+t^2+3
3*t^4+3*t^3+2*t^2+4*t+3
3*t^4+3*t^3+2*t^2+3
3*t^4+3*t^3+4*t^2+2*t+4
3*t^4+3*t^3+4*t^2+t+4
3*t^4+3*t^3+3*t^2+2*t+4
3*t^4+2*t^3+t^2+4
3*t^4+2*t^3+4*t^2+3*t+3
3*t^4+2*t^3+4*t^2+4*t+3
3*t^4+3*t^3+4*t^2
2*t^4+4*t^3+4*t^2+4*t+1
2*t^4+4*t^3+t^2+2*t+1
2*t^4+4*t^3+t^2+2*t+2
2*t^4+4*t^3+t^2+3*t+1
2*t^4+4*t^3+t^2+t+2
2*t^4+3*t^3+1
2*t^4+4*t^3+2*t+3
2*t^4+4*t^3+t+3
2*t^4+4*t^3+4*t^2+4*t+2
2*t^4+4*t^3+4*t^2+3*t+2
2*t^4+4*t^3+t^2+t+3
2*t^4+4*t^3+t^2+3
2*t^4+3*t^3+t
2*t^4+4*t^3+3*t^2+t
2*t^4+4*t^3+3*t^2+t+1
2*t^4+4*t^3+3*t^2+2*t
2*t^4+4*t^3+3*t^2+1
2*t^4+3*t^3+4*t+1
2*t^4+4*t^3+2*t^2+2*t+1
2*t^4+4*t^3+2*t^2+2
2*t^4+4*t^3+2*t^2+3
2*t^4+4*t^3+2*t^2+t+2
2*t^4+4*t^3+2*t^2+4*t+3
2*t^4+2*t^3+1
2*t^4+2*t^3+2
2*t^4+2*t^3+t+1
2*t^4+2*t^3+4*t+2
2*t^4+2*t^3+4*t+3
2*t^4+2*t^3+3*t+3
2*t^4+2*t^3+t^2+1
2*t^4+2*t^3+t^2+3*t+2
2*t^4+2*t^3+t^2+3*t+3
2*t^4+2*t^3+t^2+4*t+2
2*t^4+2*t^3+t^2+2*t+3
2*t^4+2*t^3+3*t+4
2*t^4+2*t^3+2*t+4
2*t^4+2*t^3+4*t^2+3
2*t^4+2*t^3+4*t^2+4*t+3
2*t^4+2*t^3+t^2+2*t+4
2*t^4+2*t^3+t^2+t+4
2*t^4+2*t^3+2*t^2+3*t+2
2*t^4+2*t^3+2*t^2+t+3
2*t^4+2*t^3+2*t^2+t+4
2*t^4+2*t^3+2*t^2+2*t+3
2*t^4+2*t^3+2*t^2+4
2*t^4+3*t^3+2*t^2+4
2*t^4+3*t^3+t^2+2*t+3
2*t^4+3*t^3+3*t^2+2*t
2*t^4+3*t^3+2*t^2+4*t+4
2*t^4+3*t^3+4*t^2+2*t
2*t^4+3*t^3+4*t^2+t
2*t^4+3*t^3+t^2+t+3
2*t^4+2*t^3+3*t^2+t+3
2*t^4+2*t^3+3*t^2+3
2*t^4+2*t^3+2*t^2+2*t+2
2*t^4+4*t^3+t+4
2*t^4+4*t^3+4
2*t^4+4*t^3+4*t^2+3*t+3
2*t^4+4*t^3+4*t^2+2*t+3
2*t^4+4*t^3+t^2+4
2*t^4+4*t^3+t^2+4*t+4
2*t^4+4*t^3+3*t^2+2
2*t^4+4*t^3+3*t^2+4*t+2
```

3 PRIEDAS. LYGČIŲ SISTEMOS IR JŲ GRIOBNERIO BAZĖS

1. **Pavyzdys**: $p = 2$, $m = 1$, $n = 2$, $t = 1$.

Gauname $s = 4$ lygčių sistemą su $k = 8$ kintamaisiais:

```
1*1^1*5^1+1*2^1*7^1+1 = 0
1*1^1*6^1+1*2^1*8^1 = 0
1*3^1*5^1+1*4^1*7^1 = 0
1*3^1*6^1+1*4^1*8^1+1 = 0
```

Kai naudojama leksikografinė rikiavimo tvarka, Griobnerio bazes sudaro 19 lygčių:

```
1*1^1*5^1+1*2^1*7^1+1 = 0
1*1^1*6^1+1*2^1*8^1 = 0
1*3^1*5^1+1*4^1*7^1 = 0
1*3^1*6^1+1*4^1*8^1+1 = 0
1*2^1*7^1+1*4^1*8^1+1 = 0
1*1^1*7^1+1*3^1*8^1 = 0
1*2^1*5^1+1*4^1*6^1 = 0
1*2^1*8^1+1*2^1+1*8^1+1 = 0
1*4^1*7^1+1*4^1+1*7^1+1 = 0
1*4^1*5^1+1*4^1*6^1*8^1+1*5^1+1*6^1*7^1+1*6^1 = 0
1*7^1*8^1+1*7^1+1*8^1+1 = 0
1*1^1*8^1+1*1^1+1*2^1*3^1+1*3^1*4^1*8^1+1*3^1 = 0
1*2^1*4^1+1*2^1+1*4^1+1 = 0
1*1^1*4^1+1*1^1+1*3^1*4^1*8^1+1*3^1*8^1 = 0
1*2^1*3^1+1*3^1*4^1*8^1+1*3^1 = 0
1*5^1*6^1*8^1+1*5^1*6^1+1*5^1*8^1+1*5^1 = 0
1*4^1*6^1+1*5^1*8^1+1*5^1+1*6^1*7^1+1*6^1 = 0
1*4^1*8^1+1*4^1+1*5^1*8^1+1*5^1 = 0
1*5^1*6^1*7^1+1*5^1*8^1+1*5^1 = 0
```

Kai naudojama laipsninė ir leksikografinė rikiavimo tvarka, Griobnerio bazes sudaro 20 lygčių:

```
1*1^1*5^1+1*2^1*7^1+1 = 0
1*1^1*6^1+1*2^1*8^1 = 0
1*3^1*5^1+1*4^1*7^1 = 0
1*3^1*6^1+1*4^1*8^1+1 = 0
1*2^1*7^1+1*4^1*8^1+1 = 0
1*1^1*7^1+1*3^1*8^1 = 0
1*2^1*5^1+1*4^1*6^1 = 0
1*2^1*8^1+1*2^1+1*8^1+1 = 0
```

```
1*4^1*7^1+1*4^1+1*7^1+1 = 0
1*7^1*8^1+1*7^1+1*8^1+1 = 0
1*4^1*5^1+1*4^1*6^1+1*5^1*8^1+1*6^1*7^1+1*6^1 = 0
1*2^1*4^1+1*2^1+1*4^1+1 = 0
1*1^1*4^1+1*1^1*8^1+1*2^1*3^1+1*3^1*8^1+1*3^1 = 0
1*1^1*8^1+1*1^1 = 0
1*5^1*6^1*7^1+1*5^1*6^1*8^1+1*5^1*6^1 = 0
1*4^1*6^1+1*5^1*8^1+1*6^1*7^1+1*5^1+1*6^1 = 0
1*2^1*3^1+1*3^1*8^1+1*4^1*8^1+1*3^1+1*8^1 = 0
1*4^1*8^1+1*5^1*8^1+1*4^1+1*5^1 = 0
1*5^1*8^1+1*6^1*7^1+1*6^1*8^1+1*4^1+1*5^1+1*6^1+1*8^1 = 0
1*3^1*4^1+1*3^1*8^1+1*4^1+1*8^1 = 0
```

Kai naudojama laipsninė ir atvirkštinė leksikografinė rikiavimo tvarka, Griobnerio bazes sudaro 19 lygčių:

```
1*1^1*5^1+1*2^1*7^1+1 = 0
1*1^1*6^1+1*2^1*8^1 = 0
1*3^1*5^1+1*4^1*7^1 = 0
1*3^1*6^1+1*4^1*8^1+1 = 0
1*2^1*7^1+1*4^1*8^1+1 = 0
1*4^1*5^1*8^1+1*4^1*6^1*8^1+1*6^1*7^1+1*5^1*8^1+1*6^1 = 0
1*1^1*4^1*8^1+1*3^1*4^1*8^1+1*2^1*3^1+1*1^1*4^1+1*3^1 = 0
1*2^1*8^1+1*2^1+1*8^1+1 = 0
1*3^1*4^1*8^1+1*1^1*7^1*8^1+1*2^1*3^1+1*1^1*7^1+1*1^1*8^1+1*1^1+1*3^1 = 0
1*2^1*4^1+1*2^1+1*4^1+1 = 0
1*4^1*7^1+1*4^1+1*7^1+1 = 0
1*7^1*8^1+1*7^1+1*8^1+1 = 0
1*4^1*6^1*8^1+1*6^1*7^1+1*6^1 = 0
1*1^1*2^1*3^1+1*1^1*3^1*4^1+1*1^1*3^1 = 0
1*5^1*6^1*7^1+1*5^1*6^1*8^1+1*5^1*6^1 = 0
```

2. **Pavyzdys**: $p = 2$, $m = 1$, $n = 2$, $t = 3$.

Gauname $s = 8$ lygčių sistemą su $k = 14$ kintamaisiais:

```
1*1^1*9^1+1*3^1*13^1+1 = 0
1*1^1*10^1+1*2^1*9^1+1*2^1*10^1+1*3^1*9^1+1*3^1*10^1+1*3^1*13^1+1*4^1*9^1+
+1*4^1*10^1 = 0
1*1^1*11^1+1*3^1*15^1 = 0
1*1^1*12^1+1*2^1*11^1+1*2^1*12^1+1*3^1*11^1+1*3^1*12^1+1*3^1*15^1+1*4^1*11^1+
+1*4^1*12^1 = 0
```

```
     1*5^1*9^1+1*7^1*13^1 = 0
     1*5^1*10^1+1*6^1*9^1+1*6^1*10^1+1*7^1*9^1+1*7^1*10^1+1*7^1*13^1+1*8^1*9^1+
+1*8^1*10^1+1 = 0
     1*5^1*11^1+1*7^1*15^1+1 = 0
     1*5^1*12^1+1*6^1*11^1+1*6^1*12^1+1*7^1*11^1+1*7^1*12^1+1*7^1*15^1+1*8^1*11^1+
+1*8^1*12^1+1 = 0
```

Kai naudojama leksikografinė rikiavimo tvarka, Griobnerio bazes sudaro 20 lygčių:

```
     1*5^1*9^1+1*7^1*13^1 = 0
     1*5^1*10^1+1*6^1*9^1+1*6^1*10^1+1*7^1*9^1+1*7^1*10^1+1*7^1*13^1+1*8^1*9^1+
+1*8^1*10^1+1 = 0
     1*5^1*12^1+1*6^1*11^1+1*6^1*12^1+1*7^1*11^1+1*7^1*12^1+1*7^1*15^1+1*8^1*11^1+
+1*8^1*12^1+1 = 0
     1*11^1+1*12^1 = 0
     1*1^1+1*2^1+1*3^1+1*4^1+1*5^1+1*6^1+1*7^1+1*8^1 = 0
     1*3^1+1*4^1*7^1*13^1+1*4^1*9^1*10^1*12^1+1*4^1*9^1*10^1*13^1*15^1+
+1*4^1*9^1*10^1*13^1+1*4^1*9^1*10^1*15^1+1*4^1*9^1*10^1+1*4^1*9^1*12^1+
+1*4^1*10^1*12^1*13^1*15^1+1*4^1*10^1*12^1+1*4^1*10^1*15^1+1*4^1*10^1+
+1*4^1*12^1*13^1*15^1+1*4^1*12^1*13^1+1*4^1*12^1*15^1+1*4^1*13^1+1*5^1*7^1+
+1*5^1*15^1+1*6^1*10^1*13^1*15^1+1*6^1*10^1*15^1+1*6^1*10^1+1*6^1*13^1+
+1*8^1*10^1*13^1*15^1+1*8^1*10^1*15^1+1*8^1*10^1+1*8^1*13^1+1*9^1*10^1*13^1*15^1+
+1*9^1*10^1*13^1+1*9^1*10^1*15^1+1*9^1*10^1+1*9^1*13^1*15^1+1*9^1*13^1+1*9^1*15^1+
+1*9^1+1*10^1*12^1*13^1*15^1+1*10^1*12^1*13^1+1*10^1*12^1+1*12^1*13^1*15^1+1*12^1+
+1*13^1+1*15^1+1 = 0
     1*5^1*15^1+1*5^1+1*7^1*13^1+1*7^1+1*9^1*10^1*13^1*15^1+1*9^1*10^1*13^1+
+1*9^1*10^1*15^1+1*9^1*10^1+1*10^1*12^1*13^1+1*10^1*12^1+1*10^1*13^1*15^1+
+1*10^1*13^1+1*10^1*15^1+1*10^1+1*12^1*13^1*15^1+1*12^1*13^1+1*12^1*15^1+1*12^1+
+1*13^1+1*15^1 = 0
     1*7^1+1*9^1*10^1*13^1+1*9^1*13^1*15^1+1*9^1*13^1+1*10^1*12^1*13^1*15^1+
+1*10^1*12^1*15^1+1*10^1*13^1+1*10^1*15^1+1*10^1+1*12^1*13^1*15^1+1*12^1*13^1+1= 0
     1*12^1*15^1+1*12^1+1*15^1+1 = 0
     1*9^1*15^1+1*9^1+1*10^1*15^1+1*10^1+1*15^1+1 = 0
     1*9^1*12^1+1*10^1*12^1+1*12^1+1*13^1*15^1+1*13^1+1*15^1+1 = 0
     1*10^1*12^1+1*10^1*15^1+1*10^1+1*12^1*13^1+1*15^1+1 = 0
     1*10^1*13^1+1*10^1*15^1+1*10^1+1*12^1*13^1+1*15^1+1 = 0
     1*13^1*15^1+1*13^1+1*15^1+1 = 0
     1*5^1*13^1+1*5^1+1*6^1+1*8^1+1*9^1*13^1+1*9^1+1*10^1*15^1+1*10^1+1*12^1*13^1+
+1*15^1 = 0
     1*6^1+1*8^1+1*9^1*10^1+1*9^1*13^1+1*9^1+1*10^1+1 = 0
     1*9^1*10^1+1*9^1*13^1+1*9^1+1*13^1+1 = 0
     1*9^1*13^1+1*9^1+1*10^1*15^1+1*12^1*13^1+1*13^1+1*15^1 = 0
```

```
1*2^1+1*4^1+1*10^1*15^1+1*12^1*13^1+1*12^1 = 0
1*10^1*15^1+1*12^1*13^1+1*15^1+1 = 0
```

Kai naudojama laipsninė ir leksikografinė rikiavimo tvarka, Griobnerio bazes sudaro 20 lygčių:

```
1*5^1*9^1+1*7^1*13^1 = 0
1*5^1*10^1+1*6^1*9^1+1*6^1*10^1+1*7^1*9^1+1*7^1*10^1+1*7^1*13^1+1*8^1*9^1+
+1*8^1*10^1+1 = 0
1*5^1*12^1+1*6^1*11^1+1*6^1*12^1+1*7^1*11^1+1*7^1*12^1+1*7^1*15^1+1*8^1*11^1+
+1*8^1*12^1+1 = 0
1*11^1+1*12^1 = 0
1*1^1+1*2^1+1*3^1+1*4^1+1*5^1+1*6^1+1*7^1+1*8^1 = 0
1*10^1*12^1+1*7^1+1 = 0
1*9^1*12^1+1*12^1*15^1+1*7^1+1*15^1 = 0
1*9^1*15^1+1*12^1*13^1+1*12^1*15^1+1*7^1+1*9^1+1*12^1+1*15^1 = 0
1*5^1*15^1+1*6^1*15^1+1*8^1*15^1+1*12^1*13^1+1*5^1+1*6^1+1*7^1+1*8^1+1*15^1=0
1*10^1*15^1+1*12^1*13^1+1*13^1*15^1+1*7^1+1*10^1+1*13^1+1 = 0
1*12^1*15^1+1*13^1*15^1+1*12^1+1*13^1 = 0
1*13^1*15^1+1*13^1+1*15^1+1 = 0
1*3^1+1*12^1 = 0
1*9^1*13^1+1*10^1*13^1+1*9^1+1*10^1+1*13^1+1 = 0
1*5^1*13^1+1*5^1 = 0
1*10^1*13^1+1*10^1 = 0
1*9^1*10^1+1*7^1+1*10^1+1 = 0
1*6^1+1*7^1+1*8^1+1*13^1+1 = 0
1*7^1+1*10^1+1 = 0
1*2^1+1*4^1+1*12^1+1*15^1+1 = 0
```

Kai naudojama laipsninė ir atvirkštinė leksikografinė rikiavimo tvarka, Griobnerio bazes sudaro 23 lygčių:

```
1*2^1*9^1+1*3^1*9^1+1*4^1*9^1+1*1^1*10^1+1*2^1*10^1+1*3^1*10^1+1*4^1*10^1+
+1*3^1*13^1 = 0
1*5^1*9^1+1*7^1*13^1 = 0
1*6^1*9^1+1*7^1*9^1+1*8^1*9^1+1*5^1*10^1+1*6^1*10^1+1*7^1*10^1+1*8^1*10^1+
+1*7^1*13^1+1 = 0
1*11^1+1*12^1 = 0
1*1^1+1*2^1+1*3^1+1*4^1+1*5^1+1*6^1+1*7^1+1*8^1 = 0
1*3^1*6^1*13^1+1*2^1*7^1*13^1+1*4^1*7^1*13^1+1*5^1*7^1*13^1+1*6^1*7^1*13^1+
+1*3^1*8^1*13^1+1*7^1*8^1*13^1+1*2^1*10^1*13^1+1*3^1*10^1*13^1+1*4^1*10^1*13^1+
+1*5^1*10^1*13^1+1*5^1*7^1+1*2^1*13^1+1*3^1*13^1+1*4^1*13^1+1*5^1*13^1+1*7^1*13^1+
+1*10^1*13^1+1*5^1 = 0
```

```
    1*6^1*7^1*13^1+1*7^1*8^1*13^1+1*2^1*10^1*13^1+1*3^1*10^1*13^1+
+1*4^1*10^1*13^1+1*6^1*10^1*13^1+1*7^1*10^1*13^1+1*8^1*10^1*13^1+1*2^1*10^1+
+1*3^1*10^1+1*4^1*10^1+1*6^1*10^1+1*7^1*10^1+1*8^1*10^1+1*2^1*13^1+1*3^1*13^1+
+1*4^1*13^1+1*5^1*13^1+1*7^1*13^1+1*2^1+1*3^1+1*4^1+1*5^1 = 0
    1*2^1*6^1*7^1+1*3^1*6^1*7^1+1*4^1*6^1*7^1+1*5^1*6^1*7^1+1*2^1*7^1*8^1+
+1*3^1*7^1*8^1+1*4^1*7^1*8^1+1*5^1*7^1*8^1+1*2^1*3^1*10^1+1*3^1*4^1*10^1+
+1*3^1*5^1*10^1+1*2^1*6^1*10^1+1*4^1*6^1*10^1+1*3^1*7^1*10^1+1*2^1*8^1*10^1+
+1*4^1*8^1*10^1+1*2^1*3^1*13^1+1*3^1*4^1*13^1+1*2^1*5^1*13^1+1*4^1*5^1*13^1+
+1*2^1*6^1*13^1+1*4^1*6^1*13^1+1*5^1*6^1*13^1+1*3^1*7^1*13^1+1*2^1*8^1*13^1+
+1*4^1*8^1*13^1+1*5^1*8^1*13^1+1*2^1*10^1*13^1+1*4^1*10^1*13^1+1*5^1*10^1*13^1+
+1*2^1*3^1+1*3^1*4^1+1*2^1*5^1+1*3^1*5^1+1*4^1*5^1+1*2^1*7^1+1*3^1*7^1+1*4^1*7^1+
+1*6^1*10^1+1*7^1*10^1+1*8^1*10^1+1*3^1*13^1+1*10^1*13^1+1*3^1 = 0
    1*9^1*10^1 = 0
    1*5^1*10^1+1*6^1*10^1+1*7^1*10^1+1*8^1*10^1+1*10^1 = 0
    1*5^1*7^1+1*6^1*10^1+1*7^1*10^1+1*8^1*10^1+1*10^1*13^1+1*5^1 = 0
    1*2^1*5^1+1*3^1*5^1+1*4^1*5^1+1*2^1*6^1+1*3^1*6^1+1*4^1*6^1+1*2^1*7^1+
+1*3^1*7^1+1*4^1*7^1+1*2^1*8^1+1*3^1*8^1+1*4^1*8^1+1*3^1*10^1+1*7^1*10^1+
+1*2^1*13^1+1*3^1*13^1+1*4^1*13^1+1*10^1*13^1 = 0
    1*10^1*13^1+1*10^1 = 0
    1*3^1*7^1*13^1+1*3^1*5^1+1*2^1*10^1+1*3^1*10^1+1*4^1*10^1+1*6^1*10^1+
+1*8^1*10^1+1*2^1*13^1+1*4^1*13^1+1*5^1*13^1+1*5^1 = 0
    1*3^1*10^1+1*2^1*13^1+1*3^1*13^1+1*4^1*13^1+1*2^1+1*3^1+1*4^1+1*10^1 = 0
    1*2^1*3^1+1*3^1*4^1+1*5^1*6^1+1*5^1*8^1+1*6^1*10^1+1*8^1*10^1+1*2^1*13^1+
+1*3^1*13^1+1*4^1*13^1+1*5^1*13^1+1*3^1+1*10^1 = 0
    1*5^1*6^1+1*5^1*8^1+1*6^1*10^1+1*8^1*10^1+1*5^1+1*10^1 = 0
    1*2^1*10^1+1*4^1*10^1+1*7^1*10^1+1*3^1*13^1+1*2^1+1*3^1+1*4^1 = 0
    1*3^1*5^1+1*2^1*7^1+1*3^1*7^1+1*4^1*7^1+1*6^1*10^1+1*7^1*10^1+1*8^1*10^1+
+1*2^1*13^1+1*3^1*13^1+1*4^1*13^1+1*5^1*13^1+1*2^1+1*3^1+1*4^1+1*5^1+1*10^1 = 0
    1*7^1*10^1 = 0
    1*2^1*13^1+1*3^1*13^1+1*4^1*13^1+1*2^1+1*3^1+1*4^1 = 0
    1*5^1*13^1+1*5^1 = 0
    1*3^1*9^1*13^1+1*3^1*13^1+1*10^1 = 0
```

3. **Pavyzdys**: $p = 2$, $m = 1$, $n = 2$, $t = 2$.

Gauname $s = 8$ lygčių sistemą su $k = 16$ kintamaisiais:

```
    1*1^1*9^1+1*3^1*9^1+1 = 0
    1*1^1*10^1+1*2^1*9^1+1*2^1*10^1+1*3^1*9^1+1*3^1*13^1+1*3^1*14^1+1*4^1*13^1+
+1*4^1*14^1 = 0
    1*1^1*11^1+1*3^1*11^1 = 0
    1*1^1*12^1+1*2^1*11^1+1*2^1*12^1+1*3^1*11^1+1*3^1*15^1+1*3^1*16^1+1*4^1*15^1+
```

```
+1*4^1*16^1 = 0
     1*5^1*9^1+1*7^1*9^1+1 = 0
     1*5^1*10^1+1*6^1*9^1+1*6^1*10^1+1*7^1*9^1+1*7^1*13^1+1*7^1*14^1+1*8^1*13^1+
+1*8^1*14^1+1 = 0
     1*5^1*11^1+1*7^1*11^1 = 0
     1*5^1*12^1+1*6^1*11^1+1*6^1*12^1+1*7^1*11^1+1*7^1*15^1+1*7^1*16^1+1*8^1*15^1+
+1*8^1*16^1+1 = 0
```

Kai naudojama leksikografinė rikiavimo tvarka, Griobnerio bazes sudaro 29 lygčių:

```
     1*5^1*9^1+1*7^1*9^1+1 = 0
     1*5^1*10^1+1*6^1*9^1+1*6^1*10^1+1*7^1*9^1+1*7^1*13^1+1*7^1*14^1+1*8^1*13^1+
+1*8^1*14^1+1 = 0
     1*5^1*12^1+1*6^1*11^1+1*6^1*12^1+1*7^1*11^1+1*7^1*15^1+1*7^1*16^1+1*8^1*15^1+
+1*8^1*16^1+1 = 0
     1*11^1 = 0
     1*1^1+1*3^1+1*5^1+1*7^1 = 0
     1*2^1*10^1+1*2^1+1*3^1*10^1+1*3^1*13^1+1*3^1*14^1+1*3^1+1*4^1*13^1+
+1*4^1*14^1+1*6^1*9^1+1*6^1*10^1+1*7^1*9^1+1*7^1*10^1+1*7^1*13^1+1*7^1*14^1+
+1*8^1*13^1+1*8^1*14^1+1 = 0
     1*2^1*12^1+1*3^1*12^1+1*3^1*15^1+1*3^1*16^1+1*4^1*15^1+1*4^1*16^1+1*6^1*12^1+
+1*7^1*12^1+1*7^1*15^1+1*7^1*16^1+1*8^1*15^1+1*8^1*16^1+1 = 0
     1*6^1*9^1+1*6^1+1*7^1*9^1+1*7^1 = 0
     1*2^1*5^1+1*2^1*7^1*13^1+1*2^1*7^1*14^1+1*2^1*7^1+1*2^1*8^1*13^1+
+1*2^1*8^1*14^1+1*2^1+1*3^1*5^1*13^1+1*3^1*5^1*14^1+1*3^1*5^1+1*3^1*6^1*13^1+
+1*3^1*6^1*14^1+1*3^1*7^1*13^1+1*3^1*7^1*14^1+1*3^1*7^1+1*3^1*8^1*13^1+
+1*3^1*8^1*14^1+1*3^1+1*4^1*5^1*13^1+1*4^1*5^1*14^1+1*4^1*6^1*13^1+1*4^1*6^1*14^1+
+1*5^1*7^1*13^1+1*5^1*7^1*14^1+1*5^1*8^1*13^1+1*5^1*8^1*14^1+1*5^1+1*6^1+
+1*7^1*8^1*13^1+1*7^1*8^1*14^1+1*7^1*13^1+1*7^1*14^1 = 0
     1*5^1*6^1+1*5^1*7^1+1*6^1*7^1+1*6^1 = 0
     1*7^1*9^1+1*7^1*10^1+1*8^1*9^1+1*8^1*10^1+1*9^1+1*10^1 = 0
     1*3^1*10^1+1*3^1*12^1+1*3^1+1*4^1*10^1+1*4^1*12^1+1*4^1+1*7^1*10^1+
+1*7^1*12^1+1*7^1+1*8^1*10^1+1*8^1*12^1+1*8^1+1*10^1+1*12^1+1 = 0
     1*2^1*7^1+1*2^1*8^1+1*2^1+1*3^1*6^1*15^1+1*3^1*6^1*16^1+1*3^1*6^1+
+1*3^1*7^1*13^1+1*3^1*7^1*14^1+1*3^1*8^1*13^1+1*3^1*8^1*14^1+1*3^1*8^1*15^1+
+1*3^1*8^1*16^1+1*3^1*8^1+1*3^1*13^1+1*3^1*14^1+1*3^1*15^1+1*3^1*16^1+1*3^1+
+1*4^1*6^1*15^1+1*4^1*6^1*16^1+1*4^1*6^1+1*4^1*7^1*13^1+1*4^1*7^1*14^1+1*4^1*7^1+
+1*4^1*8^1*13^1+1*4^1*8^1*14^1+1*4^1*8^1*15^1+1*4^1*8^1*16^1+1*4^1*13^1+
+1*4^1*14^1+1*4^1*15^1+1*4^1*16^1+1*6^1*7^1+1*6^1*8^1+1*6^1+1*7^1*8^1 = 0
     1*6^1*13^1+1*6^1*14^1+1*7^1*13^1+1*7^1*14^1 = 0
     1*6^1*7^1+1*6^1*8^1+1*6^1+1*7^1*8^1 = 0
     1*3^1*5^1+1*3^1*7^1*13^1+1*3^1*7^1*14^1+1*3^1*7^1*15^1+1*3^1*7^1*16^1+
```

+1*3^1*7^1+1*3^1*8^1*13^1+1*3^1*8^1*14^1+1*3^1*8^1*15^1+1*3^1*8^1*16^1+1*4^1*5^1+
+1*4^1*7^1*13^1+1*4^1*7^1*14^1+1*4^1*7^1*15^1+1*4^1*7^1*16^1+1*4^1*7^1+
+1*4^1*8^1*13^1+1*4^1*8^1*14^1+1*4^1*8^1*15^1+1*4^1*8^1*16^1+1*5^1*7^1+1*5^1*8^1+
+1*5^1+1*7^1*8^1 = 0

1*5^1*15^1+1*5^1*16^1+1*5^1+1*6^1*10^1*15^1+1*6^1*10^1*16^1+1*6^1*10^1+
+1*6^1*12^1*15^1+1*6^1*12^1*16^1+1*6^1*12^1+1*7^1*10^1*13^1+1*7^1*10^1*14^1+
+1*7^1*10^1+1*7^1*12^1*13^1+1*7^1*12^1*14^1+1*7^1*12^1+1*7^1*13^1*15^1+
+1*7^1*13^1*16^1+1*7^1*14^1*15^1+1*7^1*14^1*16^1+1*7^1+1*8^1*10^1*13^1+
+1*8^1*10^1*14^1+1*8^1*10^1*15^1+1*8^1*10^1*16^1+1*8^1*12^1*13^1+1*8^1*12^1*14^1+
+1*8^1*12^1*15^1+1*8^1*12^1*16^1+1*8^1*13^1*15^1+1*8^1*13^1*16^1+1*8^1*14^1*15^1+
+1*8^1*14^1*16^1+1*8^1*15^1+1*8^1*16^1+1*10^1*13^1+1*10^1*14^1+1*10^1*15^1+
+1*10^1*16^1+1*12^1*13^1+1*12^1*14^1+1*12^1*15^1+1*12^1*16^1+1*13^1+1*14^1+1*15^1+
+1*16^1 = 0

1*6^1*15^1+1*6^1*16^1+1*6^1+1*7^1*15^1+1*7^1*16^1+1*7^1 = 0

1*3^1*15^1+1*3^1*16^1+1*3^1+1*4^1*15^1+1*4^1*16^1+1*4^1+1*7^1*15^1+
+1*7^1*16^1+1*7^1+1*8^1*15^1+1*8^1*16^1+1*8^1+1*10^1*15^1+1*10^1*16^1+1*10^1+
+1*12^1*15^1+1*12^1*16^1+1*12^1+1*15^1+1*16^1+1 = 0

1*2^1*15^1+1*2^1*16^1+1*2^1+1*3^1*6^1+1*3^1*7^1+1*3^1*13^1+1*3^1*14^1+1*3^1+
+1*4^1*6^1+1*4^1*7^1+1*4^1*13^1+1*4^1*14^1+1*4^1*15^1+1*4^1*16^1+1*7^1*10^1*13^1+
+1*7^1*10^1*14^1+1*7^1*10^1*15^1+1*7^1*10^1*16^1+1*7^1*12^1*13^1+1*7^1*12^1*14^1+
+1*7^1*12^1*15^1+1*7^1*12^1*16^1+1*7^1*13^1+1*7^1*14^1+1*7^1*15^1+1*7^1*16^1+
+1*8^1*10^1*13^1+1*8^1*10^1*14^1+1*8^1*10^1*15^1+1*8^1*10^1*16^1+1*8^1*12^1*13^1+
+1*8^1*12^1*14^1+1*8^1*12^1*15^1+1*8^1*12^1*16^1+1*8^1*13^1+1*8^1*14^1+1*8^1*15^1+
+1*8^1*16^1+1*10^1*13^1*15^1+1*10^1*13^1*16^1+1*10^1*14^1*15^1+1*10^1*14^1*16^1+
+1*10^1*15^1+1*10^1*16^1+1*12^1*13^1*15^1+1*12^1*13^1*16^1+1*12^1*14^1*15^1+
+1*12^1*14^1*16^1+1*12^1*15^1+1*12^1*16^1+1*13^1*15^1+1*13^1*16^1+1*14^1*15^1+
+1*14^1*16^1+1*15^1+1*16^1 = 0

1*3^1*6^1+1*3^1*7^1+1*3^1*13^1+1*3^1*14^1+1*3^1+1*4^1*6^1+1*4^1*7^1+
+1*4^1*13^1+1*4^1*14^1+1*4^1+1*7^1*10^1*13^1+1*7^1*10^1*14^1+1*7^1*10^1*15^1+
+1*7^1*10^1*16^1+1*7^1*12^1*13^1+1*7^1*12^1*14^1+1*7^1*12^1*15^1+1*7^1*12^1*16^1+
+1*7^1*13^1+1*7^1*14^1+1*7^1+1*8^1*10^1*13^1+1*8^1*10^1*14^1+1*8^1*10^1*15^1+
+1*8^1*10^1*16^1+1*8^1*12^1*13^1+1*8^1*12^1*14^1+1*8^1*12^1*15^1+1*8^1*12^1*16^1+
+1*8^1*13^1+1*8^1*14^1+1*8^1+1*9^1*13^1*15^1+1*9^1*13^1*16^1+1*9^1*13^1+
+1*9^1*14^1*15^1+1*9^1*14^1*16^1+1*9^1*14^1+1*10^1*13^1*15^1+1*10^1*13^1*16^1+
+1*10^1*14^1*15^1+1*10^1*14^1*16^1+1*10^1+1*12^1*13^1+1*12^1*14^1+1*12^1+1*13^1+
+1*14^1+1 = 0

1*3^1*13^1+1*3^1*14^1+1*4^1*13^1+1*4^1*14^1+1*7^1*15^1+1*7^1*16^1+1*8^1*15^1+
+1*8^1*16^1+1*9^1*10^1*15^1+1*9^1*10^1*16^1+1*9^1*10^1+1*9^1*12^1*15^1+
+1*9^1*12^1*16^1+1*9^1*12^1+1*9^1*15^1+1*9^1*16^1+1*9^1+1*10^1*15^1+1*10^1*16^1+
+1*10^1+1*12^1*13^1*15^1+1*12^1*13^1*16^1+1*12^1*13^1+1*12^1*14^1*15^1+
+1*12^1*14^1*16^1+1*12^1*14^1+1*12^1*15^1+1*12^1*16^1+1*12^1+1*13^1*15^1+
+1*13^1*16^1+1*13^1+1*14^1*15^1+1*14^1*16^1+1*14^1+1*15^1+1*16^1 = 0

```
    1*3^1*7^1+1*3^1*8^1+1*3^1+1*4^1*7^1+1*4^1*8^1+1*4^1+1*7^1*15^1+1*7^1*16^1+
+1*8^1*15^1+1*8^1*16^1+1*10^1*15^1+1*10^1*16^1+1*10^1+1*15^1+1*16^1 = 0
    1*7^1*13^1+1*7^1*14^1+1*7^1*15^1+1*7^1*16^1+1*8^1*13^1+1*8^1*14^1+1*8^1*15^1+
+1*8^1*16^1+1*10^1*15^1+1*10^1*16^1+1*10^1+1*12^1*13^1*15^1+1*12^1*13^1*16^1+
+1*12^1*13^1+1*12^1*14^1*15^1+1*12^1*14^1*16^1+1*12^1*14^1+1*12^1*15^1+
+1*12^1*16^1+1*12^1+1*13^1*15^1+1*13^1*16^1+1*14^1*15^1+1*14^1*16^1+1 = 0
    1*13^1*15^1+1*13^1*16^1+1*13^1+1*14^1*15^1+1*14^1*16^1+1*14^1+1*15^1+1*16^1+
+1 = 0
    1*7^1*15^1+1*7^1*16^1+1*7^1+1*8^1*15^1+1*8^1*16^1+1*8^1+1*10^1*15^1+
+1*10^1*16^1+1*10^1+1*15^1+1*16^1+1 = 0
    1*12^1*15^1+1*12^1*16^1+1*12^1+1*15^1+1*16^1+1 = 0
    1*9^1*15^1+1*9^1*16^1+1*9^1+1*15^1+1*16^1+1 = 0
    1*3^1*9^1+1*3^1+1*4^1*9^1+1*4^1+1*7^1*10^1+1*7^1+1*8^1*10^1+1*8^1+1*10^1+1= 0
```

Kai naudojama laipsninė ir leksikografinė rikiavimo tvarka, Griobnerio bazes sudaro 28 lygčių:

```
    1*5^1*9^1+1*7^1*9^1+1 = 0
    1*5^1*10^1+1*6^1*9^1+1*6^1*10^1+1*7^1*9^1+1*7^1*13^1+1*7^1*14^1+1*8^1*13^1+
+1*8^1*14^1+1 = 0
    1*5^1*12^1+1*6^1*11^1+1*6^1*12^1+1*7^1*11^1+1*7^1*15^1+1*7^1*16^1+1*8^1*15^1+
+1*8^1*16^1+1 = 0
    1*11^1 = 0
    1*1^1+1*3^1+1*5^1+1*7^1 = 0
    1*2^1*10^1+1*3^1*10^1+1*3^1*13^1+1*3^1*14^1+1*4^1*13^1+1*4^1*14^1+1*6^1*9^1+
+1*6^1*10^1+1*7^1*9^1+1*7^1*10^1+1*7^1*13^1+1*7^1*14^1+1*8^1*13^1+1*8^1*14^1+
+1*2^1+1*3^1+1 = 0
    1*2^1*12^1+1*3^1*12^1+1*3^1*15^1+1*3^1*16^1+1*4^1*15^1+1*4^1*16^1+1*6^1*12^1+
+1*7^1*12^1+1*7^1*15^1+1*7^1*16^1+1*8^1*15^1+1*8^1*16^1+1 = 0
    1*7^1*9^1+1*7^1*10^1+1*8^1*9^1+1*8^1*10^1+1*9^1+1*10^1 = 0
    1*3^1*10^1+1*3^1*12^1+1*4^1*10^1+1*4^1*12^1+1*7^1*10^1+1*7^1*12^1+1*8^1*10^1+
+1*8^1*12^1+1*3^1+1*4^1+1*7^1+1*8^1+1*10^1+1*12^1+1 = 0
    1*2^1*7^1+1*2^1*8^1+1*3^1*5^1+1*3^1*8^1+1*3^1*13^1+1*3^1*14^1+1*3^1*15^1+
+1*3^1*16^1+1*4^1*5^1+1*4^1*7^1+1*4^1*13^1+1*4^1*14^1+1*4^1*15^1+1*4^1*16^1+
+1*5^1*7^1+1*5^1*8^1+1*7^1*8^1+1*2^1+1*3^1+1*5^1 = 0
    1*3^1*5^1+1*3^1*7^1+1*3^1*15^1+1*3^1*16^1+1*4^1*5^1+1*4^1*7^1+1*4^1*15^1+
+1*4^1*16^1+1*5^1*7^1+1*5^1*8^1+1*6^1*10^1+1*7^1*8^1+1*7^1*10^1+1*7^1*13^1+
+1*7^1*14^1+1*7^1*15^1+1*7^1*16^1+1*8^1*13^1+1*8^1*14^1+1*8^1*15^1+1*8^1*16^1+
+1*9^1*13^1+1*9^1*14^1+1*9^1*15^1+1*9^1*16^1+1*10^1*13^1+1*10^1*14^1+1*10^1*15^1+
+1*10^1*16^1+1*5^1+1*6^1+1*8^1+1*13^1+1*14^1+1*15^1+1*16^1+1 = 0
    1*2^1*15^1+1*2^1*16^1+1*3^1*6^1+1*3^1*7^1+1*4^1*6^1+1*4^1*7^1+1*4^1*15^1+
+1*4^1*16^1+1*5^1*15^1+1*5^1*16^1+1*7^1*13^1+1*7^1*14^1+1*8^1*13^1+1*8^1*14^1+
+1*8^1*15^1+1*8^1*16^1+1*2^1+1*3^1+1*5^1+1*8^1+1*13^1+1*14^1+1*15^1+1*16^1+1 = 0
```

```
    1*3^1*13^1+1*3^1*14^1+1*4^1*13^1+1*4^1*14^1+1*7^1*15^1+1*7^1*16^1+1*8^1*15^1+
+1*8^1*16^1+1 = 0
    1*2^1*5^1+1*2^1*8^1+1*2^1*13^1+1*2^1*14^1+1*3^1*7^1+1*3^1*8^1+1*3^1*15^1+
+1*3^1*16^1+1*4^1*5^1+1*4^1*7^1+1*4^1*13^1+1*4^1*14^1+1*4^1*15^1+1*4^1*16^1+
+1*5^1*7^1+1*5^1*8^1+1*5^1*13^1+1*5^1*14^1+1*5^1*15^1+1*5^1*16^1+1*7^1*8^1+
+1*7^1*13^1+1*7^1*14^1+1*7^1*15^1+1*7^1*16^1+1*5^1+1*6^1+1*7^1 = 0
    1*3^1*15^1+1*3^1*16^1+1*4^1*15^1+1*4^1*16^1+1*7^1*13^1+1*7^1*14^1+1*8^1*13^1+
+1*8^1*14^1+1*3^1+1*4^1+1*7^1+1*8^1+1*13^1+1*14^1+1*15^1+1*16^1 = 0
    1*7^1*13^1+1*7^1*14^1+1*8^1*13^1+1*8^1*14^1+1*7^1+1*8^1+1*13^1+1*14^1+1 = 0
    1*9^1*15^1+1*9^1*16^1+1*9^1+1*15^1+1*16^1+1 = 0
    1*3^1*7^1+1*3^1*8^1+1*4^1*7^1+1*4^1*8^1+1*5^1*15^1+1*5^1*16^1+1*7^1*15^1+
+1*7^1*16^1+1*3^1+1*4^1+1*5^1+1*8^1+1*15^1+1*16^1 = 0
    1*5^1*13^1+1*5^1*14^1+1*5^1*15^1+1*5^1*16^1+1*7^1*15^1+1*7^1*16^1+1*8^1*13^1+
+1*8^1*14^1+1*13^1*15^1+1*13^1*16^1+1*14^1*15^1+1*14^1*16^1+1*6^1+1*8^1+1 = 0
    1*13^1*15^1+1*13^1*16^1+1*14^1*15^1+1*14^1*16^1+1*13^1+1*14^1+1*15^1+1*16^1+
+1 = 0
    1*3^1*9^1+1*4^1*9^1+1*5^1*15^1+1*5^1*16^1+1*7^1*10^1+1*7^1*15^1+1*7^1*16^1+
+1*8^1*10^1+1*12^1*15^1+1*12^1*16^1+1*3^1+1*4^1+1*5^1+1*8^1+1*10^1+1*12^1+1 = 0
    1*5^1*15^1+1*5^1*16^1+1*7^1*15^1+1*7^1*16^1+1*5^1+1*7^1+1*15^1+1*16^1+1 = 0
    1*7^1*15^1+1*7^1*16^1+1*8^1*15^1+1*8^1*16^1+1*10^1*15^1+1*10^1*16^1+1*7^1+
+1*8^1+1*10^1+1*15^1+1*16^1+1 = 0
    1*2^1*13^1+1*2^1*14^1+1*4^1*13^1+1*4^1*14^1+1*10^1*15^1+1*10^1*16^1+
+1*12^1*15^1+1*12^1*16^1+1*7^1+1*8^1+1*10^1+1*12^1+1*15^1+1*16^1 = 0
    1*2^1*9^1+1*4^1*9^1+1*7^1*10^1+1*8^1*10^1+1*12^1*15^1+1*12^1*16^1+1*2^1+
+1*4^1+1*6^1+1*8^1+1*10^1+1*12^1+1*13^1+1*14^1+1*15^1+1*16^1+1 = 0
    1*12^1*15^1+1*12^1*16^1+1*12^1+1*15^1+1*16^1+1 = 0
    1*6^1+1*7^1+1*13^1+1*14^1+1 = 0
    1*9^1*13^1+1*9^1*14^1+1*9^1+1*13^1+1*14^1+1 = 0
```

Kai naudojama laipsninė ir atvirkštinė leksikografinė rikiavimo tvarka, Griobnerio bazes sudaro 26 lygčių:

```
    1*2^1*9^1+1*3^1*9^1+1*1^1*10^1+1*2^1*10^1+1*3^1*13^1+1*4^1*13^1+1*3^1*14^1+
+1*4^1*14^1 = 0
    1*5^1*9^1+1*7^1*9^1+1 = 0
    1*6^1*9^1+1*7^1*9^1+1*5^1*10^1+1*6^1*10^1+1*7^1*13^1+1*8^1*13^1+1*7^1*14^1+
+1*8^1*14^1+1 = 0
    1*11^1 = 0
    1*1^1+1*3^1+1*5^1+1*7^1 = 0
    1*5^1*10^1+1*7^1*10^1+1*10^1 = 0
    1*9^1*10^1+1*10^1 = 0
    1*7^1*10^1+1*8^1*10^1 = 0
```

```
    1*3^1*10^1+1*4^1*10^1+1*10^1 = 0
    1*10^1*13^1+1*10^1*14^1+1*10^1 = 0
    1*3^1*7^1*13^1+1*4^1*7^1*13^1+1*3^1*8^1*13^1+1*4^1*8^1*13^1+1*3^1*7^1*14^1+
+1*4^1*7^1*14^1+1*3^1*8^1*14^1+1*4^1*8^1*14^1+1*2^1*7^1+1*3^1*7^1+1*2^1*8^1+
+1*3^1*8^1+1*2^1*10^1+1*4^1*10^1+1*3^1*13^1+1*4^1*13^1+1*3^1*14^1+1*4^1*14^1+
+1*2^1+1*3^1 = 0
    1*2^1*13^1+1*3^1*13^1+1*2^1*14^1+1*3^1*14^1+1*2^1+1*3^1 = 0
    1*5^1*7^1+1*6^1*7^1+1*5^1*8^1+1*6^1*8^1+1*6^1*10^1+1*8^1*10^1+1*5^1+1*6^1+
+1*10^1 = 0
    1*2^1*10^1+1*4^1*10^1+1*3^1*13^1+1*4^1*13^1+1*3^1*14^1+1*4^1*14^1+1*2^1+
+1*3^1 = 0
    1*5^1*13^1+1*6^1*13^1+1*5^1*14^1+1*6^1*14^1+1*5^1+1*6^1 = 0
    1*2^1*5^1+1*3^1*5^1+1*2^1*7^1+1*3^1*7^1+1*2^1+1*3^1 = 0
    1*2^1*3^1+1*2^1*4^1+1*3^1*4^1+1*2^1 = 0
    1*3^1*5^1+1*4^1*5^1+1*3^1*6^1+1*4^1*6^1+1*2^1*7^1+1*3^1*7^1+1*2^1*8^1+
+1*3^1*8^1+1*6^1*10^1+1*8^1*10^1+1*10^1 = 0
    1*5^1*6^1+1*5^1*8^1+1*6^1*8^1+1*6^1*10^1+1*8^1*10^1+1*5^1+1*10^1 = 0
    1*6^1*10^1+1*8^1*10^1+1*7^1*13^1+1*8^1*13^1+1*7^1*14^1+1*8^1*14^1+1*6^1+
+1*7^1+1*10^1+1 = 0
    1*6^1*7^1+1*6^1*8^1+1*7^1*8^1+1*7^1*13^1+1*8^1*13^1+1*7^1*14^1+1*8^1*14^1+
+1*8^1 = 0
    1*3^1*6^1+1*4^1*6^1+1*2^1*7^1+1*4^1*7^1+1*2^1*8^1+1*3^1*8^1+1*7^1*13^1+
+1*8^1*13^1+1*7^1*14^1+1*8^1*14^1+1*3^1+1*4^1+1*6^1+1*7^1+1 = 0
    1*6^1*13^1+1*7^1*13^1+1*6^1*14^1+1*7^1*14^1+1*6^1+1*7^1+1*13^1+1*14^1+1 = 0
    1*3^1*9^1+1*4^1*9^1+1*3^1+1*4^1 = 0
    1*7^1*9^1+1*8^1*9^1+1*7^1+1*8^1+1*9^1+1 = 0
    1*9^1*13^1+1*9^1*14^1+1*9^1+1*13^1+1*14^1+1 = 0
```

4. **Pavyzdys**: $p = 2$, $m = 2$, $n = 2$, $t = \text{A}$.

Gauname $s = 12$ lygčių sistemą su $k = 22$ kintamaisiais:

```
    1*1^1*13^1+1*4^1*13^1+1 = 0
    1*1^1*14^1+1*2^1*13^1+1*2^1*14^1+1*4^1*13^1+1*4^1*19^1+1*4^1*20^1+1*5^1*19^1+
+1*5^1*20^1 = 0
    1*1^1*15^1+1*2^1*15^1+1*3^1*13^1+1*3^1*14^1+1*3^1*15^1+1*4^1*13^1+1*4^1*14^1+
+1*4^1*15^1+1*4^1*19^1+1*4^1*20^1+1*5^1*13^1+1*5^1*14^1+1*5^1*15^1+1*5^1*19^1+
+1*5^1*20^1+1*6^1*13^1+1*6^1*14^1+1*6^1*15^1 = 0
    1*1^1*16^1+1*4^1*16^1 = 0
    1*1^1*17^1+1*2^1*16^1+1*2^1*17^1+1*4^1*16^1+1*4^1*22^1+1*4^1*23^1+1*5^1*22^1+
+1*5^1*23^1 = 0
    1*1^1*18^1+1*2^1*18^1+1*3^1*16^1+1*3^1*17^1+1*3^1*18^1+1*4^1*16^1+1*4^1*17^1+
```

```
+1*4^1*18^1+1*4^1*22^1+1*4^1*23^1+1*5^1*16^1+1*5^1*17^1+1*5^1*18^1+1*5^1*22^1+
+1*5^1*23^1+1*6^1*16^1+1*6^1*17^1+1*6^1*18^1 = 0
     1*7^1*13^1+1*10^1*13^1+1 = 0
     1*7^1*14^1+1*8^1*13^1+1*8^1*14^1+1*10^1*13^1+1*10^1*19^1+1*10^1*20^1+
+1*11^1*19^1+1*11^1*20^1+1 = 0
     1*7^1*15^1+1*8^1*15^1+1*9^1*13^1+1*9^1*14^1+1*9^1*15^1+1*10^1*13^1+
+1*10^1*14^1+1*10^1*15^1+1*10^1*19^1+1*10^1*20^1+1*11^1*13^1+1*11^1*14^1+
+1*11^1*15^1+1*11^1*19^1+1*11^1*20^1+1*12^1*13^1+1*12^1*14^1+1*12^1*15^1+1 = 0
     1*7^1*16^1+1*10^1*16^1 = 0
     1*7^1*17^1+1*8^1*16^1+1*8^1*17^1+1*10^1*16^1+1*10^1*22^1+1*10^1*23^1+
+1*11^1*22^1+1*11^1*23^1+1 = 0
     1*7^1*18^1+1*8^1*18^1+1*9^1*16^1+1*9^1*17^1+1*9^1*18^1+1*10^1*16^1+
+1*10^1*17^1+1*10^1*18^1+1*10^1*22^1+1*10^1*23^1+1*11^1*16^1+1*11^1*17^1+
+1*11^1*18^1+1*11^1*22^1+1*11^1*23^1+1*12^1*16^1+1*12^1*17^1+1*12^1*18^1+1 = 0
```

Kai naudojama laipsninė ir leksikografinė rikiavimo tvarka, Griobnerio bazes sudaro 28 lygčių:

```
     1*7^1*13^1+1*10^1*13^1+1 = 0
     1*7^1*14^1+1*8^1*13^1+1*8^1*14^1+1*10^1*13^1+1*10^1*19^1+1*10^1*20^1+
+1*11^1*19^1+1*11^1*20^1+1 = 0
     1*7^1*15^1+1*8^1*15^1+1*9^1*13^1+1*9^1*14^1+1*9^1*15^1+1*10^1*13^1+
+1*10^1*14^1+1*10^1*15^1+1*10^1*19^1+1*10^1*20^1+1*11^1*13^1+1*11^1*14^1+
+1*11^1*15^1+1*11^1*19^1+1*11^1*20^1+1*12^1*13^1+1*12^1*14^1+1*12^1*15^1+1 = 0
     1*7^1*18^1+1*8^1*18^1+1*9^1*16^1+1*9^1*17^1+1*9^1*18^1+1*10^1*16^1+
+1*10^1*17^1+1*10^1*18^1+1*10^1*22^1+1*10^1*23^1+1*11^1*16^1+1*11^1*17^1+
+1*11^1*18^1+1*11^1*22^1+1*11^1*23^1+1*12^1*16^1+1*12^1*17^1+1*12^1*18^1+1 = 0
     1*16^1 = 0
     1*1^1+1*4^1+1*7^1+1*10^1 = 0
     1*17^1+1*18^1 = 0
     1*7^1*19^1+1*7^1*20^1+1*7^1*22^1+1*7^1*23^1+1*9^1*14^1+1*10^1*14^1+
+1*10^1*22^1+1*10^1*23^1+1*11^1*14^1+1*11^1*19^1+1*11^1*20^1+1*12^1*14^1+
+1*13^1*18^1+1*13^1*19^1+1*13^1*20^1+1*14^1*15^1+1*14^1*18^1+1*14^1*19^1+
+1*14^1*20^1+1*14^1*22^1+1*14^1*23^1+1*15^1*19^1+1*15^1*20^1+1*18^1*19^1+
+1*18^1*20^1+1*18^1*22^1+1*18^1*23^1+1*3^1+1*4^1+1*5^1+1*6^1+1*8^1+1*9^1+
+1*11^1+1*12^1+1*13^1+1*15^1+1*19^1+1*20^1 = 0
     1*7^1*22^1+1*7^1*23^1+1*8^1*14^1+1*8^1*18^1+1*9^1*14^1+1*9^1*19^1+1*9^1*20^1+
+1*9^1*22^1+1*9^1*23^1+1*11^1*14^1+1*11^1*18^1+1*11^1*22^1+1*11^1*23^1+
+1*12^1*14^1+1*12^1*19^1+1*12^1*20^1+1*12^1*22^1+1*12^1*23^1+1*13^1*15^1+
+1*13^1*19^1+1*13^1*20^1+1*14^1*15^1+1*14^1*18^1+1*15^1*22^1+1*15^1*23^1+
+1*18^1*22^1+1*18^1*23^1+1*3^1+1*6^1+1*7^1+1*9^1+1*11^1+1*12^1+1*13^1+1*15^1+1 = 0
     1*13^1*15^1+1*13^1*18^1+1*13^1*19^1+1*13^1*20^1+1*13^1*22^1+1*13^1*23^1+
+1*14^1*19^1+1*14^1*20^1+1*15^1*18^1+1*15^1*22^1+1*15^1*23^1+1*18^1*19^1+
```

```
+1*18^1*20^1+1*8^1+1*9^1+1*10^1+1*12^1+1*14^1+1*15^1+1*18^1+1*19^1+1*20^1 = 0
     1*14^1*15^1+1*14^1*22^1+1*14^1*23^1+1*15^1*18^1+1*15^1*19^1+1*15^1*20^1+
+1*18^1*19^1+1*18^1*20^1+1*18^1*22^1+1*18^1*23^1+1*3^1+1*6^1+1*9^1+1*12^1+1*14^1+
+1*15^1+1*19^1+1*20^1+1*22^1+1*23^1+1 = 0
     1*13^1*22^1+1*13^1*23^1+1*14^1*19^1+1*14^1*20^1+1*18^1*22^1+1*18^1*23^1+
+1*13^1+1*14^1+1*18^1 = 0
     1*14^1*19^1+1*14^1*20^1+1*18^1*22^1+1*18^1*23^1+1*19^1*22^1+1*19^1*23^1+
+1*20^1*22^1+1*20^1*23^1+1*2^1+1*4^1+1*10^1+1*11^1+1*14^1+1*15^1+1*18^1+1*19^1+
+1*20^1+1*22^1+1*23^1 = 0
     1*13^1*18^1+1*13^1*19^1+1*13^1*20^1+1*14^1*18^1+1*15^1*22^1+1*15^1*23^1+
+1*18^1*19^1+1*18^1*20^1+1*18^1*22^1+1*18^1*23^1+1*19^1*22^1+1*19^1*23^1+
+1*20^1*22^1+1*20^1*23^1+1*3^1+1*6^1+1*8^1+1*9^1+1*11^1+1*12^1+1*13^1+
+1*15^1+1*18^1+1 = 0
     1*14^1*22^1+1*14^1*23^1+1*15^1*18^1+1*15^1*22^1+1*15^1*23^1+1*19^1*22^1+
+1*19^1*23^1+1*20^1*22^1+1*20^1*23^1+1*2^1+1*3^1+1*4^1+1*6^1+1*14^1+1*18^1+
+1*19^1+1*20^1+1*22^1+1*23^1+1 = 0
     1*15^1*19^1+1*15^1*20^1+1*10^1+1*11^1+1 = 0
     1*13^1*19^1+1*13^1*20^1+1*18^1*22^1+1*18^1*23^1+1*19^1*22^1+1*19^1*23^1+
+1*20^1*22^1+1*20^1*23^1+1*2^1+1*4^1+1*13^1+1*18^1+1*22^1+1*23^1 = 0
     1*14^1*18^1+1*19^1*22^1+1*19^1*23^1+1*20^1*22^1+1*20^1*23^1+1*2^1+1*5^1+
+1*9^1+1*10^1+1*11^1+1*12^1+1*15^1+1*18^1+1 = 0
     1*15^1*22^1+1*15^1*23^1+1*18^1*19^1+1*18^1*20^1+1*3^1+1*4^1+1*5^1+1*6^1+
+1*9^1+1*10^1+1*11^1+1*12^1+1*19^1+1*20^1+1 = 0
     1*3^1+1*4^1+1*5^1+1*6^1+1*8^1+1*10^1+1*19^1+1*20^1+1*22^1+1*23^1 = 0
     1*15^1*18^1+1*18^1*22^1+1*18^1*23^1+1*8^1+1*10^1+1*15^1+1*18^1+1*19^1+1*20^1+
+1*22^1+1*23^1 = 0
     1*18^1*22^1+1*18^1*23^1+1*19^1*22^1+1*19^1*23^1+1*20^1*22^1+1*20^1*23^1+
+1*9^1+1*10^1+1*11^1+1*12^1+1*18^1+1 = 0
     1*2^1+1*4^1+1*22^1+1*23^1+1 = 0
     1*4^1+1*5^1+1*9^1+1*10^1+1*11^1+1*12^1+1*18^1+1*19^1+1*20^1+1 = 0
     1*8^1+1*9^1+1*10^1+1*12^1+1*15^1+1 = 0
     1*19^1*22^1+1*19^1*23^1+1*20^1*22^1+1*20^1*23^1+1*10^1+1*11^1+1*15^1+1*19^1+
+1*20^1+1*22^1+1*23^1 = 0
     1*9^1+1*10^1+1*11^1+1*12^1+1*19^1+1*20^1+1 = 0
     1*10^1+1*11^1+1*15^1+1 = 0
```

Kai naudojama laipsninė ir atvirkštinė leksikografinė rikiavimo tvarka, Griobnerio bazes sudaro 35 lygčių:

```
     1*3^1*13^1+1*4^1*13^1+1*5^1*13^1+1*6^1*13^1+1*3^1*14^1+1*4^1*14^1+1*5^1*14^1+
+1*6^1*14^1+1*1^1*15^1+1*2^1*15^1+1*3^1*15^1+1*4^1*15^1+1*5^1*15^1+1*6^1*15^1+
+1*4^1*19^1+1*5^1*19^1+1*4^1*20^1+1*5^1*20^1 = 0
```

1*7^1*13^1+1*10^1*13^1+1 = 0

1*8^1*13^1+1*10^1*13^1+1*7^1*14^1+1*8^1*14^1+1*10^1*19^1+1*11^1*19^1+
+1*10^1*20^1+1*11^1*20^1+1 = 0

1*9^1*13^1+1*10^1*13^1+1*11^1*13^1+1*12^1*13^1+1*9^1*14^1+1*10^1*14^1+
+1*11^1*14^1+1*12^1*14^1+1*7^1*15^1+1*8^1*15^1+1*9^1*15^1+1*10^1*15^1+1*11^1*15^1+
+1*12^1*15^1+1*10^1*19^1+1*11^1*19^1+1*10^1*20^1+1*11^1*20^1+1 = 0

1*16^1 = 0

1*1^1+1*4^1+1*7^1+1*10^1 = 0

1*2^1+1*3^1+1*5^1+1*6^1+1*8^1+1*9^1+1*11^1+1*12^1 = 0

1*7^1*15^1+1*10^1*15^1+1*15^1 = 0

1*13^1*15^1+1*15^1 = 0

1*14^1+1*15^1 = 0

1*8^1*15^1+1*9^1*15^1+1*11^1*15^1+1*12^1*15^1 = 0

1*4^1*15^1+1*5^1*15^1+1*15^1 = 0

1*10^1*15^1+1*11^1*15^1+1*15^1*19^1+1*15^1*20^1+1*15^1 = 0

1*4^1*10^1*19^1+1*5^1*10^1*19^1+1*4^1*11^1*19^1+1*5^1*11^1*19^1+
+1*10^1*13^1*19^1+1*11^1*13^1*19^1+1*9^1*15^1*19^1+1*12^1*15^1*19^1+
+1*4^1*10^1*20^1+1*5^1*10^1*20^1+1*4^1*11^1*20^1+1*5^1*11^1*20^1+1*10^1*13^1*20^1+
+1*11^1*13^1*20^1+1*9^1*15^1*20^1+1*12^1*15^1*20^1+1*3^1*10^1+1*4^1*10^1+
+1*5^1*10^1+1*6^1*10^1+1*8^1*10^1+1*3^1*11^1+1*4^1*11^1+1*5^1*11^1+1*6^1*11^1+
+1*8^1*11^1+1*10^1*11^1+1*3^1*15^1+1*6^1*15^1+1*4^1*19^1+1*5^1*19^1+1*8^1*19^1+
+1*11^1*19^1+1*13^1*19^1+1*4^1*20^1+1*5^1*20^1+1*8^1*20^1+1*11^1*20^1+1*13^1*20^1+
+1*3^1+1*4^1+1*5^1+1*6^1+1*11^1+1*15^1 = 0

1*15^1*19^1+1*15^1*20^1+1*15^1 = 0

1*8^1*19^1+1*10^1*19^1+1*13^1*19^1+1*8^1*20^1+1*10^1*20^1+1*13^1*20^1+1*8^1+
+1*10^1+1*13^1 = 0

1*3^1*4^1+1*3^1*5^1+1*4^1*6^1+1*5^1*6^1+1*3^1*7^1+1*4^1*7^1+1*5^1*7^1+
+1*6^1*7^1+1*4^1*9^1+1*5^1*9^1+1*7^1*9^1+1*8^1*9^1+1*4^1*10^1+1*5^1*10^1+
+1*7^1*10^1+1*8^1*10^1+1*3^1*11^1+1*6^1*11^1+1*7^1*11^1+1*8^1*11^1+1*4^1*12^1+
+1*5^1*12^1+1*7^1*12^1+1*8^1*12^1+1*10^1*13^1+1*11^1*13^1+1*7^1*19^1+1*9^1*19^1+
+1*10^1*19^1+1*12^1*19^1+1*13^1*19^1+1*7^1*20^1+1*9^1*20^1+1*10^1*20^1+
+1*12^1*20^1+1*13^1*20^1+1*7^1+1*11^1+1*19^1+1*20^1+1 = 0

1*3^1*7^1+1*6^1*7^1+1*4^1*9^1+1*5^1*9^1+1*7^1*9^1+1*7^1*10^1+1*8^1*10^1+
+1*9^1*10^1+1*3^1*11^1+1*6^1*11^1+1*7^1*11^1+1*8^1*11^1+1*4^1*12^1+1*5^1*12^1+
+1*7^1*12^1+1*10^1*12^1+1*4^1*13^1+1*5^1*13^1+1*10^1*13^1+1*11^1*13^1+1*3^1*15^1+
+1*6^1*15^1+1*9^1*15^1+1*12^1*15^1+1*3^1*19^1+1*6^1*19^1+1*9^1*19^1+1*12^1*19^1+
+1*13^1*19^1+1*3^1*20^1+1*6^1*20^1+1*9^1*20^1+1*12^1*20^1+1*13^1*20^1+1*3^1+1*6^1+
+1*8^1+1*9^1+1*11^1+1*12^1+1*15^1+1 = 0

1*4^1*8^1+1*5^1*8^1+1*3^1*10^1+1*6^1*10^1+1*8^1*10^1+1*3^1*11^1+1*4^1*11^1+
+1*5^1*11^1+1*6^1*11^1+1*8^1*11^1+1*10^1*11^1+1*3^1*15^1+1*6^1*15^1+1*9^1*15^1+
+1*12^1*15^1+1*4^1*19^1+1*5^1*19^1+1*10^1*19^1+1*11^1*19^1+1*4^1*20^1+1*5^1*20^1+
+1*10^1*20^1+1*11^1*20^1+1*3^1+1*6^1+1*8^1+1*9^1+1*12^1+1*15^1 = 0

1*3^1*8^1+1*6^1*8^1+1*3^1*9^1+1*4^1*9^1+1*5^1*9^1+1*6^1*9^1+1*8^1*9^1+
+1*3^1*10^1+1*6^1*10^1+1*9^1*10^1+1*3^1*12^1+1*4^1*12^1+1*5^1*12^1+1*6^1*12^1+
+1*8^1*12^1+1*10^1*12^1+1*4^1*13^1+1*5^1*13^1+1*10^1*13^1+1*11^1*13^1+1*7^1*19^1+
+1*9^1*19^1+1*11^1*19^1+1*12^1*19^1+1*13^1*19^1+1*7^1*20^1+1*9^1*20^1+1*11^1*20^1+
+1*12^1*20^1+1*13^1*20^1+1*7^1+1*8^1+1*19^1+1*20^1+1 = 0

1*7^1*9^1+1*8^1*10^1+1*8^1*11^1+1*9^1*11^1+1*10^1*11^1+1*7^1*12^1+
+1*11^1*12^1+1*4^1*13^1+1*5^1*13^1+1*3^1*15^1+1*6^1*15^1+1*4^1*19^1+1*5^1*19^1+
+1*7^1*19^1+1*9^1*19^1+1*11^1*19^1+1*12^1*19^1+1*4^1*20^1+1*5^1*20^1+1*7^1*20^1+
+1*9^1*20^1+1*11^1*20^1+1*12^1*20^1+1*3^1+1*6^1+1*8^1+1*9^1+1*10^1+1*11^1+1*12^1+
+1 = 0

1*9^1*10^1+1*9^1*11^1+1*10^1*12^1+1*11^1*12^1+1*4^1*13^1+1*5^1*13^1+
+1*9^1*15^1+1*12^1*15^1+1*3^1*19^1+1*4^1*19^1+1*5^1*19^1+1*6^1*19^1+1*7^1*19^1+
+1*10^1*19^1+1*13^1*19^1+1*3^1*20^1+1*4^1*20^1+1*5^1*20^1+1*6^1*20^1+1*7^1*20^1+
+1*10^1*20^1+1*13^1*20^1+1*3^1+1*6^1+1*7^1+1*9^1+1*11^1+1*12^1+1*13^1+1*15^1+1 = 0

1*4^1*7^1+1*5^1*7^1+1*4^1*9^1+1*5^1*9^1+1*8^1*9^1+1*3^1*10^1+1*4^1*10^1+
+1*5^1*10^1+1*6^1*10^1+1*7^1*10^1+1*8^1*10^1+1*3^1*11^1+1*6^1*11^1+1*7^1*11^1+
+1*8^1*11^1+1*9^1*11^1+1*4^1*12^1+1*5^1*12^1+1*8^1*12^1+1*11^1*12^1+1*4^1*13^1+
+1*5^1*13^1+1*10^1*13^1+1*11^1*13^1+1*4^1+1*5^1+1*7^1+1*10^1+1*13^1+1 = 0

1*7^1*10^1+1*8^1*10^1+1*7^1*11^1+1*8^1*11^1+1*3^1*15^1+1*6^1*15^1+1*9^1*15^1+
+1*12^1*15^1+1*4^1*19^1+1*5^1*19^1+1*13^1*19^1+1*4^1*20^1+1*5^1*20^1+1*13^1*20^1+
+1*3^1+1*4^1+1*5^1+1*6^1+1*7^1+1*8^1+1*13^1+1*15^1+1*19^1+1*20^1+1 = 0

1*9^1*15^1+1*12^1*15^1+1*10^1*19^1+1*11^1*19^1+1*10^1*20^1+1*11^1*20^1+1*9^1+
+1*10^1+1*11^1+1*12^1+1*15^1+1 = 0

1*8^1*10^1+1*8^1*11^1+1*10^1*11^1+1*4^1*13^1+1*5^1*13^1+1*10^1*13^1+
+1*11^1*13^1+1*3^1*19^1+1*4^1*19^1+1*5^1*19^1+1*6^1*19^1+1*7^1*19^1+1*11^1*19^1+
+1*3^1*20^1+1*4^1*20^1+1*5^1*20^1+1*6^1*20^1+1*7^1*20^1+1*11^1*20^1+1*3^1+1*6^1+
+1*7^1+1*8^1+1*9^1+1*11^1+1*12^1+1*13^1+1*19^1+1*20^1 = 0

1*3^1*19^1+1*4^1*19^1+1*5^1*19^1+1*6^1*19^1+1*7^1*19^1+1*9^1*19^1+
+1*11^1*19^1+1*12^1*19^1+1*13^1*19^1+1*3^1*20^1+1*4^1*20^1+1*5^1*20^1+1*6^1*20^1+
+1*7^1*20^1+1*9^1*20^1+1*11^1*20^1+1*12^1*20^1+1*13^1*20^1+1*3^1+1*4^1+1*5^1+
+1*6^1+1*7^1+1*9^1+1*11^1+1*12^1+1*13^1+1*19^1+1*20^1+1 = 0

1*4^1*9^1+1*5^1*9^1+1*8^1*9^1+1*3^1*10^1+1*6^1*10^1+1*3^1*11^1+1*6^1*11^1+
+1*9^1*11^1+1*4^1*12^1+1*5^1*12^1+1*8^1*12^1+1*11^1*12^1+1*10^1*13^1+1*11^1*13^1+
+1*3^1*15^1+1*6^1*15^1+1*4^1*19^1+1*5^1*19^1+1*10^1*19^1+1*11^1*19^1+1*13^1*19^1+
+1*4^1*20^1+1*5^1*20^1+1*10^1*20^1+1*11^1*20^1+1*13^1*20^1+1*3^1+1*6^1+1*8^1+
+1*9^1+1*11^1+1*12^1+1*19^1+1*20^1+1 = 0

1*10^1*13^1+1*11^1*13^1+1*13^1*19^1+1*13^1*20^1+1*10^1+1*11^1+1*19^1+1*20^1=0

1*13^1*19^1+1*13^1*20^1+1*13^1+1*19^1+1*20^1+1 = 0

1*8^1*9^1+1*9^1*11^1+1*8^1*12^1+1*11^1*12^1+1*4^1*13^1+1*5^1*13^1+1*7^1*19^1+
+1*10^1*19^1+1*7^1*20^1+1*10^1*20^1+1*4^1+1*5^1+1*7^1+1*8^1+1*10^1+1*11^1+1*19^1+
+1*20^1+1 = 0

1*4^1*13^1+1*5^1*13^1+1*4^1+1*5^1 = 0

```
    1*3^1*15^1+1*6^1*15^1+1*4^1*19^1+1*5^1*19^1+1*7^1*19^1+1*9^1*19^1+
+1*11^1*19^1+1*12^1*19^1+1*4^1*20^1+1*5^1*20^1+1*7^1*20^1+1*9^1*20^1+1*11^1*20^1+
+1*12^1*20^1+1*3^1+1*4^1+1*5^1+1*6^1+1*7^1+1*9^1+1*11^1+1*12^1 = 0
    1*9^1*19^1+1*10^1*19^1+1*11^1*19^1+1*12^1*19^1+1*9^1*20^1+1*10^1*20^1+
+1*11^1*20^1+1*12^1*20^1+1*9^1+1*10^1+1*11^1+1*12^1+1*19^1+1*20^1+1 = 0
    1*7^1*19^1+1*10^1*19^1+1*7^1*20^1+1*10^1*20^1+1*7^1+1*10^1+1*19^1+1*20^1+1= 0
```

4 PRIEDAS. PRANEŠIMAI

Vienos vienkryptės funkcijos apibrėžimas panaudojant braid grupę

P. Vitkus, lekt. P. Tvarijonas
Kauno technologijos universitetas

Kriptografinės sistemos, kurių pagalba subjektai apsikeičia informacija viešais ryšio kanalais, naudoja vienkryptes funkcijas privačiai informacijai paslėpti. Vienkryptė funkcija yra tokia funkcija, kurią paprasta apskaičiuoti, tačiau labai sunku (atliekamų operacijų ir trunkančio laiko prasme) apskaičiuoti jai atvirkštinę funkciją. Dažnai naudojamos landinės vienkryptės funkcijos, kurioms galima apskaičiuoti atvirkštinę funkciją panaudojant tam tikrą papildomą informaciją – landinį parametrą.

Klasikinis tokios vienkryptės funkcijos pavyzdys yra didelių pirminių skaičių daugyba. Problema iškyla sudarant atvirkštinę funkciją, kai norima gautą sandaugą išskaidyti pirminiais dauginamaisiais. Šia matematine problema remiasi plačiai paplitusi RSA šifravimo sistema. Panašus pavyzdys yra ir diskretaus logaritmo problema paremtas Diffie'io ir Hellmano raktų apsikeitimo protokolas [4].

Aukščiau paminėtos kriptografinės sistemos šiomis dienomis yra plačiai taikomos. Tačiau joms yra daugiau nei 30 metų ir sparčiai tobulėjant matematiniams metodams bei skaičiavimo technikai (kompiuteriams) šios sistemos gali tapti nebepatikimos. Todėl yra ieškoma naujų būdų kaip kriptografiškai apsaugoti duomenis. Pastarąjį dešimtmetį didelio dėmesio susilaukė braid grupės, kurios siūlo įvairių potencialiai sudėtingų matematinių problemų. Jų pagrindu galima kurti vienkryptes funkcijas.

Pirmasis braid grupes apibrėžė Artin'as [2, 3]. Nekomutatyvi grupė B_n vadinama n-braid'ų grupe, kurią aprašo n-1 generatorius σ_i (i = 1, 2, ..., n-1) bei vienetinis elementas e ir yra tenkinami sąryšiai:

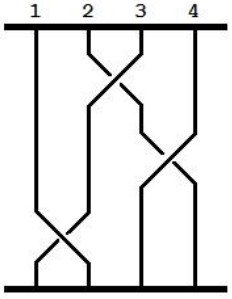
$\sigma_i\sigma_j = \sigma_j\sigma_i$, $\quad |i - j| > 1$;

$\sigma_i\sigma_j\sigma_i = \sigma_j\sigma_i\sigma_j$, $\quad |i - j| = 1$.

Taip vadinami Artin'o generatoriai turi labai gražią geometrinę interpretaciją. Įsivaizduokime, kad n-braid'ą sudaro n juostų (*angl.* braid – juosta, kasa) einančių iš viršaus į apačią, tada generatorius σ_i reiškia, kad susikerta i ir i+1 juostos. Sutarta, kad juosta i eina po juosta i+1, kai turime σ_i, ir juosta i eina virš juostos i+1, kai turime σ_i^{-1}. Pavyzdžiui, braid'as $w = \sigma_2\sigma_3\sigma_1^{-1}$ pavaizduotas **1. paveiksle**.

Viena iš matematinių problemų braid grupėse yra jungtinio paieškos problema (*angl.* conjugator search problem), kuri apibrėžiama taip: turime du dydžius $a, b \in B_n$ ir $a = xbx^{-1}$, reikia rasti nežinomą dydį $x \in B_n$.

Aukščiau pateiktas braid grupės apibrėžimas yra abstraktus ir vadinamas braid grupės apibrėžimu raiškos lygmenyje: grupė vaizduojama aibe generatorių ir aibe ryšių tarp jų. Kaip ir daugelis kitų grupė B_n gali būti atvaizduota matricų grupe virš polinomų žiedo $\mathbb{Z}[t^{\pm 1}]$. Tai matricos, kurių elementai yra polinomai, priklausantys nuo kintamojo t ir atvirkštinio kintamojo t^{-1}, o polinomų koeficientai priklauso žiedui $\mathbb{Z}$. Toks atvaizdavimas vadinamas Burau atvaizdavimu.



1.pav. Braid'as $w = \sigma_2\sigma_3\sigma_1^{-1}$.

Mes naudojame B_n atvaizdavimą matricų grupe virš Galua lauko (*angl.* Galois field) $GF(2^k)$ [4].

Apibrėšime vienkryptę funkciją $A = XB^rX^{-1}$ [1]. Čia A, B ir X yra braid grupės elementai, atvaizduoti Burau matricomis, o r – didelis sveikasis skaičius.

Apskaičiuojant vienkrypte funkciją turime X, B ir r, o reikia rasti A. Norint rasti vienkryptės funkcijos atvirkštinę reikšmę žinome A ir B, o reikia rasti X bei r.

Norint surasti slaptą parametrą r reikia išspręsti diskretaus logaritmo problemą matricoms, tai yra ne lengviau nei diskretaus logaritmo problema vienam sveikajam skaičiui. O ieškant slapto parametro A reikia išspręsti jungtinio paieškos problemą braid grupės įvaizdyje, t. y. rasti jungtinę matricą A.

Mes teigiame, kad tokios vienkryptės funkcijos atvirkštinės reikšmės apskaičiavimo uždavinys yra sudėtingas. Akivaizdu, jog vienkryptės funkcijos apskaičiavimo uždavinys yra algoritmiškai lengvai sprendžiamas, nes apskaičiuojant matricą A reikia atlikti $O(\lg r)$ matricų daugybos veiksmų.

Braid grupių panaudojimas sudarant šią vienkryptę funkciją yra naudingas tuo, kad leidžia efektyviai koduoti matricas ir lengvai rasti jų atvirkštines.

Literatūra:

1. E. Sakalauskas, P. Tvarijonas, A. Raulynaitis, Key agreement protocol (KAP) using conjugacy and discrete logarithm problems in group representation level, 2007.
2. A. Bolstad, Braid group cryptography untangled, 2004.
3. Ki Hyoung Ko, Sang Jin Lee, Jung Hee Cheon, Jae Woo Han, Ju-sung Kang, Choonsik Park, New public-key cryptosystem using braid groups, 2000.
4. A. Menezes, P. van Oorschot, S. Vanstone, Handbook of applied cryptography, 1996.

KEY AGREEMENT PROTOCOL USING ELLIPTIC CURVE MATRIX POWER FUNCTION*

Artūras Katvickis, Paulius Vitkus

Abstract: The key agreement protocol (KAP) using elliptic curve matrix power function is presented. This function pretends be a one-way function since its inversion is related with bilinear equation solution over elliptic curve group. The matrix of elliptic curve points is multiplied from left and right by two matrices with entries in Z_n.

Some preliminary security considerations are presented.

Keywords: key agreement protocol, elliptic curve cryptography, NP-complete problem.

ACM Classification Keywords: E.3 Data Encryption, F.2.1 Numerical Algorithms and Problems.

Conference topic: Cryptography.

Introduction

Key agreement protocols (KAP) is one of the basic cryptographic protocols. KAP allows two or more parties negotiate a common secret key using insecure communications.

First KAP was presented by Diffie-Hellman [Diffie, Hellman, 1976] which caused rapid development of asymmetric cryptography.

In 1993 new ideas appeared in asymmetric cryptography [Sidelnikov et al, 1993] – using known hard computational problems in infinite non- commutative groups instead of hard number theory problems such as discrete logarithm or integer factorization problems to construct one-way functions.

This idea was realized in [Anshel et al, 1999] where KAP was constructed using conjugator search problem and membership problem in Braid groups. The similar result was presented in [Ko et al, 2000].

Later, [Shpilrain, Ushakov, 2004] showed that conjugator search problem does not produce sufficient security level. The others hard problems were investigated to construct KAP and were based on triple decomposition problem [Kurt, 2006], subgroup membership problem [Shpilrain, Zapata, 2006] and elliptic curve pairing [Smart, 2002].

The idea to use non-commutative infinite group (e.g. braid group) representation was also used for the other kind of one-way functions construction as a background of both digital signature scheme and key agreement protocol [Sakalauskas, 2005], [Sakalauskas et al, 2007]. The (semi)group representation level allows us to avoid a significant problem of hiding the factors in the publicly available group word when using its presentation level. The hiding of factors in representation level occurs in a very natural way. However, the original hard problems, such as conjugator search or decomposition problems in (semi)group presentation level are considerably weakened when they are transformed into the representation level. Therefore using representation level these problems must be considerably strengthened by simultaneously adding the other additional hard problems.

In this paper we present KAP using elliptic curve matrix power function. This function pretends be a one-way function since its inversion is related with bilinear equation over elliptic curve group. The matrix of elliptic curve points is left and right side multiplied by two matrices with entries in Z_n.

* Work is partially supported by the Lithuanian State Science and Studies Foundation

Mathematical background

Let $p > 3$ be a prime integer. An elliptic curve $E_p(a, b)$ over GF(p) is defined by equation

$$y^2 = x^3 + ax + b, \quad (1)$$

where $a, b \in$ GF(p) and $4a^3 + 27b^2 \bmod p \neq 0$.

The addition operation between two points $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ on elliptic curve is written in following algebraic formulas:

$$\begin{aligned} x_3 &= \lambda^2 - x_1 - x_2, \\ y_3 &= \lambda(x_1 - x_3) - y_1, \end{aligned} \quad (2)$$

where $\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1}, & P \neq Q, \\ \dfrac{3x_1^2 + a}{2y_1}, & P = Q. \end{cases}$

A set of all points (x, y), $a, b \in$ GF(p), which satisfy (1) equation, together with special point O, called infinity point, and addition operation forms a finite cyclic group with O as its identity.

Another operation, defined on elliptic curve is multiplication of point P by integer k. This operation is defined straightforward, i.e. $4P = P + P + P + P$.

Elliptic curve group order $n = \#E_p(a, b)$ can be roughly estimated using Hasse theorem [York, 1992]:

Let $E_p(a, b)$ is a group on elliptic curve $y^2 = x^3 + ax + b$ and $t = p + 1 - \#E_p(a, b)$. Then

$$|t| \leq 2\sqrt{p}\,. \quad (3)$$

Equation (3) can be rewritten in more comfortable form:

$$p + 1 - 2\sqrt{p} \leq \#E_p(a, b) \leq p + 1 + 2\sqrt{p}.$$

Since elliptic curve group is cyclic with order n, fixed point P multiplication by any integer k can be replaced with multiplication by number $\tilde{k} \in Z_n$, where $\tilde{k} = k \bmod n$ and $0P = O$, i.e. any point multiplied by zero is an infinity point.

Key agreement protocol (KAP)

Now we propose the following two parties key agreement protocol.

1. Parties agree on publicly available matrix Q over elliptic curve $E_p(a, b)$ and matrices L, R over Z_n.

2. Alice randomly generates two secret sequences $\{x_i\}$, $\{y_i\}$, $i = 0, 1, \ldots, k$ in Z_n and computes

$$X = \sum_{i=0}^{k} x_i L^i = x_0 I + x_1 L + \ldots + x_k L^k ,$$

$$Y = \sum_{i=0}^{k} y_i R^i = y_0 I + y_1 R + \ldots + y_k R^k .$$

3. Bob randomly generates two secret sequences $\{u_i\}$, $\{v_i\}$, $i = 0, 1, \ldots, k$ in Z_n and computes

$$U = \sum_{i=0}^{k} u_i L^i = u_0 I + u_1 L + \ldots + u_k L^k ,$$

$$V = \sum_{i=0}^{k} v_i R^i = v_0 I + v_1 R + \ldots + v_k R^k .$$

4. Alice computes intermediate value K_A and sends result to Bob.

$$K_A = XQY \qquad (4)$$

5. Bob computes intermediate value K_B and sends result to Alice.

$$K_B = UQV \qquad (5)$$

6. Since matrices X, U and Y, V are commutative, both parties compute common secret key

$$K = XK_BY = UK_AV = XUQVY. \qquad (6)$$

Preliminary security analysis

The security parameters are matrix dimension m, elliptic curve group order n and secret sequences length k. They must be large enough to prevent brute force attack. To compromise the key K, the adversary must solve the (4), (5) matrix equations to find X, Y and U, V with known instances Q, K_A, K_B.

Let $X = \{x_{ij}\}$, $Y = \{y_{ij}\}$, $Q = \{Q_{ij}\}$, $A = \{A_{ij}\}$ are matrices of 2-nd order. Then matrix equation $XQY = K_A = A$ can be rewritten as system of bilinear equation over elliptic curve group:

$$\begin{cases} x_{11}y_{11}Q_{11} + x_{11}y_{21}Q_{12} + x_{12}y_{11}Q_{21} + x_{12}y_{21}Q_{22} = A_{11} \\ x_{11}y_{12}Q_{11} + x_{11}y_{22}Q_{12} + x_{12}y_{12}Q_{21} + x_{12}y_{22}Q_{22} = A_{12} \\ x_{21}y_{11}Q_{11} + x_{21}y_{21}Q_{12} + x_{22}y_{11}Q_{21} + x_{22}y_{21}Q_{22} = A_{21} \\ x_{21}y_{12}Q_{11} + x_{21}y_{22}Q_{12} + x_{22}y_{12}Q_{21} + x_{22}y_{22}Q_{22} = A_{22} \end{cases} \qquad (7)$$

We do not know the actual complexity of such systems. It is known that solution of a system of polynomial equations over any field is NP-Complete [Garey, Jonson, 1979]. But in this case the obtained system is not over the field. This system can be interpreted also as a system of equations in vector space of elliptic curve points over Z_n. Thus, we can make a conjecture that solving a system of bilinear equations over elliptic curve points vector space is not easier than solving a system of bilinear polynomial equations over any field.

We can also refer to Schaefer Dixotomy theorem for a constraint satisfiability problem denoted by SAT(S) [Schaefer, 1978]. In general, the complexity of any computational problem can be estimated by reformulating this problem into the decisional problem and reducing some known NP-Complete problem into this decisional

problem. Without proof we assert that there is a SAT(*S*) problem reducible in polynomial time to the decisional problem corresponding to (4), (5).

On the other hand, notice that proposed KAP is a generalized elliptic curve Diffie-Hellman KAP (ECDH). Indeed, if we set matrix dimension to $m = 1$ and secret sequence length to $k = 1$, we get algorithm similar to ECDH.

Further investigations are required to select the values of security parameters and estimate the security level.

Bibliography

[Anshel et al, 1999] Anshel I., Anshel M., Goldfeld D. An algebraic method for public key cryptography. Mathematical Research Letters 6, pp. 1–5, 1999.

[Diffie, Hellman, 1976] Diffe W., Hellman M.. New Directions in Cryptography. In IEEE Transaction on Information Theory, IT-22 (6, 644-654), 1976.

[Garey, Jonson, 1979] Garey M. R., Johnson D. S. Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman and Company, 1979.

[Ko et al, 2000] Ko K. H., Lee S. J., Cheon J. H., Han J. W., Kang J. S., Park C. New Public key Cryptosystem Using Braid Groups. Advances in Cryptology, Proc. Crypto 2000, LNCS 1880, Springer–Verlag, pp. 166–183, 2000.

[Kurt, 2006] Kurt Y. A New Key Exchange Primitive Based on the Triple Decomposition Problem. Available at: http://eprint.iacr.org/2006/377, 2006

[Sakalauskas, 2005] Sakalauskas E., One Digital Signature Scheme in Semimodule over Semiring, *Informatica*. ISSN: 0868-4952, vol. 16, no. 3(2005), pp. 383-394.

[Sakalauskas et al, 2007] Sakalauskas E., Tvarijonas P., Raulinaitis A., *Key Agreement Protocol (KAP) Using Conjugacy and Discrete Logarithm Problems in Group Representation Level,* Informatica, Vol. 18, No. 1, 2007, pp. 115-124.

[Sidelnikov et al, 1993] Sidelnikov V., Cherepnev M., Yaschenko V. Systems of open distribution of keys on the basis of non-commutative semigroups. Russian Acad. Sci. Dokl. Math., 48(2), pp. 566-567, 1993.

[Schaefer, 1978] Schaefer T.J., The Complexity of Satisfiability Problems. In Proceedings 10th ACM Symposium on Theory of Computing, 216-226, 1978.

[Shpilrain, Ushakov, 2004] Shpilrain V., Ushakov A., The conjugacy search problem in public key cryptography: unnecessary and insufficient, Available at: http://eprint.iacr.org/2004/321, 2004.

[Shpilrain, Zapata, 2006] Shpilrain V., Zapata G. Using the subgroup membership search problem in public key cryptography. Contemp. Math., Amer. Math. Soc. 418 (2006), 169–179

[Smart, 2002] Smart N. P. An Identity Based Authenticated Key Agreement Protocol Based on the Weil Pairing. *Electronics Letters*, 38 (13). ISSN 00135194, pp. 630–636. 2002.

[York, 1992] York E. Elliptic Curves Over Finite Fields. Aviable at: http://www.math.rochester.edu/people/grads/jdreibel/ ref/yorkECC.pdf, 1992.

Authors' Information

Artūras Katvickis *– PhD student, Department of Applied Mathematics, Kaunas University of Technology, Studentu str. 50-324a, Kaunas, LT-51368, Lithuania, e-mail arturas.katvickis@ktu.lt*

Paulius Vitkus – M.Sc. student, Department of Applied Mathematics, Kaunas University of Technology, Studentu str. 50-324a, Kaunas, LT-51368, Lithuania, e-mail paulius.vitkus@ktu.lt