VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ KATEDRA

# McEliece kriptografinės sistemos saugumo tyrimas

## Study of the security of McEliece cryptosystem

Magistro baigiamasis darbas

| | | |
|---|---|---|
| Atliko: | Mindaugas Guobys | (parašas) |
| Darbo vadovas: | Doc. Gintaras Skersys | (parašas) |
| Recenzentas: | Lekt. Albertas Šermokas | (parašas) |

Vilnius – 2008

## Santrauka

Darbo tema - viešojo rakto McEliece kriptografinės sistemos saugumo tyrimas. Darbe galima rasti kriptografijos temų detalią apžvalgą, išaiškintos rečiau naudojamos sąvokos. Atliktas atakų prieš viešojo rakto kriptografinių sistemų klasifikavimas. Tyrimo metu išanalizuotos kriptosistemos sudedamosios dalys, aptarta jų įtaka bendram sistemos saugumui. Darbe detaliai aprašytos realizuotos atakos prieš McEliece kriptografinę sistemą bei pateikti atakų vykdymo rezultatai bei apskaičiuota kiek truktų kiekviena ataka su saugiais laikomais parametrais *t, k, n*. Ištirtas atakų pavojingumas bei pateiktos išvados kaip nuo populiariausių atakų apsiginti.



**Raktiniai žodžiai:** McEliece, kriptosistema, kriptografija, viešasis raktas, privatus raktas, ataka, šifras, kriptoanalitikas, pranešimas.

## Summary

Master thesis topic is McEliece public key cryptosystem's security research. There is cryptography topics detailed review and explained in details rarely used notions. There some public key cryptosystems attacks classifications were done. There were cryptosystem components analysis and dependence on all system security also done. In this work we can find explained attacks against McEliece cryptosystem and results of tests with real world parameters t, k, n. In this work there was also explained dangerous attacks and presented conclusions how to protected against most popular attacks.

**Keywords**: McEliece, cryptosystem, cryptography, public key, private key, attack, cipher, cryptanalytic, message.

Turinys

1. Įvadas ..... 6
2. Kriptografijos apžvalga ..... 9
2.1. Kriptografijos temų apžvalga ..... 9
2.1.1. Kriptografijos aspektai ..... 9
2.1.2. Matematinės problemos ..... 11
2.1.3. Pagrindinės kriptografinės sistemos ..... 12
2.1.4. Atakų tipai ..... 13
2.1.5. Klaidas taisančių kodų taikymas kriptografijoje ..... 14
2.1.6. Kriptosistemos naudojančios tiesinius kodus ..... 14
2.2. McEliece ir Niederreiter kriptografinių sistemų veikimas ..... 15
2.2.1. McEliece kriptografinė sistema ..... 15
2.2.2. Niederreiter kriptografinė sistema ..... 16
2.2.3. Goppa kodai bei Goppa polinomai ..... 17
2.2.4. Matricos S ir P ..... 17
2.2.5. Klaidos vektorius ..... 18
2.2.6. Kriptosistemos silpnosios vietos ..... 18
2.2.7. Silpnų raktų išvengimas ..... 19
2.3. Atakos prieš McEliece kriptosistemą ..... 20
2.3.1. Plačiausiai žinomos atakos prieš McEliece ..... 20
2.3.2. Atakų realizacija ..... 21
3. Tiriamosios atakos prieš McEliece kriptosistemą ..... 22
3.1. Apžvalga ..... 22
3.2. Pavyzdinio pranešimo užšifravimas ..... 23
3.3. Susijusių pranešimų atakos algoritmas ..... 26
3.4. Susijusių pranešimų atakos realizacija ..... 29
3.5. Susijusių pranešimų atakos rezultatai ..... 32
3.6. Gavėjo atsako atakos algoritmas ..... 33
3.7. Gavėjo atsako atakos realizacija ..... 35
3.8. Visų galimų variantų perrinkimo atakos algoritmai ..... 36
3.9. Visų galimų variantų perrinkimo atakos realizacija ..... 37
3.10. Žinomo dalinio teksto atakos algoritmas ..... 38
3.11. Žinomo dalinio teksto atakos realizacija ..... 39

4. Išvados.....41
5. Šaltinių sąrašas.....42

# 1. Įvadas

Magistrinio darbo tema – vienos iš pirmųjų viešojo rakto McEliece kriptografinės sistemos [McE78] saugumo tyrimas. Ši kriptografinė sistema pavadinta jos kūrėjo vardu. Robert J. McEliece yra matematikos bei inžinerijos profesorius dirbantis Jungtinėse Amerikos Valstijose Kalifornijos technologijų institute. Profesorius labiausiai pagarsėjęs kaip besidomintis informacijos teorija, klaidas taisančiais kodais, skaitmenine komunikacija, diskrečiąja matematika ir pan.

Nuolatos besivystančios informacinės technologijos, tiek programinės įrangos, tiek ir aparatūrinės įrangos leidžia siekti aukščiausių tikslų. Norint išlaikyti geras pozicijas versle ar įgauti pranašumą prieš konkurentus būtina ne tik puikiai išmanyti dalykinę sritį, bet ir lanksčiai adaptuoti verslo poreikius tenkinančias informacines technologijas. Asmeniniame gyvenime pažangios informacinės technologijos taupo laiką, suteikia tam tikrą komfortą, atsiranda naujų galimybių, spartėja informacijos sklaida. Dėl gausėjančių Interneto vartotojų skaičiaus bei anksčiau paminėtų priežasčių informaciniais kanalais yra siunčiami svarbūs dokumentai, publikuojama tik tam tikram žmonių ratui skirta konfidenciali informacija, atliekama vis daugiau bankinių operacijų, svarstoma apie galimybę suteikti rinkėjams teisę balsuoti Internetu savivaldos, Seimo ir prezidento rinkimuose, gvildenamos skaitmeninio parašo teisinės peripetijos. Todėl duomenų saugumo, informacijos siuntėjo, tarpininko bei gavėjo autorizacijos, identifikavimo problemos įgauna vis didesnę svarbą ir tampa ypač aktualiomis šiomis dienomis.

McEliece viešojo rakto kriptografinės sistemos kritikai vienu iš didžiausiu minusu laiko, jog užšifruotas pranešimas yra didesnis nei originalusis – ne šifruotas [MOV96]. Kuo toliau, tuo šis aspektas tampa vis mažiau svarbus, nes duomenų saugojimo talpyklos bei duomenų perdavimo greičiai tobulėjant technologijoms auga dideliais tempais ir duomenų saugumas tampa prioritetiniu veiksniu. Dėl anksčiau aprašytos priežasties šios viešojo rakto kriptografinės sistemos panaudojimas praktikoje turėtų gerokai išaugti netolimoje ateityje.

Pagrindinis darbo tikslas yra ištirti McEliece kriptografinės sistemos saugumą bei pateikti rekomendacijas didesniam saugumui užtikrinti prieš galimas atakas, įvardinant kriterijus, prie kurių ši kriptografinė sistema negalėtų būti sėkmingai naudojama. Norint pasiekti darbo tikslą reiks atlikti darbe šiuos uždavinius:

1. Kriptografijos temų apžvalga
    1.1. Bendrų kriptografijos aspektų apžvalga: kriptografijos uždaviniai bei tikslai, pagrindinės saugumo sąvokos, saugumo lygiai.

1.2. Matematinės problemų, kuriomis remiamasi kriptografinėse sistemose, apžvalga (didelių skaičių faktorizavimas, diskretaus logaritmo problema, tiesinių kodų dekodavimas, elipsinės kreivės ir t.t.).
1.3. Galimų atakų identifikavimas bei jų apžvalga.
1.4. Kriptosistemų, naudojančių tiesinius kodus, apžvalga.
1.5. Klaidas taisančių kodų taikymo praktikoje apžvalga.

2. McEliece kriptosistemos sandara bei jos įtaka saugumui
   2.1. Išnagrinėti Goppa kodų bei Goppa polinomų įtaką saugumui.
   2.2. Išnagrinėti matricų S ir P įtaką saugumui.
   2.3. Išnagrinėti klaidos vektoriaus įtaką saugumui.
   2.4. Apžvelgti kriptosistemos silpnas vietas (silpnus raktus ir pan.).
   2.5. Išnagrinėti silpnų raktų bei vietų išvengimo būdus.
3. Atakos prieš McEliece kriptosistemą
   3.1. Atakų analizė
   3.2. Realizuoti kelias žinomas atakas
   3.3. Įvykdyti atakas su mažais kriptosistemos parametrais
   3.4. Surinkti bei įvertinti statistinius atakų vykdymo rezultatus
   3.5. Teoriškai įvertinti, kokie būtų rezultatai su realiais parametrais
   3.6. Palyginti atakų vykdymo rezultatus su teoriniais rezultatais.

Kaip matyti iš užsibrėžtų darbo uždavinių, didelis dėmesys tyrimo metu bus skiriamas praktinei veiklai. Tyrimas bus paremtas ne tik teorinėmis žiniomis bei argumentais, bet ir atliktais praktiniais bandymais. Toks tyrimo metodas leis tiksliau įvertinti prielaidas, neleis skaičiavimuose įsivelti klaidoms.

Tyrimo metu bus remiamasi Internetinėje erdvėje rastomis bei rekomenduotomis publikacijomis. Absoliučiai didžiausią dalį literatūros sudaro moksliniai darbai anglu kalba. Pagrindinių sąvokų bei principų galima rasti ir lietuviškoje literatūroje [Sta05]. Darbe bus skirtas didelis dėmesys literatūros apžvalgai, akcentuoti pastebėti literatūros trūkumai bei gerosios pusės. Šiame skyriuje taip pat galima bus rasti nuorodas į tam tikrų dalių, kurių plačiai darbe nebus aprašoma, detales.

Rašant magistro darbą bus remiamasi praeitų semestrų darbais, mat juose jau buvo atlikta detali literatūros apžvalga bei analizė, sudarytas detalusis darbų planas bei gauti principiniai darbo sprendimai.

Baigiamojo darbo kūrimui teks naudoti MS Word programą teksto rinkimui bei JCreator programą Java programų rašymui bei testavimui atliekant atakas prie McEliece kriptosistemą. Nepavyks išsiversti be nestandartinių Java bibliotekų. Generuojančios matricos sudarymui bus naudojama Flexiprovider paketas, skirtas darbui su kriptografiniais objektais. Tokį pasirinkimą nulėmė noras daugiau dėmesio skirti pačių atakų įgyvendinimui.

# 2. Kriptografijos apžvalga

## 2.1. Kriptografijos temų apžvalga

### 2.1.1. Kriptografijos aspektai

Kriptografijos sąvokos yra žinomos daugumai tiksliųjų mokslų bakalaurams ar aukštesnį išsilavinimą turintiems žmonėms. Daugelyje universitetų yra dėstomi vienokie ar kitokie kursai susiję su kriptografija, tad susijusios literatūros yra pakankamai. Joje galima rasti bendrus kriptografijos principus ar nagrinėjamas siauresnes sritis. Pati kriptografija apibrėžiama panašiai: „...galima kriptografiją apibūdinti kaip informacijos apsaugos priemonių kūrimo <..> sritį“ [Sta05], arba konkrečiau - „Kriptografija yra mokslas apie matematinius metodus skirtus informacijos saugumui užtikrinti kaip konfidencialumas, vientisumas, objektų bei duomenų autorizacija.“ [MOV96].

Pagrindiniu kriptografijos uždaviniu yra laikomas keturių informacijos saugumo tikslų užtikrinimas. Informacijos saugume galima rasti ir žymiai daugiau sprendžiamų problemų, tačiau yra pagrindinės keturios sritys, o likusios problemos yra pagrindinių ištakai. Taigi, kriptografija siekia užtikrinti:

- konfidencialumą;
- duomenų vientisumą;
- autorizaciją;
- veiksmų neatsisakymą (ang. *non-repudiation*).

Konfidencialumas siekia užtikrinti informacijos prieinamumą tik tiems objektams, kuriems ta informacija yra skirta. Konfidencialumui užtikrinti yra daugybė būtų, kaip fizinė duomenų apsauga ar matematiniai algoritmai, kurių dėka informacija tampa nesuprantama žmonėms, kuriems ji nėra skirta.

Duomenų vientisumas siekia neleisti modifikuoti duomenų asmenims, kuriems to negalima daryti. Duomenų modifikavimu laikomos tokios operacijos su duomenimis kaip duomenų pridėjimas, trynimas ar vienų duomenų pakeitimas kitais. Pasireiškus duomenų daugialypiškumui bei neinformavus informacijos gavėjo apie duomenų modifikavimą, jis yra suklaidinamas.

Autorizacijos sąvoka apima tiek duomenų šaltinių tiek ir jos naudotojų identifikaciją. Kuomet nori užmegzti ryšį dvi pusės, jos turi viena kitą atpažinti. Informacija siųsta per

duomenų kanalus turi būti atpažįstama pagal siuntėja, informacijos turinį, siuntimo laiką ir pan., paprastai kriptografija šioje vietoje skyla į dvi dalis: objekto autorizacija bei duomenų siuntėjo autorizaciją. Pastaroji dalis glaudžiai susijusi su duomenų vientisumo problema.

Veiksmų neatsisakymu vadinamas toks veiksmas, kuomet padaryto veiksmo objektas nebegali atšaukti, pavyzdžiui, žmogus pervedęs pinigus iš savo sąskaitos į verslo partnerio sąskaitą, tų pinigų susigrąžinti nebegali. Šio informacijos saugumo principo nebūvimas labai įtakotų elektroninę prekybą bei visas kitas finansines operacijas.

Taigi, pagrindiniu kriptografijos tikslu galima vadinti šių keturių problemų tiek teorinius, tiek ir praktinius sprendimus. Kriptografija skirta informacijos apsaugos prevencijai bei sukčiavimo atvejų atpažinimui [MOV96].

Apžvelgsime kaip literatūroje yra apibrėžiamos pagrindinės informacijos saugumo sąvokos. „Kriptosistema vadinama besąlygiškai saugia, jei net ir turėdamas beribius skaičiavimo resursus negali nežinodamas rakto iš šifro nustatyti, koks pranešimas buvo išsiųstas.“ – taip apibrėžiama besąlygiško saugumo sąvoka [Sta07]. „... kriptosistemos saugumas yra įrodomas, jeigu galima įrodyti, kad sistemos įveikimas yra tolygus matematinės (dažniausiai skaičių teorijos) problemos, kuri laikoma sunkia, išsprendimui.“ – taip autorius apibrėžia saugumo įrodomumo sąvoką [Sta07]. „Kriptosistema vadinama skaičiavimų požiūriu saugia, jeigu pasiektas skaičiavimų resursų lygis yra pernelyg žemas, kad naudojant geriausias žinomas atakas, sistema būtų įveikta.“ – tas pats konspektų apie kriptologiją autorius apibrėžia saugumo skaičiavimo požiūriu sąvoką [Sta07]. Tiesa, kriptosistemos saugumo apibrėžime naudojama skaičių teorijos sąvoka galima būtų pakeisti į klaidas taisančių kodų sąvoka. Būtent tokiu principu remiasi šiame darbe tyrinėjamas objektas.

Kriptografijos saugumo lygį yra gana sunku įvertinti, todėl paprastai yra vertinama operacijų skaičius reikalingas įveikti kriptosistemą šiuo metu geriausiais žinomais algoritmais [MOV96]. Tad kiekvienos kriptosistemos saugumo lygis matuojamas jau žinomais jos įveikimo algoritmais. Logiška yra tai, jog atsiradus dar geresniam kriptosistemos įveikimo algoritmui, jos saugumas sumažėja, o nuo sumažėjusio lygmens yra sprendžiama ar ji yra vis dar pakankamai patikima ir tinkama plačiam naudojimui informacinių technologijų pasaulyje.

### 2.1.2. Matematinės problemos

Didelių skaičių faktorizavimu vadinama dviejų pirminių daugiklių radimu. Yra teigiama, jog neegzistuoja efektyvus algoritmas galintis pakankamai greitai faktorizuoti didelius skaičius, nors iš kitos pusės, nėra įrodyta, jog toks algoritmas neegzistuoja. Tad sistemos paremtos šia matematine problema gali vadintis saugios tol, kol nėra surastas algoritmas sprendžiantis pakankamai greitai faktorizavimo problemą [Pom94].

Diskretaus logaritmo problemos sąvoka lengviausia suprasti žinant diskretų eksponentavimą (ang. *discrete exponentiation*). Tarkime turime sveikų skaičių ciklinę aibę iš *n* elementų, tuomet jei norime surasti bet kurio aibės elemento k-tąjį laipsnį, pakeliame jį k-tuoju laipsniu bei apskaičiuojame liekaną likusią po dalybos iš aibės elementų skaičiaus (*n*) [Stu02]. Štai pavyzdys:

tarkime $n = 19$, $k = 4$,

tuomet $3^4 = 81$, padaliname *81* iš *19*

ir gauname liekaną lygią *5*.

Tuo tarptu diskretaus logaritmo uždavinys sprendžia atvirkščią problemą – žinant liekaną bei skaičių *n* yra ieškomas laipsnio rodiklis *k*. Diskretus eksponentavimas natūraliųjų skaičių aibėje gali būti skaičiuojamas labai greitai, pritaikius greitojo eksponentavimo metodą operacijų skaičius tesiektų $O(\log x)$, tačiau skaičiuoti diskrečius logaritmus yra kur kas sunkiau. Rastoje literatūroje [Stu02] [Sta07] yra nagrinėjami metodai šiai matematinei problemai spręsti.

Tiesinių kodų dekodavimo algoritmas, kuris paprastai naudojamas duomenų perdavimui nesaugiais kanalais, yra plačiai nagrinėjamas literatūroje [Ske05], tad teorinių žinių juos nagrinėjant pakanka.

Elipsinių kreivių problema taip pat plačiai tyrinėjama literatūroje [Bla99], aptariami teoriniai aspektai kriptografijos moksle, taip pat pritaikymai praktikoje. Verta pastebėti tai, jog egzistuoja kriptografinis algoritmas tokiu pačiu pavadinimu, kuris remiasi būtent šia matematine problema ir sėkmingai taikomas praktikoje.

### 2.1.3. Pagrindinės kriptografinės sistemos

Kriptografinės sistemos skirstomos į viešojo rakto bei simetrinių raktų sistemas. Šių sistemų apibrėžimus nesunkiai galima rasti lietuviškoje literatūroje: „...šifravimui ir dešifravimui naudojami skirtingi raktai <...> Tokios sistemos vadinamos asimetrinėmis arba viešojo rakto (*public-key*) sistemomis.“ [Sta07]. Simetrinio rakto kriptografijoje duomenys šifruojami ir dešifruojami tuo pačiu raktu. „Simetrinėmis vadinsime ir tokias sistemas, kuriose dešifravimo raktas nesutampa su šifravimo raktu, tačiau gali būti iš jo nesunkiai surandamas.“ [Sta07]. Paprastai simetrinio šifravimo realizacijos yra žymiai spartesnės nei viešojo rakto dėl paprastumo, tačiau simetrinio šifravimo didžiausiu trūkumu yra laikomas rakto perdavimas duomenų gavėjui. Simetrinio šifro kriptosistemos skirstomos pagal savo pobūdį į srauto bei blokines. Vėliau aptarsime keletą kriptosistemų tiek iš vienos grupės, tiek ir iš kitos.

Viena populiariausių šiuo metu kriptografinė viešojo rakto sistema yra laikoma RSA (pavadinimas kilęs nuo kūrėjų pavardžių pirmųjų raidžių). Ši kriptografinė sistema buvo išrasta 1977 metais, tad jos ankstyvas atsiradimas padėjo gerus pamatus jos taikymui praktikoje. Šios sistemos saugumas remiasi matematine faktorizavimo problema.

Gana plačiai yra taikomos ir simetrinės kriptosistemos (blokinės) kaip DES (ang. *Data Encryption Standard)* [Cop94] ar vėlesnis tų pačių užsakovų iniciatyva atsiradęs AES (ang. *Advanced Encryption Standard*). „DES pagrindas – Feistelio iteracijos, kuriose naudojamos atitinkamai sukonstruotos funkcijos.“ [Sta07]. AES kriptosistema yra labai detaliai aprašoma oficialiame JAV vyriausybės pranešime [AES01]. Minėtoje publikacijoje yra aprašomas standarto taikymo sritys, teisiniai panaudojimo aspektai, reikalingumo pagrindimas ir pan. Taip pat galima rasti informacijos susijusios su šios kriptosistemos realizaciją, jos saugumo pagrindimu bei žinomiausiomis ir pavojingiausiomis atakomis, apsisaugojimo nuo jų būdais.

Viena iš plačiau paplitusių programinės įrangos realizacijose yra RC4 (simetrinė srauto) (ang. *Rivest Cipher 4*) kriptografinis algoritmas. RC4 dar žinomas kaip ARC4 arba ARCFOUR. Algoritmas buvo sukurtas dar 1987 metais ir iki 1994 metų buvo laikomas paslaptyje, tačiau per neapdairumą gana greitai pasklido Internete algoritmo realizacija. Nuo to laiko yra leidžiama realizuoti šį algoritmą, tačiau jo vardo naudoti nederėtų naudoti be kūrėjų sutikimo. Žinomiausi bei labiausiai paplitę protokolai naudojantys šį algoritmą yra TLS/SSL, Torrent (bylų apsikeitimo paslauga), WEP (skirtas belaidžiui ryšiui), WPA (pagerintas WEP protokolas) ir pan.

### 2.1.4. Atakų tipai

Kriptografinių sistemų atakų kūrimas vadinamas kriptoanalize. Atakos apibrėžimas literatūroje skamba taip: “Šifro kriptoanalizė, t.y. bandymas atkurti pranešimą be rakto, vadinama ataka.” [Sta07]. Kriptografinių sistemų atakų tipų yra labai įvairių, kad ir bandant atspėti vartotojo slaptažodį:

- bandymai atspėti dažnai pasitaikančius slaptažodžius;
- visų galimų variantų perrinkimas;
- bandymas atspėti slaptažodį žinant jo dalį;
- jungtinių žodžių kombinacijų perrinkimas;
- bandymai atspėti slaptažodžius remiantis tam tikromis taisyklėmis, kaip kai kurių raidžių rašymas iš didžiųjų raidžių ar slaptažodžio gale prirašant vienetą ir pan.;
- naudojant esamą slaptažodžių duomenų bazę.

Tarkime labiau patyrę vartotojai naudoja atsitiktinai sugeneruotus slaptažodžius, kurių atspėti ankščiau išvardintais metodais praktiškai neįmanoma. Tuomet belieka kriptoanalizės specialistams gilintis į kriptografinių sistemų sandarą, ieškoti silpnų vietų ir taip tikėtis sėkmės – pažeisti vienaip ar kitaip sistemą.

Atakos skirstomos tipais pagal tai, kokią informaciją atakuotojas turi. Jei kriptoanalitikas sugebėjo gauti tik užšifruotą pranešimą, tai jo rengiama ataka priklauso pavienių šifrų atakų grupei (ang. *ciphertext-only*). Jei atakuotojas sugebėjo gauti tiek užšifruotus duomenis, tiek ir dešifruotus, tuomet jo rengiama ataka priklauso teksto-šifro porų grupei (ang. *known-plaintext*). Jei kriptoanalitikas turi galimybę realiu laiku pateikti norimus tekstinius pranešimus šifravimo sistemai ir gauti užšifruotus pranešimus, tuomet ataka priklausytų pasirinktų teksto – šifro porų atakų grupei (ang. *chosen-plaintext*). Jei atakuotojui pasiseka dar labiau, t.y. gauti priėjimą prie sistemos tokį, kad galėtų siųsti prašymus užšifruoti pakartotinai, tai tokia ataka priklausytų adaptyvių pasirinktų teksto – šifro porų atakoms (ang. *adaptive-chosen-plaintext*). Aplaidžiausia kriptosistemos kūrėjų klaida būtų laikoma, jei būtų įmanoma įvykdyti pasirinktų šifrų ataką (ang. *chosen-ciphertext),* kuomet atakuotojas pasirenka šifrus ir gauna juos atitinkančius užšifruotus pranešimus [Sta07] [Con07]. Detalesnė atakų analizė bus pateikiama kitame skyriuje, nagrinėjant McEliece kriptosistemai aktualias atakas.

### 2.1.5. Klaidas taisančių kodų taikymas kriptografijoje

Klaidas taisančių kodų teorija literatūroje apibrėžiama taip: “Klaidas taisančių kodų teorija <..> kuria praktinius kodavimo ir dekodavimo būdus.” [Ske05]. Pats šios teorijos pavadinimas sufleruoja tai, jog užšifruota informacija persiųsta nesaugiu kanalu, kur gali būti iškraipyta dėl techninių priežasčių, sėkmingai atšifruojama. Norint pasiekti šį rezultatą prie pradinio pranešimo pridedama papildoma informacija, kuri vėliau padeda dešifruoti pradinį pranešimą. Tiesa, pradinis pranešimas šiek tiek padidėja. Klaidas taisantiems kodams yra priskiriami pakartojimo, pakartojimo n kartų, kontrolinio simbolio ( knygos ISBN, Lietuvos piliečių asmens kodas ir pan.), Hemingo kodai ir t.t. Verta dėmesio viena įdomybė rasta literatūroje: „ ... dėl kompaktinėse plokštelėse naudojamų klaidas taisančių kodų muzikos kokybė nenukentėtų net ir pradūrus joje 2mm skersmens skylę!” [DS03].

Iš tikrųjų, rasti tinkamos literatūros, skirtos klaidas taisančių kodų taikymui kriptografinėse sistemose, nepavyko. Dauguma straipsnių ar metodinių priemonių šias dvi iš pažiūros skirtingas sferas linkę atskirti, kartais net pabrėžiant skaitytojui, jog šių sąvokų nevalia painioti: „Kodavimo teorija skirta informacijos skaitymui palengvinti: nereikia sumaišyti su kriptografija, kur informacija yra užšifruojama taip, jog būtų labai sunku dešifruoti originalų pranešimą!” [Pin97]. Belieka tikėtis, jog nagrinėjant konkrečias kriptosistemas, naudojančias klaidas taisančius kodus, bus išsamiai aprašyta ir paaiškinta jų veikimo specifika.

### 2.1.6. Kriptosistemos naudojančios tiesinius kodus

Egzistuoja dvi plačiausiai taikomos kriptosistemos naudojančios tiesinius klaidas taisančius kodus: McEliece bei Niederreiter [Can98]. Jų saugumo lygį ekspertai vadina panašiu ar net tokiu pačiu. Yra ir daugiau kriptosistemų naudojančių klaidas taisančius kodus, tačiau dėl didelio tarpusavio panašumo neverta jų nagrinėti. Beje, yra nemažai McEliece kriptografinės sistemos modifikacijų, kurios šiek tiek lenkia saugumu savo pirmtakę. Galima paminėti Gabidulin kriptosistemą naudojančią tiesinius kodus, tačiau yra teigiama, jog ji neturi jokių privalumų prieš McEliece sistemą. „Gibson įrodė, jog nėra jokių privalumų naudojant Gabidulin sistemą lyginant su McEliece.“ [LM99], čia minima australų mokslininkės Keith Gibson pavardė.

## 2.2. McEliece ir Niederreiter kriptografinių sistemų veikimas

### 2.2.1. McEliece kriptografinė sistema

Viešojo rakto kriptosistemos panaudojimui yra būtini trys sudėtiniai žingsniai: privataus bei viešojo raktų generavimas, informacijos pranešimo užšifravimas bei informacijos pranešimo dešifravimas.

Privataus ir viešojo rakto generavimo algoritmas įvairioje literatūroje yra identiškas:

- pasirenkama sveikų skaičių pora *n, k*. Iš jų pagamintas tiesinis kodas *C* yra įgalus ištaisyti *t* arba mažiau klaidų;
- siunčiantysis sugeneruoja *k*x*n* generuojančią matricą *G*;
- sugeneruoja atsitiktinę, nevienetinę *k*x*k* matricą *S*;
- sugeneruoja perstatos (ang. *permutation*) matricą iš *n* elementų *P*;
- sugeneruojama matrica, kuri sudaro viešąjį raktą – $G^* = SGP$;
- taigi, viešasis raktas – ($G^*$, *t*), o privatus – (*S*, *G*, *P*).

Bandymas užšifruoti pranešimą viešuoju raktu turėtų atrodyti taip:

- pranešimu vadinkime – *m* (dvejetainio formato pavidalo);
- suskaičiuojame mG*;
- pridedame klaidos vektorių *e* ir gauname *c*;
- *c* - užšifruotas pranešimas.

Pranešimo dešifravimo algoritmas:

- suskaičiuojamas $cP^{-1} = mSG + e'$;
- suskaičiuojame $mS = [G^T|(mS)^T]$, o iš čia
- gauname *m*, siųstą pranešimą.

### 2.2.2. Niederreiter kriptografinė sistema

Ši kriptografinė sistema yra labai panaši į ankščiau aprašytą McEliece sistemą. Šiuo atveju nėra naudojama generuojanti matrica, o kontrolinė (ang. *parity check*) matrica. Sistemos veikimui reikalingi trys sudėtiniai žingsniai: viešojo bei slaptojo raktų generavimas, pranešimo šifravimas bei pranešimo dešifravimas.

Raktų poros generavimo algoritmas:

- pasirenkame atsitiktinį polinomą, kurį pažymėkime $g(z)$ ir kurio laipsnis yra t (maksimalus galimas klaidų skaičius);
- suskaičiuojame ($n$-$k$) x $n$ matricą $G$;
- pasirenkame atsitiktinai sugeneruotą nevienetinę matricą ($n$-$k$)x($n$-$k$) $S$;
- pasirenkame atsitiktinai sugeneruotą matricą $P$ tam, kad
- suskaičiuoti $G^* = SHP$;
- taigi, viešasis raktas – ($G^*$, $t$), o privatus – ($S$, $G$, $P$).

Bandymas užšifruoti pranešima viešuoju raktu turėtų atrodyti taip:

- pradinis pranešimas paverčiamas $n$ ilgio dvejetaine seka, kurios svoris lygus $t$;
- suskaičiuojame $c = m(G^*)^T$;
- $c$ - užšifruotas pranešimas.

Pranešimo dešifravimo algoritmas:

- apskaičiuojame $P^{-1}$ bei $(S^T)^{-1}$;
- apskaičiuojame $c' = c\,(S^T)^{-1}$;
- dešifruojame $m'$ iš $c'$ remdamiesi dešifravimo algoritmu;
- apskaičiavę $m = m'P^{-1}$ bei atkodavę pranešimą iš dvejetainės skaičiavimo sistemos gauname originalų pranešimą.

### 2.2.3. Goppa kodai bei Goppa polinomai

Goppa kodų bei Goppa polinomų pavadinimas kilęs nuo jų kūrėjo pavardės. V.G. Goppa – rusų matematikas, sukūręs klaidas taisančių kodų klasę apie 1978 metus. Iš tikrųjų, Goppa kodai yra BCH kodų generalizacija, tačiau pastarųjų saugumas yra nepakankamas saugioms kriptosistemoms [Joc02]. Tad būtent šia, klaidas taisančių kodų klase, pasinaudojo Robert J. McEliece ir sukūrė tiriamąją kriptosistemą.

Goppa kodų sudėtis yra plačiai aprašoma rastoje literatūroje [Joc02]. O konkrečios, Goppa, klaidas taisančių kodų klasės įtaka kriptografinių sistemų saugumui literatūroje minima labai miglotai. Tarkime jei užšifruotą pranešimą pažymėtume c, tuomet šiek tiek modifikuotą (pakeistus kelis simbolius) pranešimą pažymėkime c'. Yra šiokia tokia tikimybė, jog Goppa kodo pagalba bus dešifruotas pranešimas. Tad kriptoanalitikui atlikus nemažą skaičių modifikacijų galiausiai pavyks nusiųsti modifikuotą pranešimą, kuris sėkmingai sistemos bus dešifruotas. Tačiau klaidos vektorius nebesutaps su originaliuoju ir sistema turėtų būtent dėl šios priežasties nepateikti dešifruoto pranešimo [Sun00].

Literatūroje pažymėta, jog buvo atliktas bandymas McEliece kriptosistemoje Goppa kodus pakeisti kita, klaidas taisančių kodų klase – MRD (ang. *maximum rank distance*), tačiau anot tyrėjo sukurtoji schema pasirodė nepakankamai saugi plačiam taikymui [Ber97].

### 2.2.4. Matricos S ir P

S ir P sudaro esminį McEliece kriptosistemos saugumą. Jei kriptoanalitikas sugebėtų atspėti šias matricas, apskaičiuotų generuojančią matricą G, o vėliau ir polinomą g(z). Panagrinėkime S ir P matricas atskirai. Kaip žinia, matrica S sudarytą iš *kxk* elementų, o galimų variantų pirmajai eilutei sudaryti yra $2^k - 1$. Antroje eilutėje variantų sumažėja per vienetą, tad galimybių išdėstyti belieka $2^k - 2$ ir t.t. Taigi, S matricos variantų yra:

$$(2^{524} - 2^0) + (2^{524} - 2^1) + \ldots + (2^{524} - 2^{523}).$$

O tikimybė atspėti vienu kartu S matrica yra apytiksliai lygi $0{,}8459238718 \cdot 10^{-82655}$.

Tikimybė atspėti P matricą yra gerokai didesnė lyginant su matrica S, tačiau taip pat labai maža:

$$1/1024! = 0{,}1845519398 \cdot 10^{-2639}.$$

Iš pateiktų skaičių matyti, jog kriptografinės sistemos įveikimas atspėjant S ir P matricas yra praktiškai neįmanomas [Joc02].

### 2.2.5. Klaidos vektorius

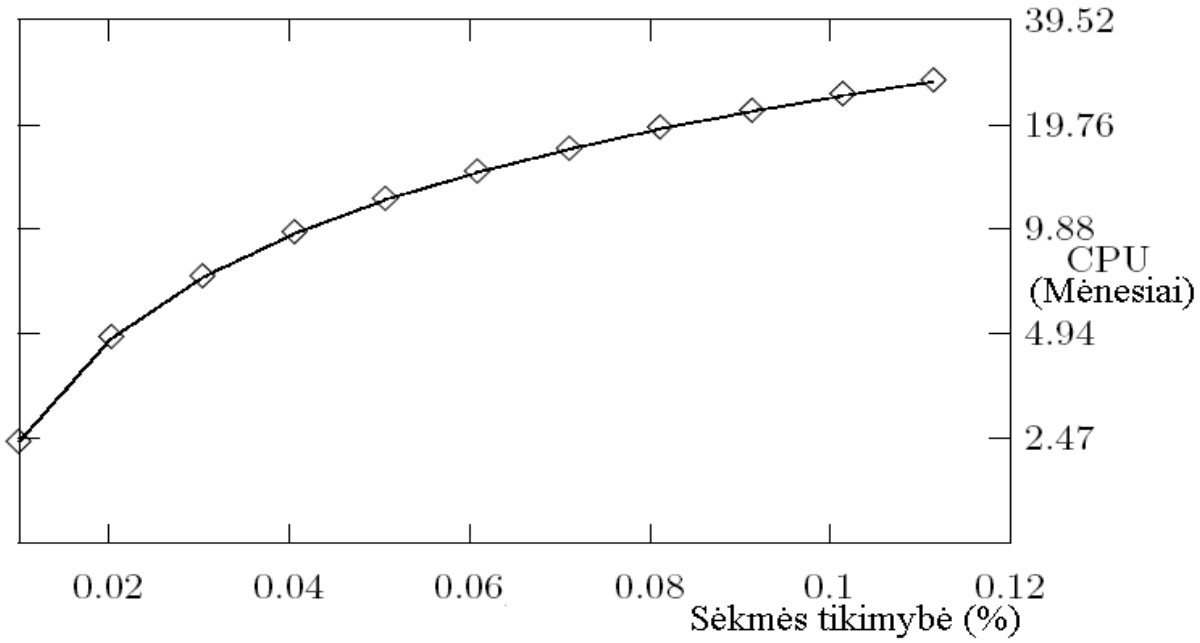
Klaidos vektoriaus pagalba yra užtikrinamas siuntėjo atpažinimas. Gavus netinkamą klaidos vektorių kriptosistema ne dešifruojama. Verta pažymėti, jog dažnoje publikacijoje, susijusioje su klaidos vektoriaus naudojimu kriptografijoje, atkreipiamas dėmesys į tai, jog naudojant tą patį raktą šifruojant pranešimą ir skirtingus klaidos vektorius iškyla nemaža grėsmė, jog sistema bus pažeidžiama [Ber97] [LM99]. Geriausias sprendimas tokiu atveju yra keisti kiekvieno pranešimo raktą unikaliu, blogesnis sprendimas būtų nekeisti klaidos vektoriaus, o naudoti tą patį kiekvieną kartą. Pastaruoju atveju, kriptoanalitikas turės ribotas galimybės spręsti apie privatų raktą. Šio tipo atakos yra pavadintos Berson vardu ir bus plačiau panagrinėjama šiame darbe vėliau.

### 2.2.6. Kriptosistemos silpnosios vietos

Kriptoanalitiko T. Bersono publikacijoje skelbiama, jog McEliece kriptosistema yra pažeidžiama dviem būdais: pranešimo persiuntimo (ang. *message-resend*) ir susijusių pranešimo (ang. r*elated-message*) atakomis. Pasak autoriaus, pasinaudojus šiomis skylėmis dešifravimas palengvėja apie $10^{15}$ kartų [Ber97]. Taip pat pažymima, jog šios atakos tokios pats grėsmingos ir kitoms viešojo rakto kriptosistemoms naudojančioms klaidas taisančius kodus.

S ir P matricų atspėjimą, t.y. priartėjimą prie privataus rakto, laikyti silpnąja vieta nederėtų, nes ankščiau aprašyti skaičiavimai rodo, jog, kad tai įvyktų yra labai maža tikimybė.

Viena iš galimų silpnųjų kriptosistemos vietų galėtų būti kriptoanalitiko sužinojimas dalies (kelių bitų) originalaus pranešimo. Tai leidžia atitinkamai sumažinti tyrimo amplitudę. Literatūros autorių atlikti bandymai bei teoriniai skaičiavimai leido padaryti kriptografinės sistemos įveikimo tikimybės augimo priklausomybę nuo atlikto skaičiuojamojo darbo kompiuteriu:




**Paveikslėlis 1**

Šis tyrimas buvo atliktas 1998 metais su dešimčia lygiagrečiai veikiančių kompiuterių, kurių kiekvieno darbo dažnis siekė apie 433 MHz [CS98]. Žinant gerokai padidėjusias techninės įrangos galimybes, šiuo metu tą patį rezultatą galima atlikti su vienu asmeniniu kompiuteriu be didesnių pastangų. Tačiau sujungus šiuolaikinių kompiuterių išteklius ar pasinaudojus viešai prieinamais kompiuterių tinklais bei darbą išlygiagretinus MPI priemonėmis, rezultato galima tikėtis kur kas įdomesnio.

### 2.2.7. Silpnų raktų išvengimas

Silpnų raktų išvengimo problemai yra skirta Pierre Loidreau ir Nicolas Sendrier publikacija [LS00]. Joje pastebėta, kaip ir daugelyje panašių leidinių, McEliece kriptosistemos saugumas paremtas dviem aspektais: viešojo rakto dydis yra pakankamas, kad negalima būtų dešifruoti bei susiduriama su rimta matematine problema dešifruojant sistemą žinant tik viešąjį raktą. Ankščiau minėtame straipsnyje pateikiami silpnų raktų aptikimo algoritmai bei sistemų su silpnais raktais dešifravimo žingsniai. Kituose literatūros šaltiniuose pavyko rasti tokias rekomendacijas:

- kodo dydis turi būti pakankamai didelis, kad liktų negalima paprasto perrinkimo ataka (ang. *brute force*);

- generuojanti matrica neturi teikti charakteristikos apie kodą.

## 2.3. Atakos prieš McEliece kriptosistemą

### 2.3.1. Plačiausiai žinomos atakos prieš McEliece

Atakos skirstomos į kritines bei nekritines pagal jų pavojingumą kriptografinės sistemos saugumui.

Nekritinių atakų išvengti paprasčiausia yra didinant kriptosistemos parametrų dydį arba pritaikant McEliece kriptosistemos Loidreau modifikaciją. Nekritinių atakų sąrašas:

- mažo svorio kodo žodžių radimas (ang. finding-low-weight-codeword);
- apibendrintos informacijos dekodavimas (ang. generalized information-set-decoding) ataka;

Kritinėmis atakomis vadinamos šios:

- žinomo dalinio teksto (ang. *known-partial-plaintext*) ataka;
- susijusių pranešimų ataka;
- Korzhik-Turkin ataka;
- gavėjo atsako (ang. *reaction*) ataka;
- lankstumo (ang. *malleability*) ataka;
- techninės įrangos klaidos ataka.

Šį atakų klasifikavimą ir pačias atakas su jų aprašais galima rasti japonų mokslininkų Kazukuni Kobara ir Hideki Imai publikacijoje [KI00]. Norint detaliau panagrinėti kiekvieną iš atakų galima rasti tam skirtos literatūros, pavyzdžiui, susijusių pranešimų ataka pakankamai išsamiai nagrinėta amerikiečio T. A. Berson [Ber97], o Korzhik-Turkin atakos oficialus pristatymas kriptologijos pasauliui įvyko 1991 metais autorių pranešimu [KT91]. Yra nedidelė, tačiau įmanoma techninės įrangos klaidos tikimybė, tuo pačiu atsiranda galimybė ja pasinaudoti kriptosistemai įveikti. Apie tokio tipo ataką detalesnės informacijos reiktų ieškoti [BDL98] publikacijoje.

Verta dėmesio daktaro disertacija parašyta Orr Dunkelman 2006 metais. Joje labai plačiai aprašomi atakų tipai bei pačios atakos prieš įvairiausias kriptosistemas, tad atrinkus tinkama informaciją, t.y. susijusią su viešojo rakto McEliece kriptosistema, galima kuo puikiausiai pasinaudoti sukoncentruota informacija, juolab, kad žinios yra nepasenusios [Dun06].

### 2.3.2. Atakų realizacija

Pilnai aprašytų atakų realizacijų rasti nepavyko, tačiau rastoje vokiečių mokslininkų publikacijoje [EOS06] pateikiamos populiariausių viešojo rakto atakų aprašai, analizė bei algoritmai. Algoritmai pateikti subkodu, o tai labai palengvina jų supratimą bei padeda realizuoti atakas prieš McEliece kriptografinę sistemą. Algoritmai (šiuo atveju – *GISD* ataka) pateikiami tokia notacija:

*while true do*

atsitiktinai pasirenkamas I iš intervalo {0, ..., n-1}, kur | I | = k.

$Q_1 = G^{-1}$; $Q_2 = Q_1 G$

$z = c + c_I Q_2$

*for i = 0 to j do*

*for all* $e_I$ *with* $wt(e_I) = i$ *do*

*if* $wt(z + e_I Q_2) = t$ *then*

*return* $((c_I + e_I)Q_1)$

Surinktos informacijos apie atakų realizaciją turėtų pakakti, beliks pasirinkti kurias ir kiek iš jų realizuoti bei remiantis jų rezultatais padaryti atitinkamas išvadas.

# 3. Tiriamosios atakos prieš McEliece kriptosistemą

## 3.1. Apžvalga

Visų pirma, remiantis anksčiau atlikta literatūros apžvalga, bus atliekama atakų analizė. Išnagrinėjus kiekvienos atakos realizacinius klausimus bus pasirinkta keleta atakų ir jos įgyvendintos. Reiktų nepamiršti paminėti, jog pačios kriptosistemos realizaciją reiks atlikti taip pat. Mat pademonstruoti šifravimą dokumente, kuomet parametrai (*n, k, t*) yra pakankamai dideli yra praktiškai neįmanoma dėl skaičiavimų apimties. Kalbant apie realizacija, nėra skirtumo rezultatams kokia programavimo kalba tai bus atlikta, nes realizuojant kiekvieną ataką ta pačia programavimo kalba, nė viena ataka neįgauna pranašumo prieš kitą vien dėl technologijos. Be to, skirtingomis technologijomis realizuotų algoritmų vykdymo laikas skiriasi nežymiai (nuo kelių iki keliolikos procentų), tad absoliutiems skaičiavimo rezultatams tai didelės įtakos neturės. Todėl dėl patirties bei turimų žinių yra pasirinkta Java programavimo kalba.

Vienas iš darbo uždavinių yra įvykdyti realizuotas atakas. Suprantama, jog su pakankamai saugiais parametrais (*n, k, t*) to padaryti nepavyks, todėl dėl riboto laiko bus rasti tokie parametrai, su kuriais sistemos galimas nulaužimas truktų priimtiną laiką (mažiau nei para su 1,6 GHz procesoriaus kompiuteriu). Norint statistiškai įvertinti atakos rezultatus galima būtų jos realizaciją modifikuoti taip, jog programos vykdymo metu būtų renkama statistinė medžiaga apie operacijų atlikimo trukmę. Tačiau statistikos rinkimas, t.y. informacijos rašymas į bylą užtrunka šiek tiek laiko, o žinant, jog operacijų skaičius yra milžiniškas, tai statistikos rinkimas trukdytų efektyviam programos, realizuojančios atakos algoritmą, darbui. Dėl šios priežasties statistika bus renkama pildant panašią į žemiau pateiktą lentelę:

| Parametras *n* | Parametras *k* | Parametras *t* | Operacijų skaičius | Vykdymo laikas |
|---|---|---|---|---|
| | | | | |

Surinktą statistika reiks įvertinti bei pamėginti nuspėti, koks atakos vykdymo laikas būtų su realaus pasaulio uždaviniuose naudojamuose parametruose (*n, k t*), kuomet $n \geq 1024, k \geq 524, t \geq 50$. Gautą spėjimą reiktų patikrinti su literatūroje rastais atakų vykdymo laikais, aišku, jei tokie duomenys egzistuoja. Jei rasti bei gauti rezultatai sutaptų, galima būtų daryti išvada, jog ataka realizuota sėkmingai, priešingu atveju tektų ieškoti klaidų realizacijoje arba optimizuoti atakos veikimą.

Pažymėsime, jog techninės įrangos klaidos atakos realizacija turėtų būti komplikuota bei sunku būtų prognozuoti kiek jos realizacija bei vykdymas galėtų užtrukti, todėl darbe ši ataka detaliau nebus nagrinėjama. Galima pažymėti, jog yra tokia tikimybė, kad įvyktų techninės įrangos klaida, tuo pačiu atsirastų ir galimybė ja pasinaudoti kriptosistemai įveikti. Apie tokio tipo ataką detalesnės informacijos reiktų ieškoti [BDL98] paskutiniajame skyriuje pateiktoje literatūroje.

Egzistuoja atakų skirstymas pagal atakuojamo objekto tipą. Didžiausio dėmesio susilaukia atakos prieš viešąjį raktą, kuomet iš turimo viešojo rakto bandoma gauti slaptąjį raktą bei atakos, kuomet turimas užšifruotas pranešimas ir bandoma rasti originalųjį pranešimą. Pastaroji atakų grupė susilaukia didesnio dėmesio iš kriptoanalitikų, mat gerai žinoma, jog naudojant tą pačią generuojančią matricą bei skirtingus klaidos vektorius užšifruojant tą patį pranešimą yra lengva dešifruoti slaptą informaciją, o tai reiškia sėkmingai įvykdyti ataką prieš McEliece kriptografinę sistemą. Šia ataka bandysime pasinaudoti šio semestro darbe.

Nagrinėjant kriptografinių sistemų saugumą verta panagrinėti trivialią visų variantų perrinkimo (ang. *brute force*) ataką. Norint atlikti tokią ataką reikia perrinkti visas kėlinių kombinacijas, Goppa kodų variantus bei visas invertuojamas matricas. Kiekvieno iš šių veiksmų galimų variantų skaičius pateikiamas žemiau esančioje lentelėje:

| **Operacijos pavadinimas** | **Galimų variantų skaičius** |
|---|---|
| Kėlinių perrinkimas | n! |
| Goppa kodų generavimas | $\frac{2^{mt}}{t}$ |
| Invertuojamų matricų generavimas | $0{,}29 \times 2^{k^2}$ |

Manant, jog realaus pasaulio uždaviniuose parametrai (*n, k, t*) yra pakankamai dideli $(n \geq 1024, k \geq 524, t \geq 50)$ ši ataka nėra realiai įvykdoma. Tačiau galima bus pamėginti ją atlikti su mažesniais parametrais bei paprognozuoti, kiek laiko truktų jos vykdymas su rekomenduojamo dydžio parametrais.

## 3.2. Pavyzdinio pranešimo užšifravimas

Pranešimo užšifravimui naudosime tiriamąją kriptosistemos algoritmą. Pademonstravimui naudosime elementarų pavyzdį, mat atlikti analogiškus veiksmus su realaus pasaulio duomenimis būtų pernelyg sudėtinga.

Tarkime, jog norime užšifruoti žodį „ė”:

- pasirinktąjį žodį konvertuojame į dvejetainį kodą:

| **Šifruojama raidė** | **Šešioliktainis kodas** | **Dvejetainis kodas** |
|---|---|---|
| ė | EB | 1110 1011 |

Tad dvejetaine išraiška šifruojamas žodis atrodo taip: $m = (1\ \ 1\ \ 1\ \ 0\ \ 1\ \ 0\ \ 1\ \ 1)$.

- pasinaudodami Goppa kodu šifruojame dvejetainio pavidalo pradinį pranešimą:

$y = mG^* + e$, kur $y$ – užšifruotas pranešimas, $m$ – originalusis pranešimas, $G^*$ - kodą generuojanti matrica, $e$ – klaidos vektorius.

$G^*$ gavimas nėra trivialus, tad aptarsime šios matricos gavimo algoritmą.

Pagal apibrėžimą $G^* = SGP$, kur $S$ – atsitiktinai pasirinkta neišsigimusi $k \times k$ matrica, $P$ – perstatos $n \times n$ matrica, $G$ – kodą generuojanti $k \times n$ matrica. Paprastumo sumetimais šiame darbe nebus atliktas pilnas generuojančios matricos išvedimas iš polinomo. Pradiniais duomenimis laikysime literatūroje aprašytą generuojančią ($G$), perstatos ($P$) bei $S$ matricas. Tuomet,

$$G = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$S = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

$$P = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Pasinaudojus anksčiau aprašyta formule gauname *G**:

$$G^* = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \end{pmatrix}$$

Akivaizdu yra tai, jog tiek *G* matricos tiek ir *G** matricos rangas nepakito. Tokiu būdu paslėpėme generuojančią matricą *G* ir toliau galime atlikti šifravimo žingsnius.

Apskaičiuojame *mG**:

$$\begin{pmatrix}1 & 1 & 1 & 0 & 1 & 0 & 1 & 1\end{pmatrix}\begin{pmatrix}0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1\\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0\\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0\\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1\\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1\\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1\\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0\\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0\end{pmatrix} =$$

$$\begin{pmatrix}1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0\end{pmatrix}$$

Galiausiai pridėję klaidos vektorių *e* gausime galutinį užšifruotą pranešimą. Kaip žinia, klaidos vektorius parenkamas atsitiktiniu būdu. Šiuo atveju jame gali būti nedaugiau nei dvi klaidos, t.y. dviejose pozicijose gali būti vienetukai, o likusiose turi likti nuliai. Tarę, jog klaidos vektorius yra $e_1 = \begin{pmatrix}0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\end{pmatrix}$ gauname užšifruotą pranešimą:

$$y_1 = \begin{pmatrix}1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0\end{pmatrix}.$$

Kitame skyrelyje bandysime šį užšifruotą pranešimą dešifruoti susijusių pranešimų ataka. Žinant, jog tokio tipo atakai įvykdyti reikia turėti to pačio pranešimo antrą šifrą, bet iškraipyto kitu klaidos vektoriumi, suskaičiuojame jį, tardami, jog naujasis klaidos vektorius:

$$e_2 = \begin{pmatrix}0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0\end{pmatrix}$$

tuomet apskaičiuojame šifrą:

$y_2 = \begin{pmatrix}1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0\end{pmatrix}$

## 3.3. Susijusių pranešimų atakos algoritmas

Šiame darbe bandysime dešifruoti ankstesniame skyrelyje užšifruotą pranešimą pasitelkdami anksčiau minėtos atakos algoritmą. Ši ataka vadinama susijusių pranešimų ataka [Ber97]. Sėkmingai įvykdytos atakos rezultatu yra laikomas dešifruotas pranešimas, tačiau privataus rakto atsekti nepavyks. Šios atakos idėja remiasi tuo, jog jei yra užšifruojamas pranešimas *m* ta pačia generuojančia matrica *G** su skirtingais klaidų vektoriais yra gaunamas skirtingas šifras. Anot nagrinėtos literatūros galima apytiksliai apskaičiuoti iš kelių bandymų ir per kiek laiko (*w*) gali pavykti įvykdyti šią ataką:

$$w = \alpha * \frac{\binom{n}{k}}{\binom{n-t}{k}}$$

čia $t$ – maksimalus klaidų skaičius galintis būti klaidos vektoriuje, α – laiko tarpas, per kurį yra invertuojama $k \times k$ matrica. Jei nagrinėtume realaus pasaulio uždavinį, kuomet $k$ ir $n$ parametrai yra pakankamai dideli, o dėl jų padidėjęs ir $t$, t.y. $k = 524$, $n = 1024$, o $t = 50$, tuomet operacijų reikalingų nustatyti pradinį pranešimą reiktų:

$$\frac{\binom{1024}{524}}{\binom{974-50}{524}} \approx 1{,}37 \times 10^{16}$$

Konkrečiu atveju mums tereikės atlikti keletą spėjimų:

$$\frac{\binom{16}{8}}{\binom{16-2}{8}} \approx 4{,}29$$

Aptarkime šios atakos algoritmą. Tarkime, jog gavome du šifrus $c_1$ ir $c_2$, kuriais yra užšifruotas tas pats pranešimas:

$$c_1 = mSGP + e_1 \text{ ir } c_2 = mSGP + e_2, \text{ o } e_1 \neq e_2.$$

Šiuo atveju persiuntimo gylis (ang. *resend depth*) yra 2, t.y. gauti du šifrai su užšifruotu tuo pačiu pranešimu. Įdomu tai, jog kuo didesnis persiuntimo gylis, tuo lengviau yra sužinoti originalų pranešimą. Nėra sudėtinga atsekti ar du šifrai slepia tą patį pranešimą. Tereikia nustatyti abiejų šifrų vektorių sumos Hamingo svorį, t.y. vienetukų esančių pranešimų sumoje skaičių. Paprastai, realaus pasaulio uždaviniuose, tų pačių pranešimų šifrų Hamingo svoris negali viršyti 100 (*t*2*), o skirtingų pranešimų yra apylygis 524 (*k*).

Mums reikia sudaryti dvi sekas $L_0$ ir $L_1$ iš šifrų sumos $(c_1 + c_2)$. Jų sudarymo principas yra labai paprastas: $L_0$ seka bus sudaryta iš pozicijų numerių, kuriose $(c_1 + c_2)$ yra nuliai, o $L_1$ seka bus sudaryta iš pozicijų numerių, kuriose $(c_1 + c_2)$ yra vienetai. Visa tai formaliai atrodytų taip:

$$L_0 = \{l \in \{1,2,...,1024\} : c_1(l) + c_2(l) = e_1(l) + e_2(l) = 0\}$$
$$L_1 = \{l \in \{1,2,...,1024\} : c_1(l) + c_2(l) = e_1(l) + e_2(l) = 1\}$$

Iš šių sekų išplaukia, jog $l \in L_0$ tikėtina, kad nei $c_1(l)$ nei $c_2(l)$ nėra pakeisti klaidos vektoriaus, tuo tarpu $l \in L_1$ privalomai $c_1(l)$ arba $c_2(l)$ yra pakeistas klaidos vektoriaus. Todėl $l \in L_0$ reiškia, jog $e_1(l) = e_2(l) = 0$ arba $e_1(l) = e_2(l) = 1$. Tariant, jog abu klaidos vektoriai buvo parinkti nepriklausomai vienas nuo kito, tuomet tikimybė su kiekvienu *l*, kad $e_1(l) = e_2(l) = 1$ yra apskaičiuojama taip:

$$\binom{t}{n}^2,$$

tariant, jog parametrai yra standartinio dydžio:

$$\binom{50}{1024}^2 \approx 0{,}0024,$$

o tai reiškia, kad dauguma $l \in L_0$ žymi $e_1(l) = e_2(l) = 0$. Tokiu būdu reikia atspėti *k* (524) neiškraipytas pozicijas iš $L_0$ sekos. Tarkime, kad *i* žymi klaidingo bito numerį, kur pranešimuose yra iškraipyta informacija abiejų klaidos vektorių. Aišku, jog $i \le t$. O tikimybė, jog abiejuose klaidų vektoriuose tose pačiose pozicijose yra *i*-tasis klaidingas bitas apskaičiuojamas taip:

$$p_i = \frac{\binom{t}{i}\binom{n-t}{t-i}}{\binom{n}{t}},$$

tariant, jog parametrai yra standartinio dydžio:

$$p_i = \frac{\binom{50}{i}\binom{974}{50-i}}{\binom{1024}{50}}$$

Realizuojant šį atakos algoritmą pradiniais duomenimis bus laikomi du šifrai, kurie slepia tą patį pranešimą užšifruotą tos pačios generuojančios matricos su vieno nuo kito nepriklausomais klaidos vektoriais. Atakos realizacijoje bus naudojami du ankstesniame skyrelyje gauti šifrai:

$$y_1 = (1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0),$$

$$y_2 = (1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0)$$

Pavykus sėkmingai dešifruoti pradinį pranešimą reiktų apskaičiuoti vidutinį bandymų, reikalingų dešifravimui, skaičių nepriklausomai nuo klaidos vektorių. Kiekvieno bandymo metu būtų generuojami atsitiktiniai klaidos vektoriai, kuriuose būtų iki dviejų iškraipytų bitų, t.y. galimos daugiausiai dvi nekorektiškos pozicijos iš šešiolikos. Apskaičiavus statistinį bandymų skaičių reikalingą šiai, gerokai supaprastintai, kriptografinei sistemai dešifruoti, galima bus palyginti su teoriškai apskaičiuota reikšme – 4,29.

Jei skaičiavimo metu gautos reikšmės bus apylygės galima bus teigti, jog skaičiavimai yra pakankamai tikslus. Be to, įvertinus vieno bandymo trukmę galima būtų prognozuoti, kiek laiko užtruktų dešifruoti kriptosistemą su pakankamai saugiais laikomais parametrais, kuomet $(n \geq 1024, k \geq 524, t \geq 50)$. Jei apskaičiuota laiko reikšmė būtų mažesnė nei viena para, galima būtų paeksperimentuoti vykdant tokią ataką. Tačiau tokiu atveju reiktų papildomai atlikti generuojančios matricos generavimo algoritmą programiškai, nes tokio didelio rango matricos skaičiavimus atlikti raštu praktiškai yra neįmanoma.

## 3.4. Susijusių pranešimų atakos realizacija

Atakos realizacijai buvo pasirinka Java programavimo kalba. Tokį pasirinkimą nulėmė programavimo kalbos populiarumas, turima asmeninė programavimo patirtis bei parašytos programos lengvas perkelimas į skirtingas platformas. Migravimas tarp skirtingu platformų yra svarbus dėl to, jog norint atlikti ataką, pasitelkus grupę kompiuterių bei naudojant MPI metodiką, jie gali veikti su skirtingomis operacinėmis sistemomis. Dideliu privalumu buvo laikoma objektinio programavimo galimybė Java kalba. Programuojant nebuvo naudojamos jokios su matricomis susijusios bibliotekos, transponavimo, daugybos ir panašūs veiksmai buvo realizuoti savarankiškai. Tokio sprendimo privalumu galima laikyti greitesni programos veikimą, bei lengvesnį integruojamumą į kitas sistemas

Sukurtoje programoje buvo realizuoti du esminiai metodai, kurių vienas apskaičiuodavo iš dviejų pateiktų šifrų, kurie yra iškraipyti klaidos vektoriaus, originalų šifrą. Tiesa, tai galima greičiau vadinti bandymu, nes nebūtinai rastasis šifras yra tikrasis. Antrasis metodas iš pateikto pirmojo metodo šifro dešifruodavo originalų pranešimą remiantis generuojančia matrica.

Pirmojo metodo logika susidarė iš dviejų detalių. Visų pirma apskaičiuojamas Hamingo atstumas tarp dviejų vektorių, kuris turi būti nedidesnis nei padvigubintas galimų klaidų kiekviename šifre skaičius. Priešingu atveju net ir žinant privatų kriptosistemos raktą dešifruoti

pranešimo nepavyktų. Priklausomai nuo apskaičiuoto vektorių Hamingo svorio yra generuojamos galimos iškraipytų klaidos vektoriumi šifrų pozicijos. Tarkime, jei vektorių Hamingo svoris yra lygus padvigubintam leistinam klaidų, esančių šifruose skaičiui, tuomet tiek viename šifre, tiek kitame yra parenkamos tarpusavyje nesutampančios pozicijos. Savaime suprantama, jog likusiose pozicijose lieka neiškraipyti duomenys. Pasirinktose pozicijose belieka atsitiktiniu būdu spėti kuriame iš dviejų vektorių yra klaidinga informacija. Tarkime, jog kriptosistema gali ištaisyti tris klaidingas pozicijas, tokiu atveju galimų spėjimų yra 20. Jei sistema sugebėtų taisyti tik dvi klaidingas pozicijas kiekviename šifre, tuomet bandymų skaičius gerokai sumažėja – 6.

Situacija šiek tiek pasikeičia, kuomet dviejų šifrų Hamingo svoris nėra lygus dvigubam galimų klaidų skaičiui. O tai reiškia, jog kažkurioje ar net keliose pozicijoje klaidingi duomenys yra abiejuose šifruose. Tokiu atveju, priklausomai nuo sutampančių pozicijų skaičiaus, surandamos vietos, kur galima tiksliai pasakyti, jog ten yra klaidingi duomenys. Spėjimai daromi tokiu pačiu principu, kuomet sutampančių klaidų pozicijų nėra. Likusioje šifro dalyje daromas atsitiktinis spėjimas, bandant atspėti, kurioje pozicijoje yra klaidingi duomenys abiejuose šifruose. Tokių spėjimų skaičių nėra sunku apskaičiuoti žinant šifro ilgį. Jei šifro ilgis yra *n*, o ištaisomų klaidų skaičių pažymėtume raide *k*, tuomet turint omenyje, jog yra tik viena pozicija, kurioje sutampa klaidinga informacija, spėjimų skaičius yra *n-k+1*.

Antrajame metode buvo realizuojamas šifro dešifravimas. Dešifravimo algoritmas nėra sudėtingas, tereikia išspręsti *n* lygčių su *k* nežinomųjų. Ši lygčių sistema yra gaunama transponavus *G** matricą bei pridedant šifrą, tariamai esantį be klaidų, kaip atsakymų stulpelį. Įdomu tai, jog skaičiavimas vyksta dvejetaine skaičiavimo sistema, todėl sprendiniai yra tikslūs ir diskretiški.

Lygčių sistemos sprendimui buvo pasitelktas Gauso tiesinių lygčių sprendimo metodas, kuomet matrica apdorojama taip, jog jos apatinis kairys kampas lieka sudarytas iš nulių, o įstrižainėje nėra koeficientų lygių nuliui. Tam, kad matrica įgautų tokį pavidalą buvo atliekami eilučių sukeitimo vietomis veiksmai, dauginimo iš reikiamų koeficientų ir sumavimas su atitinkamomis eilutėmis. Kaip žinia, išspręsti *n* lygčių su *k* nežinomųjų sistemą tereikia *k* lygčių, tačiau pradžioje nėra žinoma, kurios iš jų yra tiesiškai nepriklausomos, o kurios yra priklausomos. Todėl atlikus eliminacijos žingsnius su *k* lygčių, darbas yra nutraukiamas ir likusios *n-k* lygtys nėra naudojamos.

Paskutiniame žingsnyje belieka išsireikšti sprendinius. Matyt, patogiausiai bei suprantamiausiai būtų buvę nežinomųjų išsireiškimą realizuoti rekursiniu metodu, tačiau turint omenyje, jog programa turi veikti su ypatingai didelės apimties duomenimis, šios idėjos buvo

atsisakyta. Kaip žinia, rekursinių algoritmų realizacija reikalauja papildomos atminties išskyrimo su kiekviena iteracija, o iteracijų skaičiui gerokai išaugus gali iškilti problemų su kompiuterių atminties skirstymu bei našumu.

Gautasis tiesinių lygčių sprendinys yra bandymas atspėti tikrąjį užšifruotą pranešimą. Gautasis sprendinys nėra saugomas, tad nėra apsaugota nuo galimo tokio pačio spėjimo. Spėjimų saugojimo realizacija nėra sudėtinga, tačiau nėra prasminga, nes duomenų saugojimas bei lyginimas užtruktų daugiau laiko nei naujo bandymo sugeneravimas. Konkrečiu atveju generuojanti matrica buvo aprašyta programos išeities tekste, siekiant patogesnio bei platesnio panaudojamumo, tokio tipo pradinius duomenis reikės importuoti iš tekstinės bylos. Tokios bylos sukūrimui reiks realizuoti dar vieną programą, kuri generuotų tokią matricą, be to, ji tuo pačiu galėtų generuoti atsitiktinius pranešimus, kurie būtų užšifruojami ir kartu su sudarytais šifrais pateikiami, kaip pradiniai duomenys šį semestrą realizuotai bei aprašytai programai. Tokio tipo programa bus reikalinga ne tik tokio pobūdžio atakai pademonstruoti. Kito semestro darbe realizuojamoms atakoms reikės tokių pačių pradinių duomenų.

Šiuo metu realizuotos atakos pradiniais duomenimis yra laikomi abu skirtingų klaidos vektorių iškraipyti šifrai bei originalusis pranešimas. Atrodytų keistoka atakos realizacija, kuri reikalauja dešifruoto pranešimo, kaip įvesties duomenų, tačiau norint statistiškai apskaičiuoti iš kelių bandymų pavyksta teisingai dešifruoti, be originalaus pranešimo nepavyktų. Norint pateikti bandymų rezultatus, t.y. spėjimų sąrašą, tereikėtų nežymiai modifikuoti duomenų išvedimo logiką.

Šiuo metu atakos realizacijoje naudojami sveiko skaičiaus (*int*) duomenų tipai atliekant aritmetines operacijas, siekiant didesnio programos našumo, galima būtų nesunkiai modifikuoti programą taip, jog visi veiksmai būtų atliekami su loginio (*boolean*) tipo kintamaisiais. Kaip žinia, loginio tipo kintamiesiems pakanka vieno bito atmintyje, o sveiko skaičiaus reikia net trisdešimt dviejų.

Atakos tobulinimas įmanomas ją pritaikant nefiksuotam šifrų skaičiui. Šiuo metu yra tariama, jog yra žinomi du šifrai šifruojantys tą patį pranešimą, tačiau visada yra galimybė kriptoanalitikui gauti daugiau nei du šifrus. Tokiu atveju programos veikimas gerokai sutrumpėja, mat galimų bandymų skaičius atitinkamai sumažėja. Tačiau tokio tipo atakos realizacija tampa labiau komplikuota. Iš kitos pusės, jei nebūtų fiksuojamas žinomų šifrų skaičius, galima būtų bandyti atakuoti tiriamąją kriptosistemą turint tik vieną šifrą. Tokio tipo ataką galima būtų pervadinti į visų galimų variantų, kuriuose nėra klaidų, perrinkimo ataką. Visada yra įdomu žinoti, kiek trunka pilno perrinkimo ataka, todėl kito semestro darbe bus patobulinta susijusių pranešimų ataka, tam, kad pasiekti šį tikslą.

## 3.5. Susijusių pranešimų atakos rezultatai

Realizavus ataką buvo atlikta keletą kartų bandymai su tais pačiais duomenimis. Kiekvieno eksperimento metu buvo atliekama po penkiolika identiškų spėjimų tam, kad galima būtų tiksliau įvertinti vidutinį bandymų skaičių bei jų vykdymo trukmę:

| Eksperimento nr. | Vidutinis bandymų skaičius | Vykdymo trukmė, (ms) |
|---|---|---|
| 1 | 4,73 | 31 |
| 2 | 5,46 | 31 |
| 3 | 3,06 | 31 |
| 4 | 4,00 | 32 |
| 5 | 3,80 | 15 |
| 6 | 3,80 | 31 |
| 7 | 4,13 | 16 |
| 8 | 3,73 | 31 |
| 9 | 5,20 | 34 |
| 10 | 4,93 | 16 |
| Vidurkis: | 4,28 | 26,8 |

Gautasis vidurkis (4,28) labai nedaug skiriasi nuo teoriškai suskaičiuoto reikalingų bandymų skaičiaus – 4,29. Tai parodo, jog atakos realizacijos algoritme klaidų nėra. Dešifravimas vidutiniškai truko 26,8 milisekundes, o vieno bandymo trukmė siekė 6,26 milisekundes. Kiekvieno eksperimento trukmė svyravo tarp 15 ir 34 milisekundžių, tokį didelį kraštutinių reikšmių skirtumą galėjo nulemti netolygus kompiuterio resursų paskirstymas, bei lygiagrečiai vykdytų programų kintantis resursų poreikis. Bandymo trukmė turėtų šiek tiek sumažėti jei tik pirmo bandymo metu būtų transponuojama generuojanti matrica $G^*$ bei atliekami Gauso eliminacijos žingsniai. Tačiau reiktų prisiminti, kurios matricos eilutės buvo keičiamos, tam, kad analogiškus sukeitimus atlikti su šifro matrica. Tokiu būdu turėtų žymiai pagreitėti atakos veikimas, ypač su didesniais parametrais, kuomet vidutinis bandymų skaičius yra didesnis.

Ankstesniame skyrelyje buvo apskaičiuotas vidutinis bandymų skaičius reikalingas dešifruoti kriptosistemą, kurios parametrai yra laikomi pakankamai saugiais. Bandymų skaičius siekė $1{,}37 * 10^{16}$, todėl tokios atakos įvykdymas, laikant, kad bandymo trukmė yra lygi

eksperimento metu nustatytai bandymo trukmei, truktų apie 2,6 milijonų metų. Tačiau atliekant veiksmus su didesniais parametrais šis laiko tarpas turėtų dar padidėti. Tačiau minėtas laiko tarpas turėtų gerokai sumažėti ataką vykdant pasitelkus superkompiuterius bei optimizavus skaičiavimus, tačiau vis tiek nepavyktų vykdymo laiko sumažinti iki kelių dienų ar savaičių.

Nereiktų pamiršti, jog susijusių pranešimų atakos rezultatas yra dešifruotas pranešimas, o ne atskleistas privatus raktas. Tad įvykdžius ataką yra sužinoma konfidenciali informacija, tačiau kito pranešimo siuntimo metu reiktų vykdyti lygiai tokią pačią ataką. Anksčiau surasta informacija yra visiškai nenaudinga dešifruojant naują pranešimą.

Geriausia apsauga nuo tokios atakos yra vengimas siųsti susijusių pranešimų. Rekomenduojama prieš užšifruojant pranešimą papildyti jį atsitiktiniais duomenimis, tokiu atveju, siunčiant, kad ir tą patį pranešimą, dešifruoti jo nepavyktų.

Paplitę parametrai ($n \geq 1024, k \geq 524, t \geq 50$) tokio tipo atakai, kaip parodė eksperimentas, nėra įveikiami.

## 3.6. Gavėjo atsako atakos algoritmas

Sėkmingos gavėjo atsako (ang. *reaction*) [HGS97] [HGS98] atakos metu yra dešifruojamas originalusis pranešimas, bet nenustatomas privatusis raktas. Iš to seka, jog kiekvieną siunčiamą pranešimą reikia dešifruoti nepriklausomai, jei tam yra poreikis. Šios atakos sėkmė slypi užšifruoto pranešimo gavėjo reakcijoje į gaunamus pranešimus. Paprastai gavėjas atsakęs apie sėkmingai ar nesėkmingai dešifruotą pranešimą net neįtaria, jog pateikė nemenką užuominą apie originalų pranešimą. Norint nustatyti gavėjo atsako formatą apie prašymą pakartotinai persiųsti užšifruotą pranešimą, reikia atsitiktinai sugeneruotą pranešimą pasiųsti gavėjui ir tikėtis, jog gavėjas nesugebės atšifruoti pranešimo į reikšmingus duomenis. Tiesa, yra tokių sistemų, jog dauguma atsitiktinai sugeneruotų šifrų, dešifravus įgauną reikšminę prasmę. Šio tipo ataka prieš tokias sistemas nėra galima.

Nagrinėjama ataka yra galima tik tuo atveju, kuomet užšifruotame pranešime yra *t* arba mažiau klaidų. Tokiu atveju, pranešimo gavėjas gavęs užšifruotą pranešimą su *t+1* arba daugiau klaidų nebandys pranešimo dešifruoti, o tuo pačiu ir praneš, jog šifras yra nekorektiškas.

Ataka susideda iš dviejų dalių: klaidos vektoriaus pašalinimo iš šifro bei originalaus pranešimo dešifravimo. Pirmoje dalyje nustatomas klaidos vektorius *e*, kuris buvo naudojamas užšifruojant pranešimą *m*. Vėliau apskaičiavus šifrą be klaidingų duomenų deširuojamas pranešimas.

Tarkime, jog turime šifrą *C*, kurį dešifravus gautume ieškomą tekstą – *m*. Norint atskirti klaidos vektorių, mums reikia jį nustatyti, o tai galima realizuoti vykdant vieno bito modifikacijas *C* šifre tol, kol gausime *C'*, kuris nebus deširuojamas išviso (turės daugiau nei *t* klaidų) arba bus deširuojamas į ne *m* tekstą (turės *t* arba mažiau klaidų, bet bus dešifruojamas į kitą pranešimą).

Turėdami *C* šifrą mes galime keisti paeiliui *n* bitų jame į priešingą (0 -> 1 ir atvirkščiai 1 -> 0), t. y. pridedame klaidų. Pirmu žingsiu keičiame tik pirmą bitą, antru – pirmus du bitus, trečiu – pirmus tris bitus ir t.t. Jei atlikus bitų modifikacijas *C'* yra nebeatšifruojamas – reikia nutraukti šio algoritmo darbą bei pereiti prie tolesnių darbų. Paprastai, šio algoritmo veikimas turi nutrūkti vėliausiai per *2t+1* žingsnį, nes blogiausiu atveju mums tektų ištaisyti *t* klaidų bei privelti naujų – *t+1*.

Baigę darbą mes žinome, jog *C´* šifras turi *t+1* klaidų, priešingu atveju, žinutės gavėjas būtų dešifravęs sėkimingai jam siųstą šifrą į žinutę *m*. O klaidų daugiau būti negalėjo, nes to nebūtų leidęs algoritmas, kuris sako, jog pasiekus *t+1* klaidų, darbą reikia baigti. Be to, galioja tokia nelygybė – $t<n/2$.

Turėdami *C'* šifrą, bandykime rasti kurie bitai šiame šifre yra klaidingi. Blogiausiu atveju teks atlikti *n-1* bandymą. Pirmuoju žingsniu keičiame *C'* pirmąjį bitą priešingu bei nukopijavę visus likusius suformuojame *C''*. Jei gavėjas atsako, jog jam pavyko dešifruoti *C''*, tuomet pirmasis bitas yra klaidingas, jį pasižymine. Tuos pačius veiksmus atliekame ir su likusiais *2 .. n-1* bitais, tol kol gavėjas praneša apie nesugebėjimą dešifruoti šifro. Kuomet tai įvyksta (nebūtinai reikia perrinkti visus *n-1* bitus), pasižymėtose pozicijose *C'* sukeičiame bitus priešingais. Tokiu būtų gauname šifrą be klaidos vektoriaus, nes bet kuris *C''* kuris dešifruojamas sėkmingai turi turėti daugiausiai *t* klaidų. Mat Hammingo atstumas tarp *C''* ir *C'* yra 1, o *C''* negali turėti *t+2* klaidų, nes jis nebūtų dešifruotas gavėjo.

Pirmasis žingsnis reikalauja *2t+1* iteracijų, o antrasis mažiau arba lygiai *n-1*. Tai gi, norint atsikratyti klaidos vektoriaus pakanka *n+2t* operacijų, turint omenyje standartinius parametrų dydžius pakaktų 1124 operacijų, o tai laiko prasme užimtų labai nedaug.

Turėdami šifrą be klaidos vektoriaus, galime apskaičiuoti originalųjį pranešimą *m*. Pasižymėkime turimą šifrą raide $C_c$. Pasirenkame *k* bitų iš $C_c$ ir suformuojame $C'_c$. Tokiu pačiu principu pasirenkame *k* stulpelių iš generuojančios matricos *G* ir suformuojame *G'*, kurios dimensijos bus $k \times k$. Tuomet galioja tokia lygybė:

$$C_c^{'} = m * G^{'}$$

iš čia galime išsireikšti *m*, jei egzistuoja $G'$ atvirkštinė. Priešingu atveju pranešimo dešifruoti nepavyks.

$$m = C_c^{'} * G^{'-1}$$

Veiksmų skaičius reikalingas apskaičiuoti *m* pagal aukščiau aprašytą formulę galima sutapatinti su matricos $G'$ inversijai atlikti reikalingų veiksmų skaičiumi.

Taigi, iš teorinių skaičiavimų bei prielaidų galima teigti, jog ataka tam tikromis salygomis gali būti pavojinga. Pagrindinis pavojus – siunčiamos žinutės dešifravimas, tuo tarpu privatus raktas lieka nežinomas. Kitame skyriuje aprašysime atakos realizaciją Java programavimo kalba bei pateiksime atakos vykdymo rezultatus bei išvadas. Patikrinsime, ar teoriniai skaičiavimai pagrindžiami praktine veikla.

## 3.7. Gavėjo atsako atakos realizacija

Realizuojant ataką teko supaprastinti atakos veikimo schemą. Gavėjo atsakas buvo imituojamas vienu metodu. Metodui pateikus kodą, jis atsakydavo ar gavėjas gali dešifruoti pranešimą ar ne. Realizacija nebuvo sudėtinga – tereikėjo patikrinti pateikto šifro bei šifro be klaidos vektoriaus Hammingo atstumą. Jei pastarasis atstumas buvo lygus arba mažesnis leistinam klaidų šifre skaičiui, metodas grąžindavo teigiamą asakymą, priešingu atveju – neigiamą. Siekiant didesnio atitikimo realaus pasaulio uždaviniui, dėl duomenų transportavimo ir pan., metodo realizacijoje buvo įgyvendinta pusės sekundės delsa.

Remiantis ankstesniame skyriuje aprašytu algoritmu buvo realizuotas klaidos vektoriaus eliminavimas iš šifro metodas. Visų pirma buvo keičiami šifro pirmieji bitai, norint suformuoti naują šifrą su *t+1* klaidų. Po kiekvieno bito pakeitimo buvo tikrinama, ar šifras yra dar de6ifruojamas gavėjo, todėl tik gavus teigiamą atsaką buvo atliekamos tolesnės bitų modifikacijos. Baigus darbą šiai metodo sričiai buvo paruoštas šifras su *t+1* klaidų tolimesniam darbui. Turint omenyje, jog pakeitus bitą priešingu naujai sugeneruotame šifre, galima daryti prielaidas apie toje pozicijoje esančio bito korektiškumą, buvo formuojamas masyvas, kuriame buvo saugomos pozicijos, kuriose yra klaidos vektoriaus iškraipyti bitai. Galiausiai, turint klaidos vektoriaus pozicijas, beliko išvalyti nuo jo šifrą. Tokiu būdu, gautą šifrą metodas grąžiną jį kvietusiajam.

Turint šifrą be klaidų, galima, pasitelkus anksčiau realizuotą metodą, gauti gavėjui siųstą originalųjį pranešimą. Tai įvykdžius galima teigti, jog ataka įvykdyta sėkmingai. Žemiau pateiktoje lentelėje galima pamatyti atliktų testų duomenis:

| Eil. nr. | Parametrai (*t; k; n*) | Iteracijų sk. | Vykdymo laikas (s) |
|---|---|---|---|

| | | | |
|---|---|---|---|
| 1 | 50; 515; 1024 | 2048 | 597 |
| 2 | 20; 260; 512 | 1024 | 270 |

Kaip matyti iš gautų rezultatų ši ataka yra labiau nei pavojinga. Pranešimo dešifravimas, kuris trunka mažiau nei dešimt minučių su paprastu asmeniniu kompiuteriu, yra labai greitas. Apsisaugojimas nuo tokios atakos atrodytų paprastas – nereaguoti į prašymus dešifruoti panašius šifrus, tačiau iškyla duomenų saugojimo bei analizavimo problema. Toks apsisaugojimas iš dalies galimas, kuomet užklausų siuntėjas tai daro per Internetą ar vietinį tinklą bei turi pakankamai resursų duomenų kaupimui. Tokiu atveju yra įmanomas užklausų siuntėjo identifikavimas pagal IP adresą, tačiau kitais atvejais apsisaugoti yra gana kėblu. Tarkime, jog USB rakte yra užšifruota informacija ir yra prietaisas galintis tą informaciją dešifravus nuskaityti. Kriptoanalitikas gali imituoti tokio prietaiso darbą bei dinamiškai renkant informaciją apie klaidos vektorių minėtu būdu dešifruoti duomenis.

Šio tipo atakos sėkmingos baigties tikimybės sumažinimui nėra prasmės didinti parametrus, mat jų didinimas atakos laiką tik labai nežymiai didina – galioja tiesinis priklausomumas. Nesant galimybei gavėjui rinkti siunčiamų užklausų, rekomenduojame šios kriptosistemos tokiose programų sistemose nenaudoti.

## 3.8. Visų galimų variantų perrinkimo atakos algoritmai

Ši ataka yra pati trivialiausia ir visiems geriausiai žinoma. Visų galimų variantų perrinkimo (ang. *brute force*) ataka gali būti kelių rušių. Pati paprasčiausia yra mėginimas atspėti pranešimą *m* ir jį užšifravus palyginti su šifru *c*. Jei užšifruoto pranešimo *mG* ir šifro *c* Hammingo atstumas yra mažesnis arba lygus *t*, galime laikyti, jog pranešimą *m* atspėjome. Tačiau yra ir kitas būdas, kuris atsižvelgia į kriptografinės sistemos sandarą. Tokios atakos metu yra bandoma atspėti klaidos vektorių *e* ir atėmus jį iš šifro *c* gauti *n* lygčių su *k* nežinomųjų uždavinį, kurį išsprendus galima dešifruoti pranešimą *m*.

Galime apskaičiuoti, kurios atakos galimų variantų yra daugiau. Taigi, pirmuoju atveju pranešimo *m* ilgis yra *k*, tad galimų variantų yra $2^k$, nes kiekvienoje pozicijoje gali būti arba *0* arba *1* nepriklausomai nuo kitų pozicijų. Tuo tarptu galimų klaidos vektorių *e* yra $\frac{n!}{t!(n-t)!}$. Norint palyginti šias išraiškas galime apskaičiuoti apytiksles jų vertes su rekomenduojamais McEliece kriptosistemos parametrais ($n \geq 1024, k \geq 524, t \geq 50$).

$$2^k = 2^{514} = 5{,}49 \cdot 10^{157}$$

$$\frac{n!}{t!(n-t)!} = \frac{1024!}{50!(1024-50)!} \approx 3{,}19 \cdot 1085$$

Kaip matyti iš apskaičiuotų reikšmių, pirmosios rūšies ataka turi gerokai didesnį variantų spektrą, tačiau atsitiktinio varianto patikrinimas yra žymiai paprastesnis. Tad norint įvertinti kuri ataka našesnė, nepakanka anksčiau atliktų skaičiavimų. Kitame skyrelyje apžvelgsime šį aspketą atlikdami praktinius skaičiavimus, be to galėsime įvertinti tokios atakos efektyvumą bei realų pavojingumą.

## 3.9. Visų galimų variantų perrinkimo atakos realizacija

Realizuojant klaidos vektoriaus atspėjimo būdu paremtą ataką buvo naudojamas *n* lygčių su *k* nežinomųjų sprendimo metodas iš susijusių pranešimų atakos realizacijos. Kadangi spėjimo variantai nėra saugomi ir kiekvienos iteracijos metu generuojamas atsitiktinis spėjimas, teko panaudoti mechanizmą, kurio pagalba buvo geriau parenkamas galimas klaidų skaičius klaidos vektoriuje. Jei būtų atsitiktinai parenkamas galimas klaidų skaičius kiekvienos iteracijos metu, tai ataka būtų ne tokia efektyvi, nes šifrų su didesniu klaidų skaičiumi yra daugiau, o tuo pačiu ir yra didesnė tikimybė, jog klaidos vektorius ir bus su didesniu klaidų skaičiumi.

Sugeneravus klaidos vektorių *e* bei jį atėmus iš šifro *c* gauname šifrą $c_1$. Iš lygybės $c_1 = m' * G$ gauname pranešimą *m'* bei tikriname ar *m = m'*. Atakos realizacijoje yra naudojamas pranešimas *m*, nes kito būdo kaip patikrinti ar šifras $c_1$ yra tas, kurio ieškome nėra. Kitaip tariant, pranešimo *m* žinojimas realizacijoje imituoja šifro gavėjo atsaką siuntėjui. Realaus pasaulio uždaviniuose toks tikrinimas atliekamas taip pat automatizuotai, tačiau yra remiamasi kitomis savybėmis, kaip bandomos įveikti sistemos atsaku ar pan.

Atakos realizacija, kuomet bandoma atspėti pranešimą *m* niekuo nesiremiant, yra gana paprasta. Atsitiktinai sugeneruojamas pranešimas *m*, padauginimas iš generuojančios matricos *G'* bei apskaičiavus Hammingo atstumą tarp gauto rezultato ir šifro *c* nustatoma ar spėjimas yra teisingas.

Žemiau pateikiame lentelę, kurioje galima pamatyti abiejų rūšių visų variantų perrinkimo atakų rezultatus:

| Eil. nr. | Parametrai (*t; k; n*) | *m* spėjimas | | *e* spėjimas | |
|---|---|---|---|---|---|
| | | Iteracijų sk. | Vykdymo laikas (s) | Iteracijų sk. | Vykdymo laikas (s) |
| 1 | 2; 128; 256 | 10 000 | 144 | 10 000 | 11 400 |
| 2 | 2; 34; 64 | 100 000 | 12,5 | 4165 | 1,375 |

Turėdami šiuos duomenis galime apskaičiuoti, kiek truktų atakos vykdymas, jei tartume, jog iteracijų reiks tiek, kiek yra galimų perrinkimo variantų.

$$\frac{2^{128}}{10000} * 144 \approx 4{,}9 \cdot 10^{36} \text{ ir } \frac{256!}{2!(256-2)! * 10000} * 11400 = 37209$$

$$\frac{2^{34}}{100000} * 12{,}5 \approx 2{,}1 \cdot 10^{6} \text{ ir } \frac{64!}{2!(64-2)! * 4165} * 1{,}375 \approx 0{,}666$$

Tiek vienu atveju, tiek ir kitu *e* spėjimas yra laiko atžvilgiu naudingesnis. Todėl pabandysime atlikti šios rūšies ataką prieš kriptosistemą su rekomenduojamais parametrais, su kuriais ji yra laikoma pakankamai saugi. Taigi, atliktų bandymų rezultatai surašyti lentelėje:

| Eil. nr. | Parametrai (*t; k; n*) | *e* spėjimas | |
|---|---|---|---|
| | | Iteracijų sk. | Vykdymo laikas (s) |
| 1 | 50; 512; 1024 | 10 | 10,9 |
| 2 | 50; 512; 1024 | 100 | 112,7 |

Taigi, vienos iteracijos apdorojimas trunka apie vieną sekundę (1,09 – 1,12s) atliekant ataką su 1,6 GHz procesoriumi, todėl galime apskaičiuoti, kiek truktų visų variantų perrinkimas:

$$\frac{1024!}{50!(1024-50)!} \approx 3{,}19 \cdot 10^{85}\, s$$

Kaip matome, ši ataka galėtų trukti $1{,}01 \cdot 10^{78}$ metų. Tai yra daugiau, nei pakankama.

## 3.10. Žinomo dalinio teksto atakos algoritmas

Šio tipo atakos pavojingumas tiesiogiai priklauso nuo žinomo teksto dydžio. Kuo didesnis žinomo teksto santykis su bendru teksto dydžiu tuo trumpesnis yra atakos vykdymo laikas. Minėtu santykiu yra sumažinamas McEliece kriptografinės sistemos vienas iš parametrų *k*.

Tarkime, jog $m_r$ yra žinoma užšifruoto pranešimo dalis, o $m_l$ yra likusioji – nežinoma dalis. Iš to seka, jog:

$$m = m_l + m_r .$$

Žinodami, jog generuojanti matrica *G´* yra sudaryta iš *k* eilučių bei *n* stulpelių, galime pasižymėti tariamai žinomų eilučių skaičių $k_r$, o likusių eilučių skaičių – $k_l$. Iš to seka, jog:

$$k = k_l + k_r$$

Taigi, galime išvesti formulę, kurios pagrindu yra vykdoma ataka:

$$c = mG^{'} + e$$

$$c = m_l G_l^{'} + m_r G_r^{'} + e$$

$$c^{'} = m_l G_l^{'} + e\,.$$

Čia $c^{´}$ yra apskaičiuojamas remiantis turimomis $c$ bei $m_r G^{´}_r$ reikšmėmis. Formulėje $G^{´}_r$ pažymėta generuojančios matricos $k_r$ eilutės, o $G^{´}_l$ yra likusios, nežinomos generuojančios matricos eilutės $k_r$. Tokiu atveju belieka spręsti McEliece kriptografinę sistemą, kurios parametrai yra $n$ bei $k_l$. Jei pastarasis parametras gerokai sumažėjęs, galima naudoti visų variantų perrinkimo algoritmą bei tokiu būdu dešifruoti siųstą pranešimą.

Atakos realizacijai bus naudojama jau realizuota visų galimų perrinkimų atakos realizacija pridedant generuojančios matricos sumažinimo bei ieškomo teksto sutrumpinimo funkcionalumą.

## 3.11. Žinomo dalinio teksto atakos realizacija

Šio tipo atakos realizacija praktiškai nesiskyrė nuo visų variantų perrinkimo atakos realizacijos. Tad verta pažymėti, jog atakos sėkmės tikimybė auga turint didesnį žinomą dalinį tekstą. Žemiau pateiktoje lentelėje galima pamatyti kaip priklauso atakos našumas nuo žinomo dalinio teksto dydžio santykio su viso teksto dydžiu.

| Eil. nr. | Parametrai ($t$; $k$; $n$) | Žinoma dalis (%) | Iteracijų skaičius | Vykdymo laikas ($s$) |
|---|---|---|---|---|
| 1 | 50; 512; 1024 | 10 | 100 | 69 |
| 2 | 50; 512; 1024 | 50 | 100 | 34 |
| 3 | 50; 512; 1024 | 98 | 100 | 3 |

Galime apskaičiuoti kiek vidutiniškai truktų tokia ataka, tardami, kad ataka baigs darbą sėkmingai po visų galimų variantų perrinkimo.

$$\frac{2^{512*90\%}}{100} * 69 \approx 3{,}57 \cdot 10^{138}$$

$$\frac{2^{512*50\%}}{100} * 34 \approx 3{,}93 \cdot 10^{76}$$

$$\frac{2^{512*2\%}}{100} * 3 \approx 36$$

Kaip matome iš apskaičiuotų reikšmių, net žinojimas pusės teksto neleidžia efektyviai dešifruoti likusios dalies. Jei yra reali grėsmė, jog kriptoanalitikas gali turėti dalį siunčiamo pranešimo, verta parametrus (*t, k, n*) padidinti, tuomet dar didesnio informacijos kiekio žinojimas neįgalina dešifruoti likusios dalies.

Laikydami, jog sistema yra pakankamai saugi, kai per metus (~$3{,}15 \cdot 10^7$ s) su vidutinių techninių savybių kompiuterių yra įmanoma perrinkti visus galimus variantus, pasinaudojus bandymo metu gautais duomenimis pateikiame rekomenduojamo dydžio parametrus esant tikimybei būti žinomo teksto daliai:

| Eil. nr. | Žinoma dalis (%) | Parametrai (*t; k; n*) | Vykdymo laikas (*s*) |
|---|---|---|---|
| 1 | 10 | 50; 512; 1024 | $3{,}57 \cdot 10^{138}$ |
| 2 | 50 | 50; 512; 1024 | $3{,}93 \cdot 10^{76}$ |
| 3 | 80 | 50; 512; 1024 | $10^{29}$ |
| 4 | 98 | 150; 1536; 3072 | $2{,}5 \cdot 10^{9}$ |

Kaip matyti anksčiau pateiktoje lentelėje, žinant net 80% originalaus pranešimo pakanka saugumui užtikrinti standartinių parametrų dydžių ($n \geq 1024, k \geq 524, t \geq 50$). Tačiau žinant 98% pradinio pranešimo standartinių parametrų nepakanka saugumui užtikrinti. Teko padidinti parametrus 300%, kad pasiekti pakankamai saugios sistemos atžymą. Pastaruoju atveju, nežinoma 2% teksto dalis būtų dešifruota per daugiau nei 79 metus.

# 4. Išvados

Darbo metu buvo nagrinėjama McEliece kriptografinė sistema. Išskirtos visos sistemos sudedamosios dalys: Goppa kodai, S ir P matricos bei klaidos vektoriai. Aptarta kiekvienos dalies įtaka bendram sistemos saugumui. Kriptosistemos saugumui tirti buvo užšifruotas pavyzdinis pranešimas bei detaliai aprašyta susijusių pranešimų ataka, kuri tą pranešimą dešifravo.

Didelis dėmesys buvo skirtas kito tipo atakoms: gavėjo atsako, visų variantų perrinkimo, žinonomo dalinio teksto. Kiekviena ataka buvo realizuota Java programavimo kalba bei atlikti bandymai su įvairaus dydžio parametrais (*t, k, n*). Kiekvienoje atakoje buvo realizuotas *n* lygčių su *k* nežinomųjų sprendimo algoritmas, kuris padėjo efektyviai spręsti sudarytas lygčių sistemas. Tyrimo metu paaiškėjo, jog gavėjo atsako ataka yra labai pavojinga net ir su rekomenduojamo dydžio parametrais, todėl rekomenduojama realizuojant programų sistemas su tiriama kriptosistema didelį dėmesį atkreipti į šią problemą. Jei yra galimybė, nuo tokios atakos apsiginti lengviausia nepateikiant atsakymo siuntėjui apie pranešimo dešifravimo sėkmę ar nesėkmę, jei to padaryti negalima, belieka ignoruoti prašymus dešifruoti pranešimus, kurių tarpusavio Hammingo atstumas yra lygus arba mažesnis nei parametras *t*. Tačiau tokia rekomendacija yra prasminga, kuomet gavėjas turi pakankamai resursų kaupti bei analizuoti jam siunčiamus duomenis, priešingu atveju šios kriptografinės sistemos nederėtų naudoti.

Visų variantų perrinkimo ataka ištirta spėjant pranešimą *m* bei klaidos vektorių *e*. Tiek vienu atveju, tiek ir kitu buvo gautas tas pats rezultatas – dešifruoti tokių būdu pranešimo neįmanoma per priimtiną laiko tarpą. Tiriant žinomo dalinio teksto ataką pasitvirtino spėjimas, jog kuo didesnė pranešimo *m* dalis yra žinoma, tuo lengviau dešifruoti likusią dalį. Jei yra tikimybė, jog dalis pranešimo *m* gali būti žinoma kriptoanalitikams, tuomet vertėtų padidinti kriptosistemos parametrus.

## 5. Šaltinių sąrašas


[AES01] Specification for the Advanced Encryption Standard (AES). 2001, pp 3-24.

[BDL98] Dan Boneh, Richard A. DeMillo, Richard J. Lipton. On the Importance of Eliminating Error in Cryptographic Computations. [žiūrėta 2008-05-08]. Prieiga per Internetą: <http://crypto.stanford.edu/~dabo/papers/faults.ps.gz>

[Ber97] Thomas A. Berson. Failure of the McEliece Public-Key Cryptosystem under Message-Resend and Related-Message Attack. Springer – Verlag, Berlin,1997, pp. 213-220.

[Bla99] Simon Blake-Wilson. Elliptic Curve Cryptography. Standarts for Efficient Cryptography. Certicom Research, September 1999, pp 1 – 88.

[Can98] Anne Canteaut. Public-key cryptosystems based on error-correcting codes. [žiūrėta 2008-05-02]. Prieiga per Internetą: <http://www-rocq.inria.fr/codes/Anne.Canteaut/English/mceliece.html>

[Con07] Eric Conrad. Types of Cryptography Attacks. GIAC Research in the Common Body of Knowledge. 2007, 1 – 7 pp.

[Cop94] Don Coppersmith. The Data Encryption Standard (DES) and its strength against attacks. [žiūrėta 2008-05-02]. Prieiga per Internetą: <http://www.research.ibm.com/journal/rd/383/coppersmith.pdf>

[CS98] A. Canteaut and N. Sendrier. Cryptanalysis of the original McEliece cryptosystem. Springer – Verlag, Berlin, 2000, pp 187-199.

[DS03] Valdas Dičiūnas, Gintaras Skersys. Mokymo priemonė „Diskrečioji matematika“. Vilnius, 2003.

[Dun06] Orr Dunkelman. Techniques for Cryptanalysis of Block Ciphers. Research Thesis. [žiūrėta 2008-05-08]. Prieiga per Internetą: <http://lj.streamclub.ru/crypto/theses/dunkelman.pdf>

[EOS06] Daniela Engelbert, Raphael Overbeck, Axel Schmidt. A Summary of McEliece-Type Cryptosystems and their Security. [žiūrėta 2008-05-08]. Prieiga per Internetą: <http://lj.streamclub.ru/crypto/reports/mceliece.ps>

[HGS97] Chris Hall, Ian Goldberg, Bruce Schneier. Reaction Attacks Against Several Public-Key Cryptosystems. [žiūrėta 2008-05-16] Prieiga per Internetą: <http://www.cypherpunks.ca/~iang/pubs/paper-reaction-attacks.pdf>

[HGS98] Chris Hall, Ian Goldberg, Bruce Schneier. Security-Related Comments Regarding McEliece's Public-Key Cryptosystem. Prieiga per Internetą: <http://dsns.csie.nctu.edu.tw/research/crypto/HTML/PDF/C87/224.PDF>

[Joc02] Ellen Jochemsz. Goppa Codes & the McEliece Cryptosystem. Universiteit Amsterdam, 2002.

[KI00] Kazukuni Kobara, Hideki Imai. Semantically Secure McEliece Public-Key Cryptosystems -Conversions for McEliece PKC-. [žiūrėta 2008-05-08]. Prieiga per Internetą: <http://www.mathmagic.cn/Crypt1998-2003/papers/1992/19920019.pdf>

[KT91] Valery I. Korzhik and Andrey I. Turkin. Cryptanalysis of McEliece's Public-Key Cryptosystem.

[LM99] Sergio Loureiro, Refik Molva. Function Hiding Based on Error Correcting Codes. Institut Eurecom, Sophia Antipolis, France.

[LS00] Pierre Loidreau, Nicolas Sendrier. Weak keys in the McEliece public-key cryptosystem. Project CODES, INRIA Rocquencourt Domaine de Voluceau, B. P. 105 78 153 Le Chesnay CEDEX, France, 2002, pp. 1 – 13.

[McE78] Robert J. McEliece. A Public-Key Cryptosystem Based On Algebraic Coding Theory. DNS Progress Report 42-44, pp. 114 – 116.

[MOV96] Alfred J. Menezes, Paul C. van Oorschot ir Scott A. Vanstone. Handbook of applied cryptography. Press, October 1996, pp. 1 – 816, Chapter 8.

[Pin97] Richard Pinch. Coding theory: the first 50 years. [žiūrėta 2008-05-02]. Prieiga per Internetą: <http://plus.maths.org/issue3/codes/index.html>

[Pom94] Carl Pomerance. Smooth numbers and the quadratic sieve. [žiūrėta 2008-05-02]. Prieiga per Internetą: <http://www.math.leidenuniv.nl/~reinier/ant/sieving.pdf>

[Ske05] Gintaras Skersys. Klaidas taisančių kodų teorija. Paskaitų konspektai. [žiūrėta 2008-05-02]. Prieiga per Internetą: <http://www.mif.vu.lt/~skersys/ktkt/KTKT1-1.pdf>

[Sta05] Vilius Stakėnas. Šifrų istorijos. TEV, 2005.

[Sta07] Vilius Stakėnas. Kriptologijos 2007 metų kursas. [žiūrėta 2008-05-02]. Prieiga per Internetą: <http://www.mif.vu.lt/katedros/matinf/asm/vs/pask/crypto/cr_07.html>

[Stu02] Chris Studholme. The Discrete Log Problem. 2002, pp. 1 – 57.

[Sun00] Hung-Min Sun. Enhancing the Security of the McEliece Public-Key Cryptosystem. [žiūrėta 2008-05-02]. Prieiga per Internetą: <http://www.iis.sinica.edu.tw/JISE/2000/200011_01.pdf>