Kauno technologijos universitetas

Matematikos ir gamtos mokslų fakultetas

Dviejų blokinių šifrų, paremtų matricinio laipsnio funkcija, skaičiuoklės režimų taikymas trumpiems ir ilgiems pranešimams šifruoti

Baigiamasis magistro studijų projektas

Matas Levinskas

Projekto autorius

Doc. dr. Aleksėjus Michalkovič

Vadovas

Prof. dr. Eligijus Sakalauskas

Konsultantas

Kaunas, 2023

Kauno technologijos universitetas

Matematikos ir gamtos mokslų fakultetas

Dviejų blokinių šifrų, paremtų matricinio laipsnio funkcija, skaičiuoklės režimų taikymas trumpiems ir ilgiems pranešimams šifruoti.

Baigiamasis magistro studijų projektas

Taikomoji matematika (6211AX006)

Matas Levinskas

Projekto autorius

Doc. Aleksėjus Michalkovič

Vadovas

Prof. dr. Eligijus Sakalauskas

Konsultantas

Lekt. dr. Tadas Telksnys

Recenzentas

Kaunas, 2023

Kauno technologijos universitetas

Matematikos ir gamtos mokslų fakultetas

Matas Levinskas

Dviejų blokinių šifrų, paremtų matricinio laipsnio funkcija, skaičiuoklės režimų taikymas trumpiems ir ilgiems pranešimams šifruoti

Akademinio sąžiningumo deklaracija

Patvirtinu, kad:

1. baigiamąjį projektą parengiau savarankiškai ir sąžiningai, nepažeisdama(s) kitų asmenų autoriaus ar kitų teisių, laikydamasi(s) Lietuvos Respublikos autorių teisių ir gretutinių teisių įstatymo nuostatų, Kauno technologijos universiteto (toliau – Universitetas) intelektinės nuosavybės valdymo ir perdavimo nuostatų bei Universiteto akademinės etikos kodekse nustatytų etikos reikalavimų;

2. baigiamajame projekte visi pateikti duomenys ir tyrimų rezultatai yra teisingi ir gauti teisėtai, nei viena šio projekto dalis nėra plagijuota nuo jokių spausdintinių ar elektroninių šaltinių, visos baigiamojo projekto tekste pateiktos citatos ir nuorodos yra nurodytos literatūros sąraše;

3. įstatymų nenumatytų piniginių sumų už baigiamąjį projektą ar jo dalis niekam nesu mokėjęs (-usi);

4. suprantu, kad išaiškėjus nesąžiningumo ar kitų asmenų teisių pažeidimo faktui, man bus taikomos akademinės nuobaudos pagal Universitete galiojančią tvarką ir būsiu pašalinta(s) iš Universiteto, o baigiamasis projektas gali būti pateiktas Akademinės etikos ir procedūrų kontrolieriaus tarnybai nagrinėjant galimą akademinės etikos pažeidimą.

Matas Levinskas

Patvirtinta elektroniniu būdu

Levinskas, Matas. Dviejų blokinių šifrų, paremtų matricinio laipsnio funkcija, skaičiuoklės režimų taikymas trumpiems ir ilgiems pranešimams šifruoti. Magistro studijų baigiamasis projektas / vadovas doc. dr. Aleksėjus Michalkovič / konsultantas prof. Eligijus Sakalauskas; Kauno technologijos universitetas, Matematikos ir gamtos mokslų fakultetas.

Studijų kryptis ir sritis (studijų krypčių grupė): Taikomoji matematika (Matematikos mokslai).

Reikšminiai žodžiai: Blokinis šifras, simetrinis šifras, šifravimo režimai, matricinio laipsnio funkcija, puikus saugumas, CBC, CTR.

Kaunas, 2023. 50 p.

Santrauka

Tobulėjantys kvantinių kompiuterių pajėgumai artina prie dienos, kuomet šiuo metu taikomi kriptografiniai algoritmai duomenų šifravimui neužtikrins pakankamos saugos. Šiame darbe lyginami simetrinio šifravimo algoritmų režimai, kurie teoriškai turėtų užtikrinti pakankamą saugą prieš kvantinius kompiuterius. Lyginant gautuosius rezultatus, pastebėta, kad taikant CBC šifravimo režimą, gaunamų šifrogramų elementų skirstiniai yra artimi tolygiajam, o taikant CTR režimą, reikia įvesti papildomų modifikacijų siekiant užtikrinti pakankamą elementų sumaišymą, leidžianti paslėpti pradinį pranešimą, tiek šifrui paremtu komutatyvia grupe, tiek nekomutatyvia grupe. Didinant matricos eilę didėja ir šifravimo laikas, todėl rekomenduojama rinktis mažesnės eilės blokus, o šifruojamas didesnis bitų skaičius naudojant didesnę grupės eilę užtikrina greitesnį tekstogramos užšifravimą, todėl jeigu įmanoma rekomenduojama rinktis didesnę grupės eilę.

Levinskas, Matas. Application of the CTR modes of two matrix power function-based block ciphers to encrypt short and long messages / supervisor doc. dr. Aleksėjus Michalkovič / consultant prof. Eligijus Sakalauskas; Faculty of Mathematics and Natural Sciences, Kaunas University of Technology.

Study field and area (study field group): Applied Mathematics (Mathematical Sciences).

Keywords: Block cipher, symmetric cipher, modes of operations, matrix power function, perfect secrecy, CBC, CTR.

Kaunas, 2023.50 p.

Summary

Advancements in quantum computer computational capabilities brings us closer to the day when currently used cryptographic algorithms will not be sufficient to ensure the safety of encrypted data. In this paper we are focusing on two perfectly secure block cyphers which should provide enough complexity to ensure security against quantum computers. After comparing obtained results, it was noticed that by applying CBC mode of operations it ensures that no information about the plaintext is visible by having only ciphertext. However, it was spotted that there is a need for some additional modifications for CTR mode to ensure that the ciphertext does not contain any information about the plaintext. Furthermore, it was noticed that the larger is the order of the block, the more time it takes to encrypt the plaintext. Also, if we would take 16-bit elements and increase order of the group it will ensure faster encryption time, reducing number of blocks needed to encrypt the plaintext.

Turinys

Paveikslų sąrašas

Lentelių sąrašas

Santrumpų ir terminų sąrašas

Santrumpos:

MLF – Matricinio laipsnio funkcija;

CBC – Blokų grandinės režimas (angl. Cipher block chaining mode);

CTR – Skaičiuoklės režimas (angl. Counter mode);

ECB - Elektroninės knygos režimas (angl. Electronic code book);

NIST – nacionalinis standartų ir technologijos institutas (angl. National Institute of Standards and Technology);

AES – simetrinis šifravimo algoritmas vadinamas pažengusiu duomenų šifravimo standartu (angl. advanced encryption standard)

DES – simetrinis šifravimo algoritmas vadinamas duomenų šifravimo standartu (angl. data encryption standard)

TDES – trigubas DES algoritmas (angl. triple encryption standard)

IV – inicijavimo vektorius.

Terminai:

Tekstograma – pranešimas skirtas šifravimui. Tai gali būti tekstinis, skaitinis pranešimas, dažniausiai pateikiamas bitų eilute.

Šifrograma – rezultatas gautas, užšifravus tekstogramą.

Įvadas

Šiame darbe nagrinėjami algoritmai remiasi matricinio laipsnio funkcija. Jos tobulinimas ir pritaikymas kriptografiniams algoritmas yra viena iš pagrindinių Kauno technologijos universiteto Kriptografijos ir blokų grandinių mokslinės grupės krypčių, kurios vadovas yra profesorius Eligijus Sakalauskas. Šiame darbe pateikti algoritmai publikuoti ir tobulinti 2020-2022 laikotarpiu ir šiems yra pateikti vieno bloko, bei CBC šifravimo režimo puikaus saugumo įrodymai, įvestos papildomos modifikacijos saugumo, sumaišymo savybėms gerinti, pateikiamos papildomos rekomendacijos ir įrodymai [1-7]. Visų šių algoritmų tobulinimu siekiama apibrėžti ir užtikrinti pakankamą algoritmų saugą prieš kvantinę kriptoanalizę. Čia pateikiama naujausios algoritmų versijos atsižvelgiant į papildomus pakeitimus taikant skirtingus šifravimo režimus.

Darbo tikslas - ištirti skaičiuoklės režimo taikymą dviem pateiktiems šifravimo algoritmams apibrėžtais virš matricinio laipsnio funkcijos šifruojant ilgus ir trumpus pranešimus.

Darbo uždaviniai:

- Ištirti CTR šifravimo režimą, nustatyti ar šis užtikriną pakankamą elementų sumaišymą;
- Palyginti gautuosius rezultatus taikant CBC šifravimo režimą, bei su AES šifravimo algoritmu taikant CBC ir CTR režimus;
- Palyginti gaunamus šifravimo laikus;
- Pateikti rekomendacijas.

Rezultatų aprobavimas

Šis baigiamasis darbas yra tąsa pristatytų ar ruošiamasis pristatyti tyrimų rezultatų, bei išleistų publikacijų:

- 2022 Lietuvos mokslo tarybos vasaros praktikos tyrimas „Blokinio šifro, veikiančio CTR režimu saugumo analizė“. Gautuosius rezultatai pateikti ICIST konferencijai ir šiuo metu laukiama atsakymo.
- Išleisto straipsnio bendraautorius: MIHALKOVICH, Aleksejus, **Matas LEVINSKAS**, Lina DINDIENĖ ir Eligijus SAKALAUSKAS. CBC Mode of MPF Based Shannon Cipher Defined Over a Non-Commuting Platform Group. *Informatica*, 2022, 33.4: 833-856. DOI: 10.15388/22-INFOR499
- Išleisto straipsnio bendraautorius: MIHALKOVICH, Aleksejus; **Matas LEVINSKAS** ir Eligijus SAKALAUSKAS. Counter Mode of the Shannon Block Cipher Based on MPF Defined over a Non-Commuting Group // Mathematics. Basel : MDPI. ISSN 2227-7390. 2022, vol. 10, iss. 18, art. no. 3363, p. 1-17. DOI: 10.3390/math10183363.
- Pristatytas pranešimas „Blokinis Shannon’o šifras paremtas nekomutyviaja kriptografija“ 2022 metų Lietuvos matematikos draugijos organizuojamoje konferencijoje.
- Ruošiamasi pristatyti pranešimą „Simetrinio blokinio šifro taikymas blokų grandinės ir skaičiuoklės režimais ilgiems ir trumpiems pranešimams užšifruoti“ birželio 21-22 d Lietuvos matematikos draugijos organizuojamoje konferencijoje.

1. Literatūros apžvalga

1.1. Duomenų šifravimas

Kriptografiniai algoritmai skirstomi į asimetrinius ir simetrinius. Asimetriniai algoritmai taikomi bendraujant dviem šalim, šios turi susijusių privačiojo ir viešojo raktų poras, kuriomis pasinaudojant yra užšifruojami ir atstatomi pradiniai pranešimai. Šio tipo šifravimo algoritmai naudojami elektroniniams parašams, duomenų užšifravimui arba bendro rakto sudarymui papildomai taikant simetrinius šifravimo algoritmus. Norint užšifruoti pranešimą taikant asimetrinį šifravimo algoritmą, naudojamas šalies, kuriai yra siunčiamas pranešimas, viešasis raktas, o iššifravimo procesas atliekamas pasinaudojant savo privačiuoju raktu gavus šifrogramą. Asimetriniai šifravimo algoritmai gali būti pritaikyti dokumentų pasirašymui, čia veiksmai atliekami atvirkštine tvarka, t. y. privatusis raktas naudojamas parašui padėti, o patikrinimui naudojamas viešasis šalies, kuri pateikė parašą, raktas. Kai dvi bendraujančios šalys nori apsikeisti slapta užšifruota informacija, šios turi turėti bendrą slaptą raktą, kurio pagalba galėtų užšifruoti ir iššifruoti norimą pranešimą. Šio rakto sudarymui naudojami asimetriniai algoritmai, kurie turi užtikrinti, kad bendrasis raktas gautas naudojant šalies kuriai siunčiamas pranešimas viešojo ir mūsų privačiojo rakto rezultatas būtų lygus mūsų privačiojo ir šalies, kuri gaus pranešimą, privačiojo rakto kombinacijai. Vienas tokių pavyzdžių būtų 1976 metais publikuotas Diffio ir Helmano raktų apsikeitimo protokolas [8], kurio pagalbą dvi komunikuojančios šalys gali sudaryti bendrą raktą naudojamą tolesniuose duomenų šifravimo žingsniuose.

Skirtingai nei asimetriniuose šifruose, simetriniai šifrai turi tik vieną bendrą rakta, čia remiamasi prielaida, kad bendrasis raktas jau yra sutartas, tai gali būti užtikrinta fiziškai apsikeitus šia informacija ar pasinaudojant raktų apsikeitimo algoritmais, sudarant bendrą raktų porą.

Simetrinio šifravimo schema arba kitaip šifras, yra sudaromas iš trijų algoritmų:

- Raktų generavimo algoritmo, žymimo $Gen()$. Tai probabilistinis algoritmas, kuris sugeneruoja raktą k naudojant apibrėžtą skirstinį.
- Šifravimo algoritmo, žymimo $Enc()$. Šiai funkcijai pateikiamas sugeneruotas raktas k ir tekstograma η, o gražinama šifrograma c:

$$c = Enc(\eta, k) \tag{1.1}$$

- Iššifravimo algoritmo, žymimo $Dec()$. Šiai funkcijai pateikus šifrogramą ir bendrą raktą k, išgaunama pradinė tekstograma η:

$$\eta = Dec(c, k) \tag{1.2}$$

Turint raktų generavimo procedūrą galima apibrėžti visų įmanomų raktų aibę $\mathcal{K}$. Analogiškai visų tekstogramų aibę galima pažymėti $\mathcal{M}$. Kadangi šifrograma yra gaunama šifruojant pranešimą $\eta \in \mathcal{M}$ su pasirinktu raktu $k \in \mathcal{K}$, tai visų įmanomų šifrogramų aibė C sugeneruojama naudojant aibių $\mathcal{M}$ ir $\mathcal{K}$ kombinacijas.

Norint užtikrinti sėkmingą duomenų šifravimą ir iššifravimą bet kokiai šifravimo schemai, ši turinti sugeneruotą raktą, pasinaudojant generavimo funkcija *Gen()* su visomis tekstogramos $\eta \in \mathcal{M}$ reikšmėmis, turi užtikrinti lygybę:

$$Dec(Enc(\eta, k), k) = \eta \quad (1.3)$$

Kitaip sakant, šifravimo schema turi užtikrinti, kad visada būtų galima atstatyti pranešimą turint šifrogramą su atitinkamu bendru raktu.

Norint užšifruoti pranešimą šis yra apdorojamas suskaidant jį į daug dalių, šios dalys gali būti atskiri baitai arba baitų blokai. Todėl skiriamos dvi skirtingos kategorijas, blokiniai ir srautiniai šifrai. Srautiniai šifravimo algoritmai, skirtingai nei blokiniai šifruoja kiekvieną baitą atskirai, šio tipo algoritmuose i-tasis šifrogramos baitas priklauso nuo i-tojo tekstogramos baito, naudojant slaptąjį raktą ir jo būseną. Pagrindinis šių šifrų uždavinys užtikrinti, kad naudojamas raktą būtų galima išplėsti iki begalybės, užtikrinant, kad visi elementai būtų užšifruoti, t. y. užtikrinti gerą pseudo atsitiktinių skaičių generatorių rakto generavimui. Galima išskirti kelis srautinių šifrų pavyzdžius tokius kaip Salsa20, ChaCha20, Vernamo šifras [9-11].

Blokinis šifras vaizduoja tekstogramą į dažniausiai tokio paties ilgio šifrogramą. Šio šifravimo tipo sistemos dažniausiai sujungia perkėlimų ir maišos idėjas siekiant sugeneruoti geresnį elementų sumaišymą ir kiek įmanoma panaikinti ryšius tarp naudojamo rakto ir gaunamos šifrogramos. Jeigu naudojamas šifravimo algoritmas yra saugus, tuomet blokiniai šifrai gali būti naudojami srautinių šifrų sudarymui, t. y. naudojami šifravimo režimai, kurių pagalba įmanoma tai realizuoti. Tuo siekiama užtikrinti saugumo sąlygas prieš pasirenkamų tekstogramų atakas [12]. Keli blokinių šifrų pavyzdžių būtų DES, TDES, AES, Feistelio šifravimo algoritmai [13-16].

1.2. AES šifravimo algoritmas

Advanced encryption standard (AES) yra simetrinis šifravimo algoritmas (dar vadinamas *Rijindael* algoritmu), tai nacionalinio standartų ir technologijos instituto (NIST) patvirtintas ir laikomas saugiu naudoti algoritmas. Detalus šio šifro aprašas ir rekomendacijos pateikiamos NIST išleistoje publikacijoje [15]. Tai blokinis šifras, kuris remiasi perstatymo ir sukeitimo operacijomis, leidžiančiomis globaliai ir lokaliai pakeisti elementus. Norint pasiekti geresnį rezultatą naudojami raundai, tai algoritmo kartojimas šifruojant pranešimą daugiau negu vieną kartą. Raundų tikslas - užtikrinti gautoje šifrogramoje pakankamą gautųjų elementų sumaišymą. Jų skaičius, turimame algoritme, priklauso nuo naudojamo rakto ilgio. Toliau esančioje lentelėje pateikiama rekomenduojamų raundų skaičiaus priklausomybė nuo skirtingų raktų ilgių (1 lentelė).

1 lentelė. Rakto ilgio, blokų dydžio ir raundų skaičiaus priklausomybė nuo naudojamo šifravimo algoritmo

	Rakto ilgis	**Bloko dydis**	**Raundų skaičius**
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

Čia rakto ilgiai 4,6 ir 8 atitinkamai parodo, kiek 32 bitų žodžių yra naudojama raktų sudarymui, atitinkamai šie sudaromi naudojant 128, 192 ir 256 bitus. Nepriklausomai nuo pasirinkto rakto ilgio, vieno šifruojamo bloko dydis visada bus 4 eilės matrica sudaryta iš 8 bitų elementų, kitaip sakant blokas sudarytas iš 128 bitų. Kiekvienas elemento b_i bitas yra interpretuojamas kaip daugianario $p(x)$ koeficientas:

$$p(x) = b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0 = \sum_{i=0}^{7} b_i x^i. \quad (1.4)$$

Pasinaudojant apibrėžtomis sandaugos ir sudėties operacijomis virš Galua lauko $GF(2^8)$ galima suformuoti transformacijas, kurios yra naudojamos AES šifravimo algoritmui atlikti: 1) baitų perstatymo naudojant apibrėžtą lentelę, 2) eilučių skirtingo postūmio, priklausančio nuo pozicijos matricoje, 3) stulpelių sumaišymo, 4) raundo rakto pridėjimo funkcijas. Šių operacijų ir jų atvirkštinių, leidžiančių atstatyti pradinį pranešimą detalus sudarymo aprašymas randamas NIST pateiktoje publikacijoje [15].

Remiantis [17] ir [18, 25psl] šiuo metu tiek AES-128, tiek AES-192 ar AES-256 užtikrina pakankamą saugą. Nors ir pateikiama pasiūlymų AES-192, AES-256 nulaužimui [19], tačiau jie dar nekelia grėsmės taikant rekomenduojamus šifro parametrus. [18] šaltinyje pateikiamas pavyzdys, kuriame taikomas Groverio atakos algoritmas, jame teigiama, kad rasti naudojamą raktą pranešimui užšifruoti taikant pilnojo perrinkimo (angl. Brute force) algoritmą, tai pavyktų padaryt per 30 metų arba per metus, jeigu sujungtume maždaug 900 tokių kvantinių kompiuterių. Reikia atkreipti dėmesį, kad šiame pavyzdyje apibrėžiamas kvantinis kompiuteris, kuris gali vieną AES bloką užšifruoti per vieną nanosekundę, o tai jam reikėtų padaryti išbandyti visus įmanomus scenarijus, t. y. 2^{120} kombinacijų. Nors prireikia pakankamai daug laiko vienai tekstogramai gauti, tačiau šios atakos nereikėtų nuvertinti ir rekomenduojama pereiti prie ilgesnio naudojamo rakto (256 bitų), siekiant užtikrinti geresnį saugumą [18], [20].

1.3. Blokų paruošimas šifravimui

Taikant blokinius šifrus ir šifruojant didelį duomenų kiekį susiduriama su problema, kada naudojamo rakto ilgis yra per trumpas visam pranešimui užšifruoti. Jeigu rakto neįmanoma praplėsti dėl algoritmo, atminties are greičio apribojimų, tuomet pranešimai gali būti dalinami į atskirus blokus, kurie yra šifruojami atskirai naudojant tą patį bendrą raktą. Reikia atkreipti dėmesį, kad padalinus pranešimą į fiksuoto dydžio blokus, ne visi elementai gali lygiai išsidalinti, tokiu atveju atliekama papildoma modifikacija, leidžianti tuščius matricos elementus užpildyti arba atsitiktiniais elementais, arba sutartomis reikšmėmis, kurias galima identifikuoti atstačius tekstogramą. Išskiriami skirtingi būdai kaip tai galima padaryti, vienas iš būdų - likusius elementus užpildyti bitais, apibrėžtais atitinkamomis taisyklėmis arba paprasčiausiai pridėti atitinkamus baitus, taip pat apibrėžtus iš anksto sutartomis taisyklėmis. Tarkime, turime žodį „padding“, kurį norėsime sutalpinti į 3 eilės matricą. Galima pastebėti, kad norint užpildyti matricą, reikia dar dviejų simbolių. Šį žodį išreiškus šešioliktainiu formatu turime: *0x70 0x61 0x64 0x64 0x69 0x6e 0x67*. Keletas skirtingų variantų su duotuoju pavyzdžiu pateikiami toliau [21].

Bitų užpildas:

- 1 bito užpildas (angl. *Bit padding*). Taikant šį užpildą, gale pranešimo įrašomas papildomas bitas lygus 1, o likę bitai priskiriami 0. Pritaikius šį metodą, duotasis pavyzdys šešioliktainiu formatu atrodytų: *0x70 0x61 0x64 0x64 0x69 0x6e 0x67* ***0x80 0x00***.
- Vienodų bitų užpildymas (angl. *Trailing Bit Complement padding*). Priklausomai nuo paskutiniojo bito, jei šis yra 0, likę prirašomi 1 ir atvirkščiai, jeigu paskutinysis bitas yra lygus 1, tuomet likę prirašomi 0. Pateiktame pavyzdyje paskutinysis baitas lygus *0x67* (01100111), kadangi paskutinysis bitas lygus 1, todėl likę 16 bitų priskiriami 0.

Šešioliktainiu formatu pavyzdys atrodytų taip: *0x70 0x61 0x64 0x64 0x69 0x6e 0x67* ***0x00 0x00***.

Baitų užpildas:

- PKCS#5 ir PKCS#7. Šie užpildo variantai remiasi panašiu principu, prideda pasikartojantį baitą priklausomai nuo trūkstamų baitų. Pavyzdžiui, jeigu trūksta 2 baitų, tuomet pridedami du elementai *0x02*. PKCS#5 apibrėžtas 8 baitams, o PKCS#7 galima naudoti nuo 1-255 baitų užpildui [22], [23]. Duotajam pavyzdžiui trūksta 2 baitų, todėl atlikę šį užpildymo būdą turėtume: *0x70 0x61 0x64 0x64 0x69 0x6e 0x67* ***0x02 0x02***.
- ISO 7816-4. Šis užpildo tipas po paskutinio elemento padeda *0x80* baitą ir kiti elementai, priklausomai nuo trūkstamo kiekio, yra lygus *0x00*. Galima pastebėti, kad šis užpildas sutampa su pirmuoju bitų užpildymo metodu, jei elementai sudaryti iš 8 bitų. Jeigu elementai būtų sudaryti iš skirtingo bitų skaičiaus arba pranešimas būtų suformuotas kitaip, tuomet šie užpildymo metodai išsiskirtų.
- ISO 10126-2. Šis užpildo variantas pasirenka atsitiktinius baitus priklausomai nuo trūkstamo kiekio ir paskutinis baitas yra lygus kiekiui kiek papildomų elementų reikėjo įrašyti. Pavyzdžiui jeigu iki pilno bloko prie tekstogramos reikia prirašyti papildomi 4 baitus, tuomet šie gali atrodyti taip: *0x13 0xae 0xff 0x04*. Čia *0x04* parodo, kad 4 paskutinieji baitai yra užpildas. Realizavus šį užpildymą turimas pranešimas atrodytų taip: *0x70 0x61 0x64 0x64 0x69 0x6e 0x67* ***0xff 0x02***.
- Nulinio baito užpildas. Šiame variante papildomi baitai yra lygūs *0x00*. Šio užpildo trūkumas išryškėja, jeigu originalios tekstogramos paskutinis baitas lygus *0x00*. Taikant šį metodą atskirti originalų pranešimą nuo užpildo yra labai sudėtinga. Dėl šios priežasties šis užpildymo variantas nėra rekomenduojamas. Šis metodas būtų tinkamas, jeigu būtų galima garantuoti, kad paskutinysis baitas niekada nebus lygus *0x00* arba būtų žinomi konkrečios tekstogramos dimensijos. Duotasis pavyzdys realizavus šį užpildymą: *0x70 0x61 0x64 0x64 0x69 0x6e 0x67* ***0x00 0x00***.

Priklausomai nuo taikomo algoritmo gali būti taikomas tiek vienas, tiek kitas užpildas, svarbiausia, kad metodą būtų galima identifikuoti komunikuojančioms šalims. Jeigu naudojami elementai nėra lygūs vienam baitui arba negalima jo lygiai išskaidyti, tuomet bitų užpildo metodai gali būti pritaikomi, priešingu atveju galima taikyti baitų užpildo alternatyvas, kurios, kaip galima pastebėti, kai kuriai atvejais duoda identiškus rezultatus.

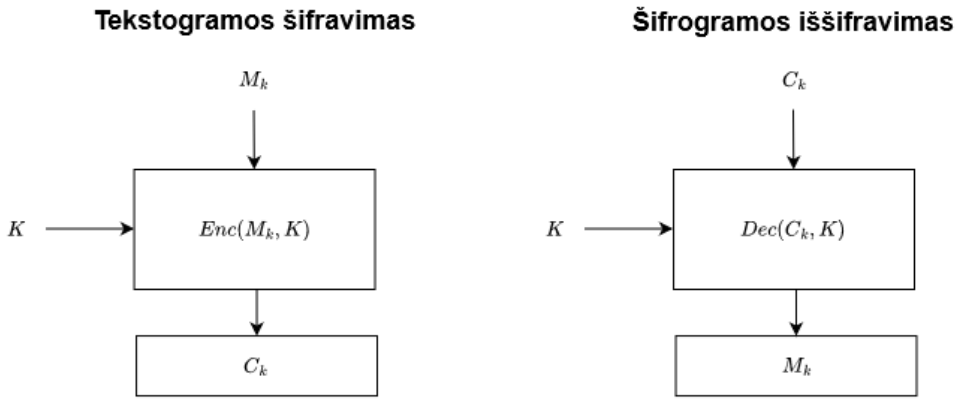
1.4. Šifravimo režimai

Turint pilnai užpildytus blokus galima taikyti norimą šifravimo režimą. Režimai tokie kaip ECB, CBC, OFB, ir CFB buvo apibrėžti 1980-1981 metų NIST publikacijoje [24]. Šiuos režimus buvo siūlyta taikyti DES algoritmui. Tačiau buvo įsitikinta, kad šis algoritmas nėra saugus [25] ir patvirtintas kitas standartas, AES šifravimo algoritmas, tačiau ta pati šifravimo režimo metodika galėjo būti pritaikoma AES ir kitiems blokiniams šifravimo algoritmams. Vėlesnėje 2001 publikacijoje pateikiama šių režimų schemos, taip pat įterptas papildomas CTR režimo apibrėžimas [26]. Vien šiais režimais nėra apsiribojama ir yra kuriamos bei bandomos įvairios modifikacijos, leidžiančios užtikrinti geresnį saugumą bei papildomas savybes. Pavyzdžiui, režimas GCM sujungia tiek CBC, tiek CTR režimų idėjas, kur tekstograma yra ne tik užšifruojama, bet ir sukuriama identifikacijos žyma, kuri leidžia įsitikinti turinio autentiškumu. Šis šifravimo režimas gali būti

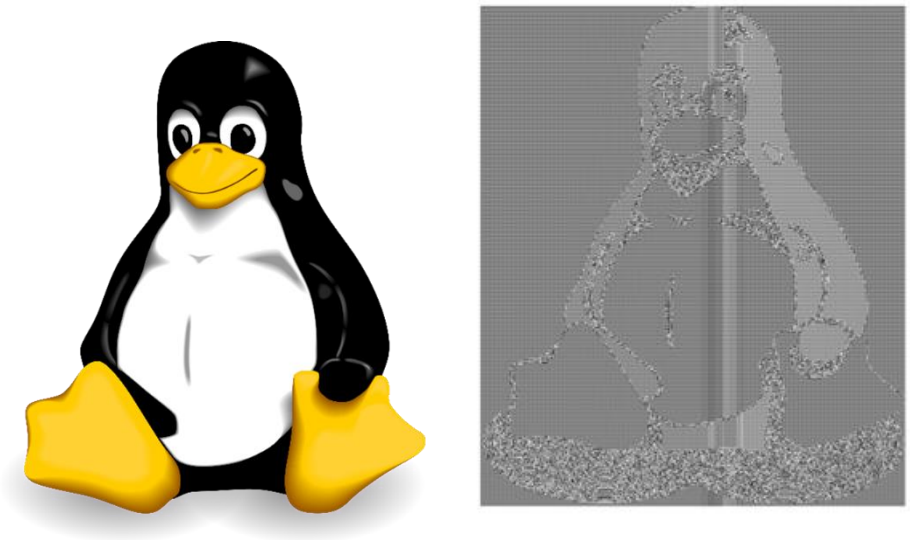
randamas etherneto, belaidžio tinklo saugumui užtikrinti, SSH ir TLS protokoluose. Toliau pateikiamas ECB, CBC ir CTR režimų trumpas aprašymas.

1.4.1. Elektroninės knygos režimas (ECB)

Taikant ECB (ang. *Electronic code book*) šifravimo režimą, kiekvienas blokas gali būti užšifruojamas ir iššifruojamas nepriklausomai nuo kitų blokų. Pagrindinė šio režimo problema yra vienodos išvesties gavimas pateikiant tokią pačią įvestį. Šifruojant didelį duomenų kiekį, sudarytą iš vienodų blokų, galima pastebėti dėsningumus šifrogramose, kurios suteikia informacijos apie pradinę tekstogramą. Šifravimo ir iššifravimo schema pateikta 1 pav.



1 pav. Elektroninės knygos režimo šifravimo ir iššifravimo schemos

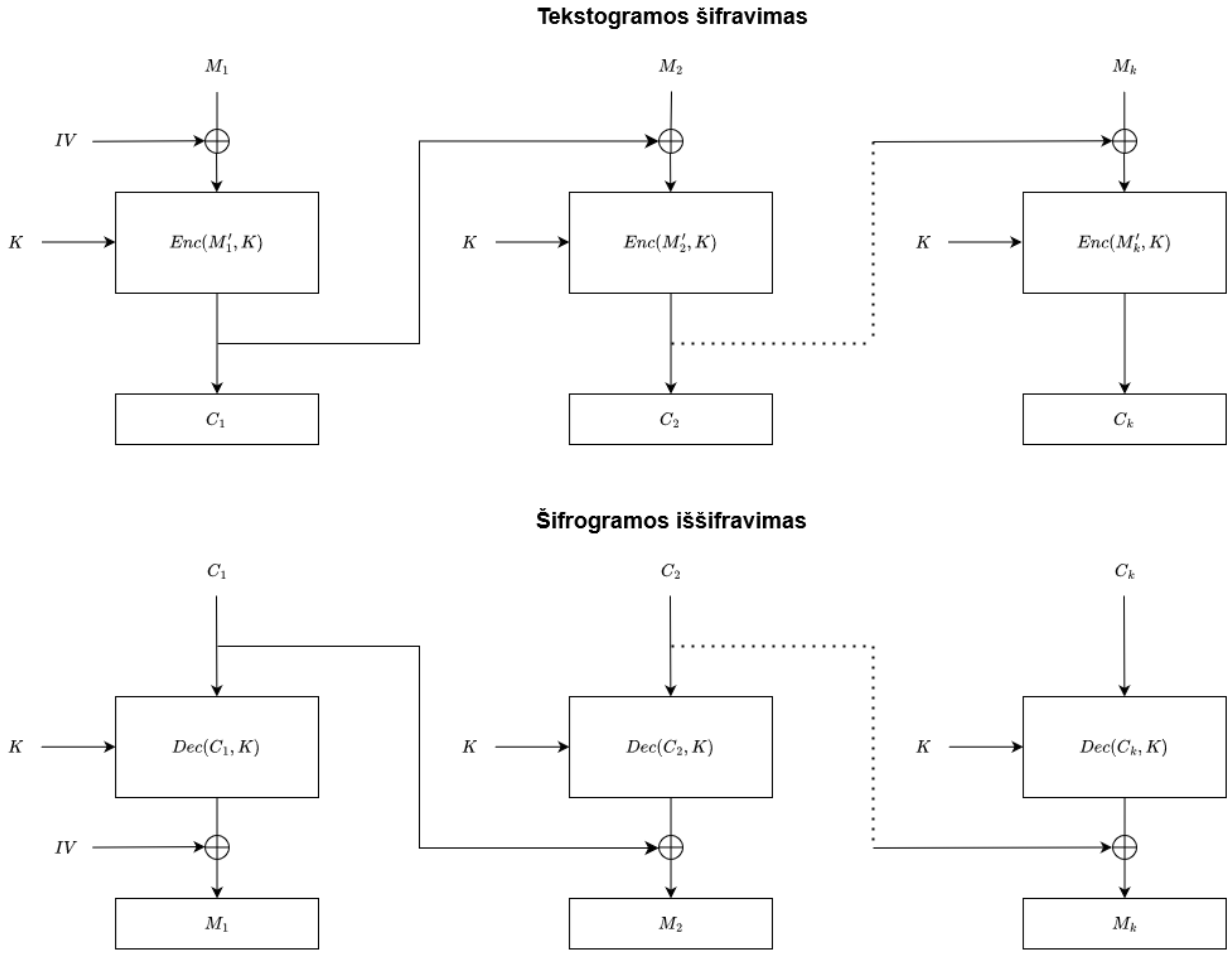


2 pav. Pradinis paveikslas ir gautoji šifrograma ECB šifravimo režimu

2 pav. pateiktas pradinis ir ECB režimu užšifruotas paveikslas, naudojant 2.2. skyrelyje apibrėžtą šifravimo algoritmą. Galima pastebėti, kad gautojoje šifrogramoje yra įžvelgiami kontūrai, kurie leidžia lengvai identifikuoti pradinio pranešimo turinį. Dėl šios problemos nėra rekomenduojama naudoti šio šifravimo režimo.

1.4.2. Blokų grandinės šifravimo režimas (CBC)

Šis šifravimo režimas skirtingai nei ECB režimas, užšifruotus blokus papildomai perduoda sekantiems blokams, kur šie yra sudedami moduliu 2 bitų atžvilgiu, t. y. pritaikoma XOR operacija ($\oplus$). Šis blokinio šifravimo režimas pagerina elementų sumaišymą, išvengiant problemos su kuria susiduria ECB režimas šifruojant vienodus blokus, tačiau praranda galimybę lygiagretinti procesus. 3 pav. schemoje galima pastebėti, kaip vyksta šifravimo ir iššifravimo procesas. Čia reikėtų atkreipti dėmesį, kad pirmajame bloke pradinis tekstogramos blokas kartu su inicijavimo (pradžios) vektoriumi (IV) yra sumuojamas taikant XOR operaciją, skirtingai negu tai yra daroma kituose blokuose. IV yra pasirenkamas atsitiktinai kiekvienos duomenų šifravimo sesijos metu. Schemoje elementai $M_1' = IV \oplus M_1; M_2' = C1 \oplus M_2; \ldots; M_k' = C_{k-1} \oplus M_k$.



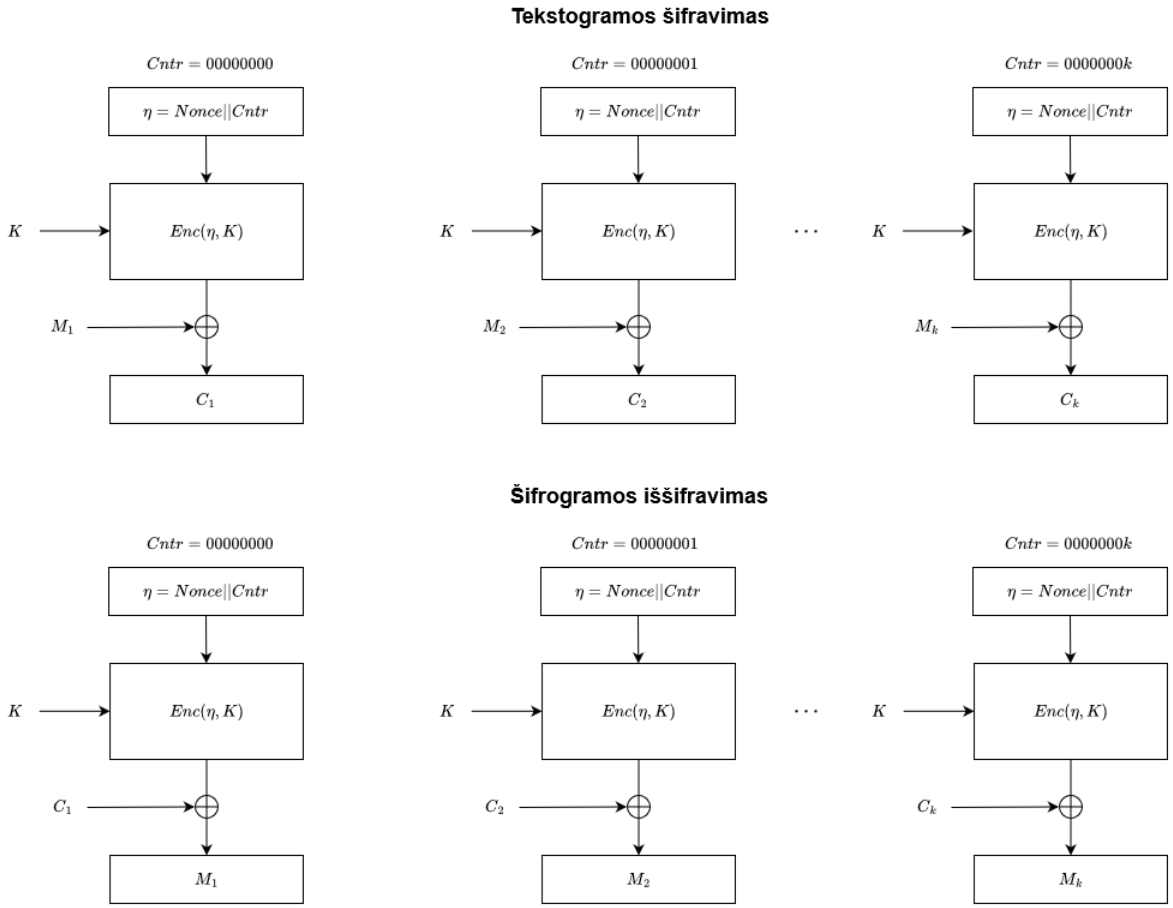
3 pav. Šifrogramos blokų sujungimo režimo šifravimo ir iššifravimo schemos

1.4.3. Skaičiuoklės režimas (CTR)

Šis šifravimo režimas, lyginant su apibrėžtais prieš tai, išsiskiria pora savybių. Pirmiausia šiame režime naudojamas skaitiklis, kuris yra didinamas pereinant prie kito šifruojamo bloko, taip keičiant įvesties bloką, kuris susideda iš kamšalo (angl. *nonce*) ir minėtojo bloko indekso sujungimas. Taip pat taikant šį režimą naudojama tik užšifravimo funkcija. Šis šifravimo režimas savo metodika yra panašus į Vernamo šifrą [11], kuriame pranešimas ir raktas yra sumuojamas taikant XOR operaciją, taip gaunant šifrogramą. Priešingai nei Vernamo šifre, CTR režime raktas, kuris yra naudojamas užšifruoti pranešimą, gaunamas kamšalo ir skaitiklio elementą užšifravus pasirinktu atskiru

algoritmu, taikant privačiuosius raktus. Abiejų šifrų tekstogramos yra atstatomos taikant identiškus žingsnius kaip ir tekstogramos šifravime, t. y. taikant XOR operaciją. Šis režimas taip pat kaip ir ECB režimas yra lygiagretinamas, bei neturi jokios priklausomybės nuo kitų blokų, todėl veiksmai gali būti atliekami nepriklausomai vieni nuo kitų.

Šio režimo schema pateikiama 4 pav., čia skaitiklis yra pažymėtas *Cntr*, kamšalas *Nonce*, o įvestis pateikiama šifravimo funkcijai pažymima η. Reikia atkreipti dėmesį, kad įvesties η ilgis turi atitikti turimo šifravimo algoritmo įvesties parametrus, $Nonce$ parametras turi būti fiksuotas, o parametras $Cntr$ didinamas vienetu kiekviename kitame bloke kaip parodyta schemoje.



4 pav. Skaičiuoklės režimo šifravimo ir iššifravimo schemos

1.5. Matricinio laipsnio funkcija

Pirmą kartą matricinio laipsnio funkcija pristatyta profesoriaus Eligijaus Sakalausko 2007 metais [27] ir 2012 autoriai MLF apibrėžė [28] formaliai kaip vaizdavimą $Mat_m(\mathbb{R}) \times Mat_m(\mathbb{S}) \times Mat_m(\mathbb{R}) \mapsto Mat_m(\mathbb{S})$. Bendru atveju operatorius $Mat_m(\cdot)$ leidžia identifikuoti, turimą $m \times m$ kvadratinę matricą, kurios elementai yra apibrėžiami virš pasirinktos algebrinės struktūros. Šiuo atveju tai būtų elementai apibrėžti virš platforminės pusgrupės $\mathbb{S}$ arba baigtinio sveikųjų skaičių žiedo $\mathbb{R}$. Tarkime turime matricas $X, Y \in Mat_n(\mathbb{R})$ ir $W, E \in Mat_m(\mathbb{S})$, tuomet matricinio laipsnio funkcija apibrėžiama:

$$^{X}W^{Y} = E \tag{1.5}$$

Kiekvienas matricos E elementas gali būti išreikštas formule:

$$e_{ij} = \prod_{k=1}^{n}\prod_{l=1}^{n} w_{kl}^{x_{ik}y_{lj}} \tag{1.6}$$

Jeigu platforminė pusgrupė $\mathbb{S}$ yra komutatyvi, tada šis vaizdavimas yra asociatyvus tiek iš kairės, tiek iš dešinės pusės ir galioja toliau pateikta lygybė:

$${}^{X}({}^{U}W^{V})^{Y} = {}^{XU}W^{VY} \tag{1.7}$$

,čia matricos $U, V \in Mat_n(\mathbb{R})$.

Ši savybė užtikrina, atvirkštinį vaizdavimą leidžiantį atstatyti pradinį pranešimą, jeigu egzistuoja tokios atvirkštinė matricos $X^{-1}, Y^{-1} \in Mat_m(\mathbb{R})$.

$${}^{X^{-1}}({}^{X}W^{Y})^{Y^{-1}} = W. \tag{1.8}$$

1.6. Puikaus saugumo apibrėžimas

Remiantis [12] šifravimo schema (Gen, Enc, Dec) virš pranešimų erdvės $\mathcal{M}$ yra puikaus saugumo jeigu su bet kokiu $\mathcal{M}$ pasiskirstymu, kiekvienas pranešimas $m \in \mathcal{M}$, ir kiekviena šifrograma $c \in C$ kurios $\Pr[C = c] > 0$ galioja lygybė:

$$\Pr[M = m | C = c] = \Pr[M = m] \tag{1.9}$$

Kitaip sakant jeigu yra žinoma šifrograma, tai tikimybė atspėti pradinę tekstogramą žinant šifrogramą yra lygiai tokia pati kaip atspėti tekstogramą neturint šifrogramos.

Vienas tokių šifrų, kuriam galima įrodyti šią savybę yra jau minėtas Vernamo šifras [11]. Tai 1917 metais Gilberto Vernamo užpatentuotas šifravimo algoritmas, kurį formaliai būtų galima apibrėžti taip. Tarkime, kad $a \oplus b$ apibrėžia dviejų binarinių eilučių a ir b XOR operaciją, t. y. kiekvienas bitas sudedamas moduliu 2, jeigu $a = a_1, \dots, a_l$ ir $b = b_1, \dots, b_l$ tai $a \oplus b = a_1 \oplus b_1, \dots, a_l \oplus b_l$. Šis šifravimo algoritmo sudarymas apibrėžiamas taip:

- Fiksuojamas parametras $l > 0$. Tada tiek tekstogramų, tiek raktų ir šifrogramų aibės yra lygios $\{0,1\}^l$, t. y. l bitų ilgio.
- Pasinaudojant raktų generavimo algoritmu *Gen()*, sugeneruojamas raktas $k \in K$ remiantis normaliuoju skirstiniu, t. y. kiekvienas skirtingas raktas iš sugeneruotos aibės turi vienodą tikimybę būti paimtas, šiuo atveju su tikimybe lygia 2^{-l}.
- Pasinaudojant šifravimo funkcija *Enc()*, pasirinktu raktu k ir tekstograma η, suformuojama šifrograma: $c = Enc(\eta, k) = \eta \oplus k$.
- Iššifravimas įvykdomas pasinaudojant *Dec()* funkcija, kuriai pateikiama šifrograma ir bendras raktas naudotas šifrogramai sudaryti: $\eta = Dec(c, k) = c \oplus k$.

Galima įsitikinti, kad lygybė $Dec(Enc(\eta, k), k) = \eta$ teisinga, kadangi $\eta = \eta \oplus k \oplus k$, čia XOR operaciją taikoma tokiam pačiam elementui duodą neutralų elementą XOR operacijos atžvilgiu, t. y. $e \oplus \eta = \eta \oplus e = \eta$.

Šaltinyje [12] galima rasti šio šifro puikaus saugumo savybės įrodymą, tačiau reikia atkreipti dėmesį į naudojamų parametrų ilgius, siekiat užtikrinti, kad ši savybė visada galiotų. Literatūroje teigiama, kad jeigu šifravimo schema (Gen, Enc, Dec) laikoma puikaus saugumo šifru virš pranešimų aibės $\mathcal{M}$ ir raktų aibė $\mathcal{K}$ gaunama pasinaudojant *Gen()* funkcija, tuomet $|\mathcal{K}| \geq |\mathcal{M}|$. Tuo remiantis galima teigti, kad norint užtikrinti puikaus saugumo savybę, turime užtikrinti, kad naudojamas raktas šifruojant duomenis nebūtų trumpesnis negu siunčiamas pranešimas. Galima atkreipti dėmesį, kad tai apriboja siunčiamos informacijos kiekį, priklausomai nuo turimų resursų. Nors šis šifras laikomas saugiu, tačiau niekada negalima naudoti to paties rakto skirtingiems pranešimams šifruoti, jei šis reikalavimas nėra užtikrinamas, tada piktavalis gali labai lengvai išgauti siunčiamą informaciją.

1.7. Temos aktualumas

Remiantis atliktais ir besitęsiančiais tyrimais, šiame darbe palyginti gaunami rezultatai praktiškais šifruojant skirtingo ilgio tekstogramas, taikant pasirinktus šifravimo režimus. Išleistuose darbuose [1-7] pateikiami tyrimų rezultatai ir teoriniai įrodymai, remiantis gautaisiais įverčiais ir papildomomis modifikacijomis siekiama įsitikinti naudojamų šifravimo algoritmų patikimumu taikant skirtingus šifravimo režimus. Gautieji rezultatai, padės geriau suprasti pasiūlytų algoritmų savybes, papildomas modifikacijas, reikalingas pakankamam sumaišymui pasiekti, bei laikų priklausomybę nuo grupės ir matricos eilės.

Teoriniai puikaus saugumo rezultatai sutampa su NP-pilnojo uždavinio apibrėžimu remiantis Yao teorema. Publikacijoje [7] yra teigiama, kad NP-pilnasis uždavinys kvantiniams kompiuteriams nėra išsprendžiamas, todėl pasiūlytieji algoritmai užtikrinantys pakankamą saugą ir efektyvumą būtų puikus pakaitalas dabartiniams algoritmams, kuomet kvantinių kompiuterių pajėgumai kels grėsmę šiuo metu naudojamiems algoritmams.

2. Metodika

2.1. Grupė M_{2^t}

Nekomutatyvi grupė M_{16} pirmą kartą paminėta [29]. Čia apibrėžiama tik grupė iš 16 elementų, tačiau ją galima išplėsti ir apibrėžti bendruoju atveju:

$$M_{2^t} = \langle a, b | a^{2^{t-1}} = e, b^2 = e, bab^{-1} = a^{2^{t-2}+1} \rangle. \tag{2.1}$$

Grupę sudaro du generatoriai a ir b, bei trys apibrėžti sąryšiai, iš viso grupę sudaro 2^t elementų. Ši grupė nėra komutatyvi, atlikę pertvarkius galima įsitikinti, kad $ab \neq ba$.

Remiantis sąryšiais galima pastebėti, kad generatoriai b laipsniai yra redukuojami moduliu 2, o generatoriau a laipsniai redukuojami moduliu 2^{t-1}. Atliekant pertvarkius apibrėžiamos daugybos ir laipsnio kėlimo operacijas, kai $\alpha, \alpha_1, \alpha_2 \in \{0,1\}$ ir $k, k_1, k_2, n \in \{0,1, \dots, 2^{t-1} - 1\}$, tai:

Dviejų elementų $b^{\alpha_1}a^{k_1}$ir $b^{\alpha_2}a^{k_2}$ daugyba:

$$\left(b^{\alpha_1}a^{k_1}\right)\left(b^{\alpha_2}a^{k_2}\right) = \begin{cases} b^{\alpha_1+\alpha_2}a^{k_1+k_2+2^{t-2}}, \text{jeigu } k_1 = 1 \; mod \; 2 \text{ ir } \alpha_2 = 1; \\ b^{\alpha_1+\alpha_2}a^{k_1+k_2} \text{ kitu atveju.} \end{cases} \tag{2.2}$$

Elemento $b^{\alpha}a^{k}$ kėlimas laipsniu:

$$\left(b^{\alpha}a^{k}\right)^n = \begin{cases} b^{\alpha n}a^{nk+2^{t-2}\left\lfloor\frac{n}{2}\right\rfloor}, \text{jeigu } k = 1 \; mod \; 2 \text{ ir } \alpha = 1; \\ b^{\alpha n}a^{nk}, \text{kitu atveju.} \end{cases} \tag{2.3}$$

Grupės M_{2^t} atvirkštinis $b^{\alpha}a^{k}$ elementas:

$$\left(b^{\alpha}a^{k}\right)^{-1} = \begin{cases} b^{\alpha}a^{2^{t-2}-k}, \text{jeigu } k = 1 \; mod \; 2 \text{ ir } \alpha = 1; \\ b^{\alpha}a^{-k}, \text{kitu atveju.} \end{cases} \tag{2.4}$$

Visų šių formulių įrodymus galima rasti [30] publikacijoje.

2.2. Puikaus saugumo šifras paremtas komutatyvia grupe

Pirmą kartą puikaus saugumo šifras paremtas komutatyvia grupe pristatytas profesoriaus Eligijaus Sakalausko su komanda 2020 metais [7]. Atliekant tolimesnius tyrimus ir nagrinėjant skirtingus šifravimo režimus buvo atlikti papildomi algoritmo pakeitimai, leidžiantys praplėsti naudojamą algoritmą, bei užtikrinantys puikaus saugumo sąlygą [5]. Toliau pateikiamas algoritmas jau su atliktais algoritmo pakeitimais.

Tarkime turime pranešimą M ir slaptojo rakto vektorių $\vec{K} = \{X, Y, Z\}$, tenkinančius sąlygą $M, X, Z \in Mat_m(\mathbb{Z}_q)$, $Y \in Mat_m(\mathbb{Z}_q \backslash \{0\})$ tuomet algoritmas apibrėžiamas vadovaujantis žingsniais:

$$\begin{aligned} &C_1 = X + M; \\ &C_2 = F(Z) \odot^Y F(C_1)^Y; \\ &C = C_3 = F^{-1}(C_2) + X; \end{aligned} \tag{2.5}$$

Čia vaizdavimas $F(X): Mat_m(\mathbb{Z}_q) \mapsto Mat_m(\mathbb{G}_q)$ yra viešai žinomas vienas į vieną vaizdavimas, kuris yra ne izomorfizmas, leidžiantis matricos įvesties elementus pakeisti į $\mathbb{G}_q$ - Sylovo $\mathbb{Z}_p$ pogrupį. Jeigu turime sveikųjų skaičių multiplikatyvų žiedą $\mathbb{Z}_p$, kur $p = kq + 1$, p ir q pirminiai skaičiai ir $\mathrm{DBD}(k, q) = 1$. Tuomet grupė $\mathbb{G}_q$ vadinama Sylovo pogrupiu, jeigu gautasis pogrupis yra q eilės, t.y. $g^q \equiv 1 \; mod \; p$. Kadangi q yra pirminis skaičius, todėl remiantis Lagranžo teorema, kiekvienas $\mathbb{G}_q$ elementas sugeneruoja visą grupę išskyrus 1.

Operacija $\odot$ yra dviejų matricų Adamaro sandauga, t. y. matricos yra dauginamos ir redukuojamos moduliu p paelemenčiui. Šios operacijos atžvilgiu apibrėžiama atvirkštinė matrica:

$$T^H \odot T = T \odot T^H = \mathbb{1} \tag{2.6}$$

čia gaunama matrica $\mathbb{1}$, kurios kiekvienas elementas e_{ij} yra neutralusis, t. y. $\{e_{ij}\} = 1$.

Norint atstatyti pradinį pranešimą turi būti užtikrinta, kad matrica Y turėtų atvirkštinę. Jeigu ši sąlyga galioja, tuomet galima apibrėžti iššifravimo žingsnius:

$$\begin{aligned} &D = C - X; \\ &D_2 = {}^{Y^{-1}}\left(F(D_1) \odot F(Z)^H\right)^{Y^{-1}}; \\ &M = D_3 = F^{-1}(D_2) - X; \end{aligned} \tag{2.7}$$

2.3. Puikaus saugumo šifras paremtas nekomutatyvia grupe

Pirmoji simetrinio šifravimo algoritmo paremto matricinio laipsnio funkcija virš nekomutatyvios grupės buvo nagrinėjama bakalauriniame darbe. Atlikę papildomus tyrimus buvo padaryta papildomų korekcijų, kurios suteikia algoritmui geresnį sumaišymą. Naujausia šifravimo algoritmo versija pateikiama [4].

Du papildomi vaizdavimai ϕ ir ψ yra įvedami, kurie padeda lengviau nagrinėti algoritmą paremtą nekomutatyvia struktūra. Šie vaizdavimai padeda M_{2^t} elementus atvaizduoti į atitinkamas $\mathbb{Z}_2$ ir $\mathbb{Z}_{2^{t-1}}$ struktūras, t.y. $\phi: M_{2^t} \mapsto \mathbb{Z}_2$ ir $\psi: M_{2^t} \mapsto \mathbb{Z}_{2^{t-1}}$.

$$\begin{aligned} &\phi\left(b^\beta a^\alpha\right) = \beta; \\ &\psi\left(b^\beta a^\alpha\right) = \alpha. \end{aligned} \tag{2.8}$$

Analogiškai matricoms naudojami vaizdavimai Φ ir Ψ, kurie leidžia atlikti (2.8) pateiktas operacijas kiekvienam duotosios matricos elementui atskirai.

Tarkime turime pranešimą M, kurį norime iššifruoti naudojant bendrojo rakto vektorių $\vec{K} = \{X, Y, \Delta\}$. Pirmiausia pranešimas M padalinamas į dvi atskiras matricas $M_a \in \mathbb{Z}_2$ ir $M_b \in \mathbb{Z}_{2^{t-1}}$, tai padaryti galima paprasčiausiai pirmus elementų matricoje M bitus priskyrus M_a matricai o likusią dalį surašant į matricą M_b. Tuomet galima įvykdyti toliau pateiktus žingsnius:

$$\begin{aligned} &C_1 = b^{M_b + \Delta} \odot a^{M_a + X}; \\ &C_2 = {}^{Y}\, C_1{}^{Y}; \\ &C = C_3 = Shift_\kappa(\Phi(C_2) || \Psi(C_2)) + (\Delta || X); \end{aligned} \tag{2.9}$$

Čia operatorius || apibrėžia dviejų matricų sujungimo operaciją kiekvienam matricos elementui atskirai, $Shift_{\kappa}$ – cikliškas bitų postūmis per κ bitų į dešinę pusę. Galima pastebėti, kad atlikę veiksmus paskutiniame žingsnyje iš elementų apibrėžtų virš M_{2^t} gauname elementus apibrėžtus virš $\mathbb{Z}_{2^t}$. Svarbu paminėti, kad parametras κ yra sutartinis, tačiau pastebėta, kad užtenka pasirinkti jį lygų 2, atliekant lavinos efekto tyrimus pastebėta, kad didelio pokyčio pasirinkus $2 < \kappa < t$ nepastebima. Skirtingai negu prieš tai nagrinėta algoritme, kadangi čia yra naudojamas 2 laipsniai, todėl visi t bitų yra išnaudojami grupės elementų sudarymui. Kai naudojamas pirminis skaičius to pasiekti nėra įmanoma, galima tik stengtis parinkti tokį pirminį skaičių, kuris išnaudotų kiek įmanoma daugiau bitų.

Norint atstatyti pradinį pranešimą reikia užtikrinti, kad C_2 žingsnyje būtų įmanoma atstatyti C_1. Tačiau turint nekomutatyvią grupę tai padaryti nėra taip paprasta, jeigu platforminė grupė apibrėžta virš M_{2^t}, tuomet turime:

$$\begin{aligned} &(W^Y)^{Y^{-1}} \neq W; \\ &{}^{Y^{-1}}\left({}^{Y}W\right) \neq W; \\ &\left({}^{Y}W\right)^{Y} \neq {}^{Y}(W^Y). \end{aligned} \tag{2.10}$$

Tačiau buvo nustatyta, kad pasirinkus atitinkamą matricos Y struktūrą, aukščiau minėtosios lygybės tampa teisingos. 2022 metų straipsnyje [6], pateikti rakto Y generavimo žingsniai padedantys suformuoti matricą, kuri turėtų atvirkštinę ir tenkintų savybes leidžiančias atstatyti pradinį pranešimą:

- Sugeneruojama laikina matrica Y', kurios elementai tolygiai pasiskirstę virš $\mathbb{Z}_{2^{t-2}}$;
- Pasirenkama binarinė kėlinių matrica P iš m! eilės kėlinių aibės $\mathbb{P}_m \subset Mat_m(\mathbb{Z}_2)$;
- Matrica Y gaunama: $Y = 2Y' + P$

Galima pastebėti, kad atlikę šiuos žingsnius turime tik m nelyginių elementų, kurie tolygiai pasiskirstę iš aibės $\{1,3,5,\dots,2^{t-2}-1\}$, o lyginiai elementai tolygiai pasiskirstę iš elementų aibės $\{0,2,4,\dots,2^{t-2}\}$.

Sugeneravus parametrą Y pagal žingsnius, užtikrinus, kad matrica Y turės atvirkštinę, iššifravimo algoritmas gali būti apibrėžtas, leidžiantis išgauti pradinį pranešimą:

$$\begin{aligned} &D_1 = Shift_{t-\kappa}(C - \Delta||X); \\ &D_2 = b^{D_{1b}} a^{D_{1a}}; \\ &D_3 = {}^{Y^{-1}} D_2 {}^{Y^{-1}}; \\ &D_a = \Psi(D_3) - X; \\ &D_b = \Phi(D_3) - \Delta; \end{aligned} \tag{2.11}$$

Čia matricos D_{1b} ir D_{1a} yra suformuojamos tokiu pačiu principu kaip buvo formuojamos matricos M_b ir M_a, kur pirmieji D matricos elementų bitai yra sudedami į matricą D_{1b}, o likę perrašomi į D_{1a}. Norint gauti iššifruotą pranešimą, matricų D_b ir D_a elementai turi būti sujungiami, t. y. tekstograma D gaunama $D = D_b||D_a$.

2.4. Algoritmų modifikacija šifravimo režimų pritaikymui

Skyrelyje 1.4. pateikiami šifravimo režimai bendruoju atveju. ECB režimo realizacija dviem pristatytiems šiframs išlieka nepakitus, tačiau CTR ir CBC šifravimas turi savo subtilybių. Remiantis NIST aprašu [26] tiek CBC, tiek CTR režimuose elementai yra sumuojami taikant XOR operaciją, t. y. abiejų elementų bitai yra sudedami moduliu 2. Aukščiau pateiktiems algoritmams išbandyta sudėtis atitinkamu moduliu priklausomai nuo pasirinktos grupės eilės, kurie užtikrina geresnį elementų sumaišymą. Gauti rezultatai ir rekomendacijos naudojant abi metodikas pateikti 3 skyriuje. Nepaisant šios modifikacijos likęs algoritmas lieka nepakitęs ir šifravimo algoritmai realizuojami taikant skirtingus šifravimo režimus bendruoju atveju apibrėžiami taip:

- CBC režimui:

$$C^{(i)} = Enc\left(M^{(i)} \oplus C^{(i-1)}, \vec{K}\right);$$
$$M^{(i)} = Dec\left(C^{(i)}, \vec{K}\right) - C^{(i-1)} \quad (2.12)$$

Čia $*^{(i)}$ nurodo i-tąjį bloką, operatorius $\oplus$ nurodo XOR operaciją kiekvienam matricos elementui, tačiau kaip ir buvo minėta anksčiau šis operatorius bus pakeičiamas suma grupės eilės moduliu. Taip pat matrica $C^{(0)} \in \mathbb{Z}_{2^t}$ yra lygi IV, t. y. kiekvienos sesijos metu atsitiktinai sugeneruotai matricai. Vektorius $\vec{K}$ – slaptasis raktas.

- CTR režimui:

$$C^{(i)} = Enc\left(\eta^{(i)}, \vec{K}\right) \oplus M^{(i)}$$
$$M^{(i)} = Enc\left(\eta^{(i)}, \vec{K}\right) - C^{(i)} \quad (2.13)$$

Čia i-tojo bloko įvestis su atitinkamu i-tuoju bloko indeksu $\eta^{(i)} = Nonce||Cntr^{(i)}$. Šis parametras šifravimo funkcijai pateikiama kaip m-tos eilės matrica. Čia taip pat operatorius $\oplus$ bus keičiamas suma grupės eilės moduli atskiruose tyrimo pavyzdžiuose.

2.4.1. Tekstogramos elementų modifikacija

Šiame darbe nagrinėjami paveikslai ir tekstiniai pranešimai, šie gali būti koduojami įvairiai. Paveikslai yra sudaryti iš pikselių, kurių reikšmės yra nuo 0 iki 255 kas reiškia, kad kiekvienas pikselis yra koduojamas 8 bitais arba 1 baitu. Tekstas gali būti koduojamas įvairiai, priklausomai nuo tekstogramoje esančių simbolių. Šiame darbe naudojama ASCII koduotė, kuri vieną teksto simbolį paverčia į 1 baitą. Ši koduotė buvo pasirinktina, kadangi šifravimui naudojami tik lotyniški simboliai, norint šifruoti daugiau simbolių galima naudoti ir kitokias šifruotes kaip UTF-8 ar UTF-16, kas leidžia nuskaityti ne tik lotyniškas raides, bet ir kiniškus, japoniškus simbolius ar matematinius operatorius. Galime pastebėti, kad abiejų tyrime naudojamų tekstogramų elementai sudaryti iš 1 baito, kas leidžia naudoti tas pačias grupės eiles tiek vienu tiek kitu atveju. Tačiau, kadangi šiame darbe aprašytų algoritmų parametrai gali būti laisvai keičiami (lyginant su AES, kur bloko dydis fiksuotas, lygus 4), todėl tekstogramos įvestis galima koreguoti, bitus apjungiant arba išskaidant, t. y. 1 baitą išskaidyti į du elementus po 4 bitus arba apjungti 2 baitus ir padaryti vieną elementą iš 16 bitų. Ši modifikacija leistų naudoti skirtingas grupių eiles priklausomai nuo atliktos modifikacijos, taip sumažinant blokų skaičių.

2.4.2. Papildomi vaizdavimai

Atliekant lavinos efekto tyrimus CTR šifravimo režimui pastebėta, kad nors ir vieno bloko lavinos efektas yra pakankamas (artimas 0.5), tačiau didinant matricų blokų indeksą, šiame šifravimo režime susiduriama su problema, kuomet sumaišymas yra ne toks geras (lavinos efektas artimas 0.2). Šiai problemai spręsti vienas iš pasiūlymų buvo įvesti papildomą vaizdavimą, kuris leistų sumaišyti pradinės įvesties elementus. Tai nėra pats geriausias sprendimo būdas, nes atmintyje turi būti saugomos papildomos matricos leidžiančios atitinkama tvarka sumaišyti elementus. Ruošiamas ieškoti geresnių alternatyvų ateities darbuose. Vaizdavimas gali būti apibrėžtas taip:

$$F_\zeta(x) = \vec{v}_\zeta, 0 \leq \zeta < p \tag{2.14}$$

čia $\vec{v}_\zeta$ yra kėlinių vektorius sudarytas iš elementų $\overline{1, m^2}$.

Vaizdavimas pasirenkamas priklausomai nuo įvesties η bitų sumos moduliu p.

$$\zeta = sum(\eta_b)\ mod\ p \tag{2.15}$$

Nauja matricos η įvestis gaunama:

$$\eta' = F_\zeta(\eta) \tag{2.16}$$

Svarbu atkreipti dėmesį, kad parametras p yra pirminis skaičius, kuris gali būti pasirenkamas laisvai, priklausomai nuo vaizdavimų skaičiaus. Trumpiems pranešimams šis dydis nebūtinai turi būti didelis, pavyzdžiui $p = 5$, dažniausiai užtikrins pakankamą šifrogramos elementų sumaišymą. Tačiau pastebėti, kad turint ilgesnį pranešimą, prie skirtingų sugeneruotų naudojamų raktų pastebėti netolygumai, kurie išsprendžiami padidinus vaizdavimų skaičių. Tiksli priklausomybė ir šio vaizdavimo tobulinimas planuojamas tolimesniuose tyrimuose.

2.5. Pradinės įvesties apdorojimas šifravimui

Šiame darbe nagrinėjamos dviejų tipų tekstogramos, paveikslai ir tekstiniai pranešimai. Abiejų realizacijos yra labai panašios, žvelgiant į bitų lygmenį, tai yra tik 1 ir 0 rinkinys, tačiau kuomet norime išskaidyti pirminį pranešimą ir sudaryti atskirus blokus, bei atstatyti jį į originalą, tai padaryti tampa sudėtingiau. Toliau pateikiami pavyzdžiai kaip buvo apdorojami paveikslai ir tekstiniai pranešimai paruošiant šiuos šifravimui, bei kas turėtų būti pritaikoma atstačius tekstogramą siekiant išvengti papildomų elementų, kurie gali būti pridedami siekiant suformuoti pilną bloką.

2.5.1. Paveikslų apdorojimas

Idealiu atveju kai turimą paveikslą norime padalinti į fiksuoto dydžio blokus, pradinio paveikslo matmenys turėtų būti norimo bloko matmenų kartotiniai. Tačiau toks scenarijus sutinkamas retai ir susiduriama su situacija, kai turimi blokai nėra pilnai užpildyti (5 pav. kairėje pateiktas paveikslas, raudonos linijos išskiria paveikslą į keturis blokus šiame pavyzdyje). Šiai problemai išspręsti yra skirtingų priemonių priklausomai nuo norimos realizacijos. Vienas iš metodų, būtų dėti blokus iš eilės ir jeigu atsiranda nepilnas blokas, tuomet sekančius pikselius prijungti prie egzistuojančio bloko. Tačiau šis metodas reikalauja papildomos logikos, kuri leistų paveikslo pikselius kilnoti ir tą patį reikėtų padaryti atstačius pranešimą. Šiuo atveju taip pat reiktų užpildyt trūkstamus pikselius.

Kitas metodas, kuriam nereikia papildomų perstatymų remtųsi idėja, kad pradinio paveikslo pikseliai yra papildomi pasirinktomis reikšmėmis arba pasirinkta logika taip, kad paveikslo išmatavimai būtų turimo bloko dydžio kartotiniai (5 pav. paveikslas dešinėje). Šiame darbe buvo pasirinktas pastarasis metodas, nors taikant šį metodą bus turima daugiau pridedamų papildomų reikšmių, tačiau išvengiama papildomo tikrinimo ir logikos, kuri būtų taikoma apjungiant ir išskaidant blokus. 1.4. skyrelyje pateikiama keletas būdų kaip galima užpildyti tuščius blokus, patogumo dėlei buvo pasirinkta tuščius baitus priskirti 0x00, t. y. padaryti juodą rėmelį.

5 pav. Paveikslo padalinimas į blokus (raudona linija) ir trūkstamų elementų užpildymas

2.5.2. Teksto apdorojimas

Skirtingai negu paveikslai, tekstiniai pranešimai neturi stačiakampės struktūros. Žinoma, ją galima suformuoti jeigu turimas ilgas tekstas ir yra išskiriamas atitinkamas bitų skaičius vienai eilutei. Tačiau tai apsunksta, kuomet turimas trumpas tekstinis pranešimas su išskirta atitinkama vieta tekstui, kitaip sakant dideliame lape gali būti labai daug tuščios vietos, kuri neturi būti šifruojama. Todėl šiems pranešimams buvo pasirinkta kitokia metodika. Teksto paversto į baitus eilutės buvo talpinamos į blokus ir likęs paskutinysis blokas buvo užpildomas papildomais baitais jeigu to prireikdavo. Skirtingi užpildymo būdai pateikti 1.4. skyrelyje, čia vėl patogumo dėlei pasirinkti *0x00* baitai užpildymui. Praktikoje gali būti taikomi ir kiti metodai. Toliau pateikiamas pavyzdys žodžio „MATRICA“ įdėto į 3x3 matricą, kairėje turimos raidės surašytos į matricą, o dešinėje vaizdavimas šešioliktainiu formatu su užpildu *0x00*.

M	A	T
R	I	C
A		

0x4d	0x41	0x54
0x52	0x49	0x43
0x41	0x00	0x00

6 pav. Pradinio pranešimo vaizdavimas ir jo išraiška šešioliktainiu pavidalu su užpildu

3. Rezultatai

Šiame darbe pateikiami CBC ir CTR šifravimo režimų palyginimai. Jie atlikti tiek paveikslams, tiek tekstiniams pranešimams. Siekiant įvertinti šifrogramos kokybę su pasirinktinu šifravimo režimu, žiūrima į elementų skirstinį ir gautąjį vaizdą. Tikslas turėti tolygiai pasiskirsčiusius elementus, užtikrinus šią sąlygą galima būtų teigti, kad kiekvieno elemento pasirodymo tikimybė yra vienoda, neleidžianti rasti dėsningumų ar naudingos informacijos apie tekstogramą turint šifrogramą. Nereiktų pamiršti ir gautosios šifrogramos atvaizdavimo, kadangi nors ir šifrograma gali atrodyti artima tolygiajam skirstiniui, tačiau žiūrint į gautąjį rezultatą galima pastebėti kontūrus ar kitą informaciją apie pradinį pranešimą. Papildomai buvo žiūrima į šifravimo laikus ir bandoma nustatyti, kurie šifravimo algoritmai yra pranašesni. Visi tyrimo rezultatai palyginti su AES-128 šifravimo algoritmu.

Svarbu atkreipti dėmesį, kad šifruojant spalvotą paveikslą, šie yra sudaryti iš 3 spalvų kanalų, t. y. kiekviena paveikslas sudarytas iš raudonos, žalios ir mėlynos spalvos tokių pačių dimensijų matricų. Taigi norint užšifruoti spalvotą paveikslą, tai darome 3 kartus kiekvienai spalvai. Yra įvairių būdų kaip tai galima padaryti, pavyzdžiams esantiems toliau turime 3 kartus daugiau blokų negu būtų turima pilko paveikslo atveju. Taip pat reikia atkreipti dėmesį, kad turint šifravimo algoritmą, kuris sugeneruoja didesnius elementus negu 255 paveikslų vaizdavimui, šie turi būti redukuojami norint išgauti rezultatą, tačiau neredukuotos šifrogramos yra naudojamos tekstogramoms atstatyti. Šiame skyriuje pateikti paveikslai buvo šifruojami nagrinėjant jų RGB formatus, t. y. šifruojant visus 3 spalvų kanalus.

Visi 3 skyriuje pateikiami rezultatai įvertinami naudojant 32 bitų Python 3.10.9 programinę įrangą, o skaičiavimai atlikti naudojant kompiuterį su parametrais:

– Procesorius: Intel(R) Core(TM) i7-8750H 2.20GHz

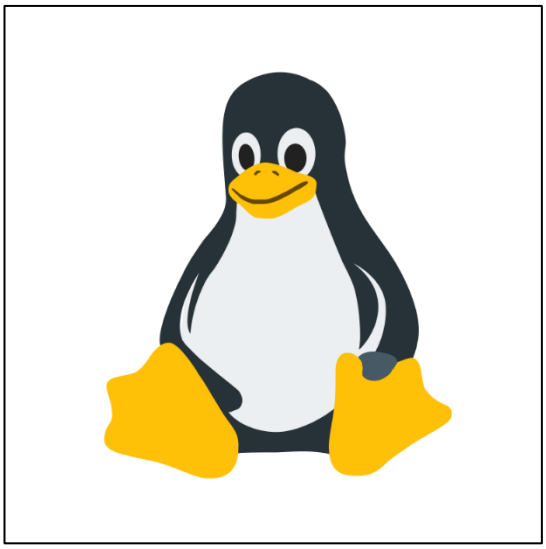
– Operatyvioji atmintis: 24 GB

– Vaizdo plokštė: NVIDIA GeForce GTX 1050Ti

– Operacinė Sistema: Windows 64 bitų

3.1. Paveikslų šifrogramų elementų pasiskirstymas

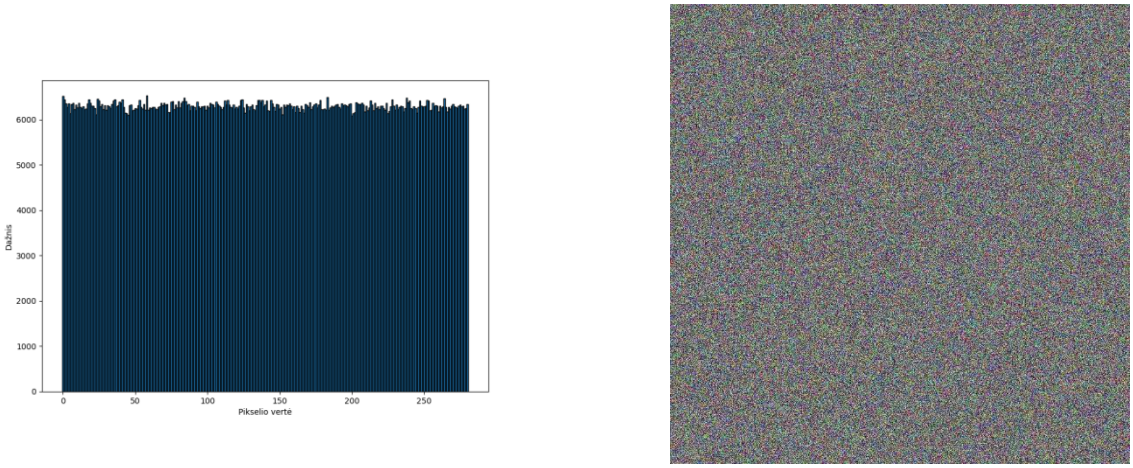
Šifrogramos elementų pasiskirstymui nustatyti buvo pasirinktas paveikslas 7 pav. Daugiau pavyzdžių su skirtingais paveikslais pateikiama 1 priede. Pasirinkto paveikslo išmatavimai lygūs 768x768 pikselių. Pasirinkus bloko eilę lygią 4, iš viso gauta 110592 blokų. Užšifravus ir iššifravus šiuos elementus jie yra atstatomi tokia pačia tvarka, t. y. žiūrint į užšifruotą ir iššifruotą vaizdus, atitinkami vaizdų elementai yra tose pačios vietose. Siekiant įvertinti šifrogramų elementų pasiskirstymų dėsningumus, pasirinkta mažesnis paveikslas turinti mažesnį kiekį pikselių, priede pateikiami papildomi pavyzdžiai šifruojant skirtingų matmenų ir dydžių paveikslus, tačiau didelių skirtumų tarp tiriamų skirtingų objektų nebuvo pastebėta.



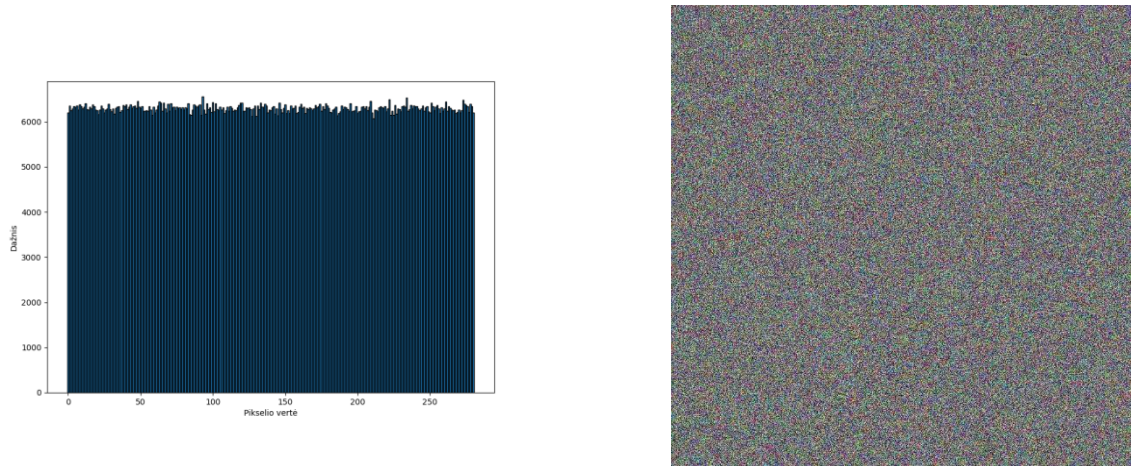
7 pav. Pradinis paveikslas

3.1.1. CBC šifravimo režimas

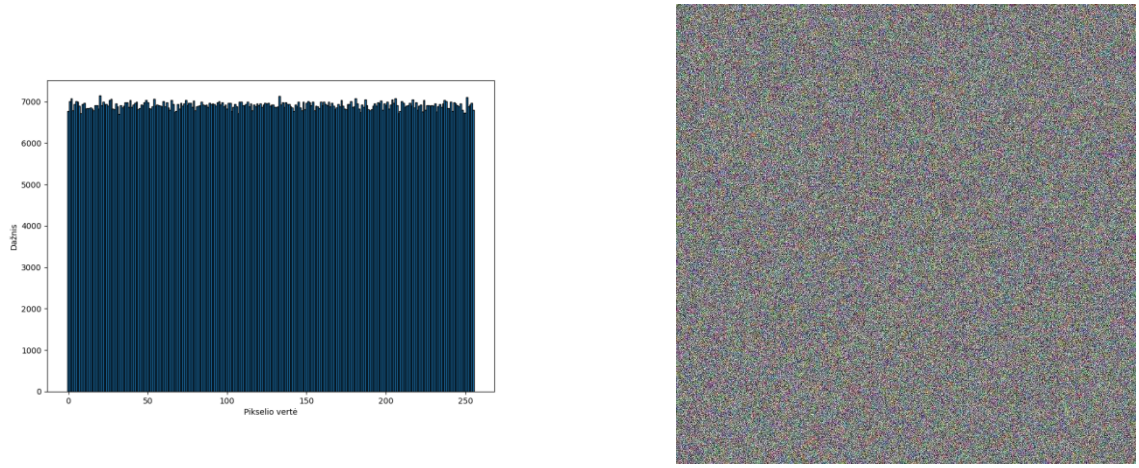
Taikant šį šifravimo režimą buvo taikytas tiek sumavimas taikant XOR operaciją, tiek elementų suma moduliu sudedant prieš tai buvusio bloko šifrogramą su toliau esančio bloko tekstograma. Galima pastebėti (8-11 pav.), kad tiek taikant vieną tiek taikant kitą sudėtį, gaunami rezultatai ženkliai nesiskiria. Panašu, kad žingsnio, kuriame sudedama prieš tai buvusi šifrograma su toliau esančia šifrograma dėka, nematomas skirtumas taikant skirtingus šių dviejų elementų sudėties metodus. Tiek taikant šifravimo algoritmą paremtą komutatyvia grupe, tiek taikant šifravimo algoritmą paremtą nekomutatyve grupe didelių skirtumų nebuvo pastebėta. Kiekvienu variantu, pasirinkto bloko dydis buvo fiksuotas, t. y. pasirinkta 4 eilės matrica.



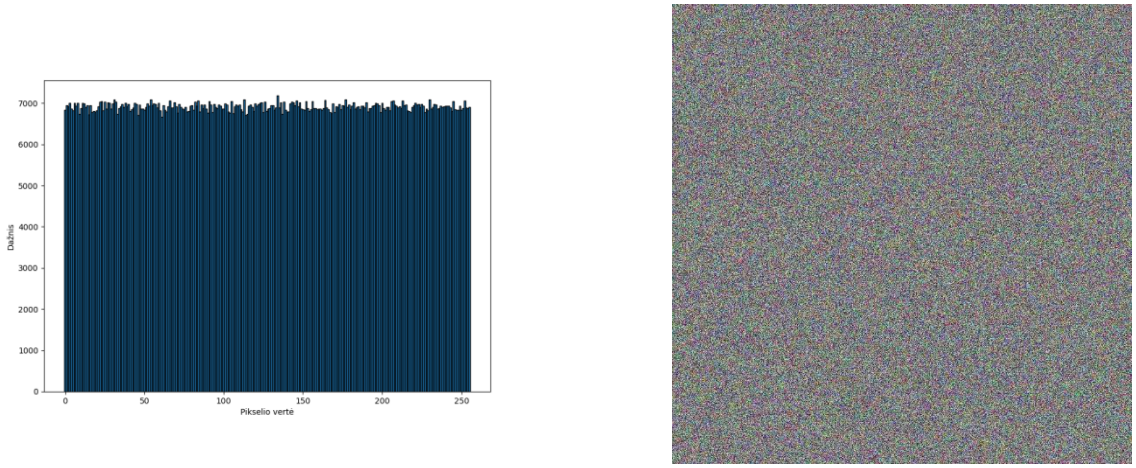
8 pav. Rezultatai gauti taikant šifravimo režimą paremtą komutatyvia grupe CBC režimu, naudojant 4 eilės blokus ir atliekant sumavimą taikant XOR operaciją



9 pav. Rezultatai gauti taikant šifravimo režimą paremtą komutatyvia grupe CBC režimu, naudojant 4 eilės blokus ir atliekant elementų sumavimą moduliu q

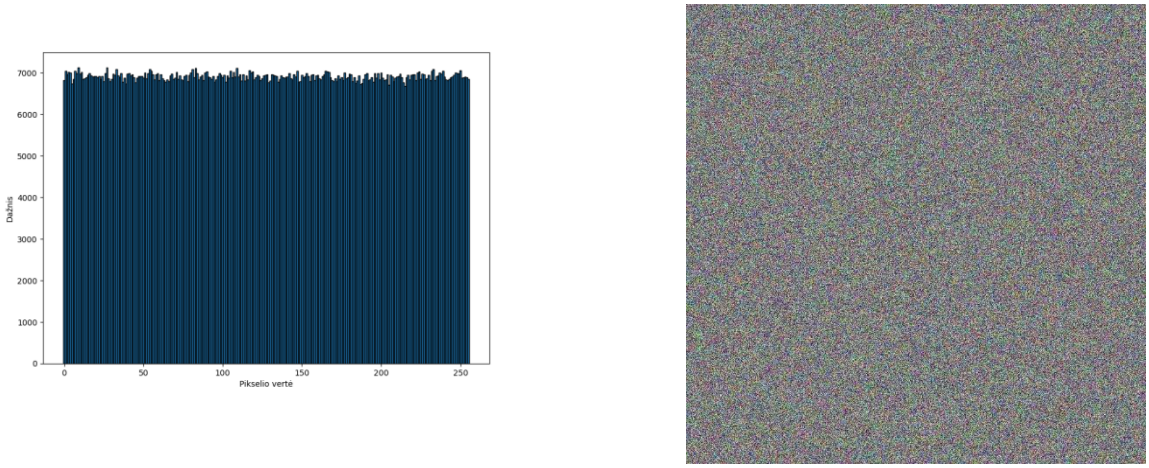


10 pav. Rezultatai gauti taikant šifravimo režimą paremtą nekomutatyvia grupe CBC režimu, naudojant 4 eilės blokus ir atliekant sumavimą taikant XOR operaciją



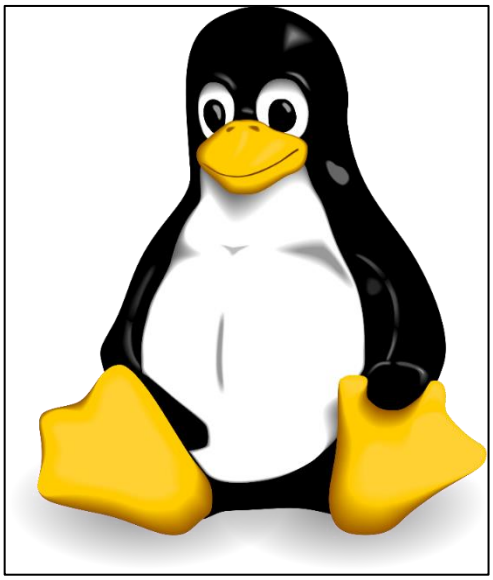
11 pav. Rezultatai gauti taikant šifravimo režimą paremtą nekomutatyvia grupe CBC režimu, naudojant 4 eilės blokus ir atliekant elementų sumavimą moduliu 2^t

Taikant AES-128 šifravimo algoritmą gaunami rezultatai pateikiami 12 pav. Čia gaunami rezultatai neišsiskiria nuo aukščiau pateiktųjų.



12 pav. Rezultatai gautu taikant AES-128 šifravimo algoritmą CBC režimu

Pavyzdžiams viršuje buvo pasirinktas sąlyginai mažas paveikslas, siekiant įvertinti šifrogramos elementų pasiskirstymą, taikant neparametrinę Chi kvadrato hipotezę, pasirinktas didesnis paveikslas 2000x2352 pikselių (13 pav.), gaunant 882000 blokų, kai pasirenkama 4 eilės bloko matrica.



13 pav. Pradinis paveikslas

Turint šį paveikslą suformuojama neparametrinė Chi kvadrato hipotezė, kur nulinė hipotezė teigia, kad elementai pasiskirstę tolygiai, o alternatyvi hipotezė teigia, kad elementai nėra pasiskirstę pagal tolygųjį skirstinį. Atlikę šį testą kiekvieno šifravimo algoritmo atveju ir gauti rezultatai pateikiami 1 lentelėje.

2 lentelė. Chi kvadrato hipotezės p-reikšmės skirtingiems šifravimo algoritmams

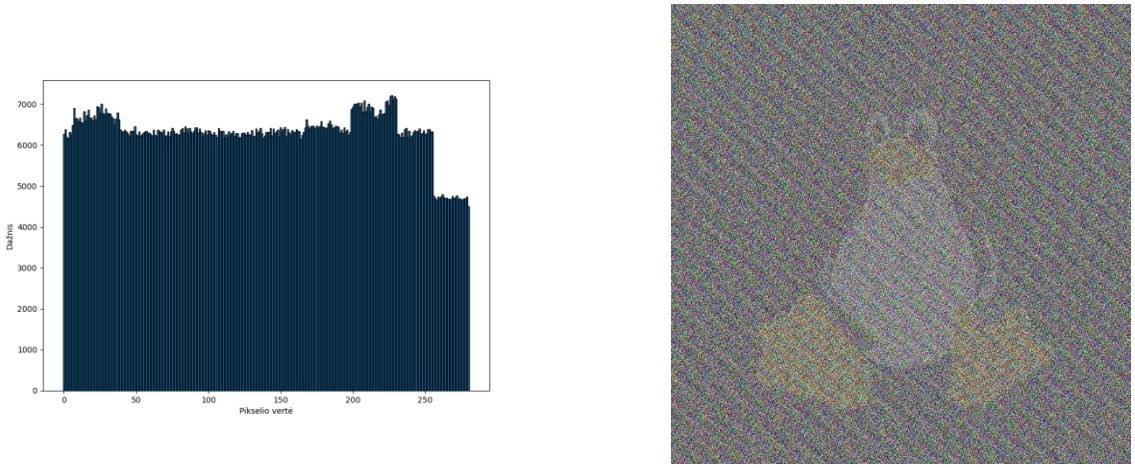
Šifras	Chi kvadrato testo p-reikšmė
Šifras paremtas komutatyvia grupe taikant sumą moduliu q CBC režimu	0.192841
Šifras paremtas nekomutatyvia grupe taikant sumą moduliu 2^t CBC režimu	0.926513
Šifras paremtas komutatyvia grupe taikant XOR operaciją CBC režimu	0.366800
Šifras paremtas nekomutatyvia grupe taikant XOR operaciją CBC režimu	0.457409
AES-128 CBC režimu	0.051877

Remiantis gautosios lentelės rezultatais galima pastebėti, kad visais atvejais hipotezė yra priimtina su reikšmingumo lygmeniu $\alpha = 0.05$, tai leidžia su 95 % garantija teigti, kad elementai gautosiose šifrogramose yra pasiskirstę tolygiai.

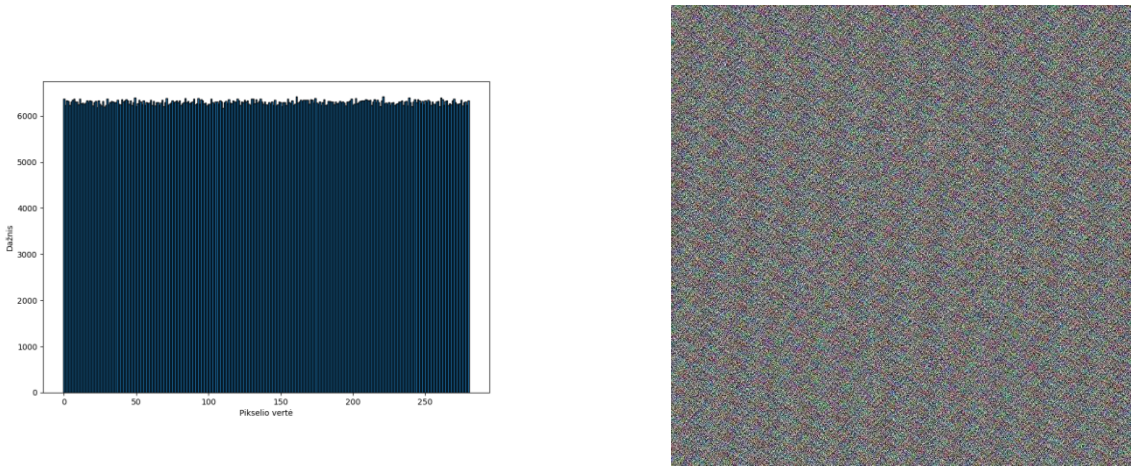
Gautųjų rezultatų buvo tikimasi, kadangi [4, 5] teoriškai pagrįsti puikaus saugumo įrodymai, leidžiantys teigti, kad turint šifrogramas nieko negalima pasakyti apie tekstogramas. Tuo buvo įsitikinta atliekant Chi kvadrato testą kiekvienam skirtingai šifravimo metodikai.

3.1.2. CTR šifravimo režimas

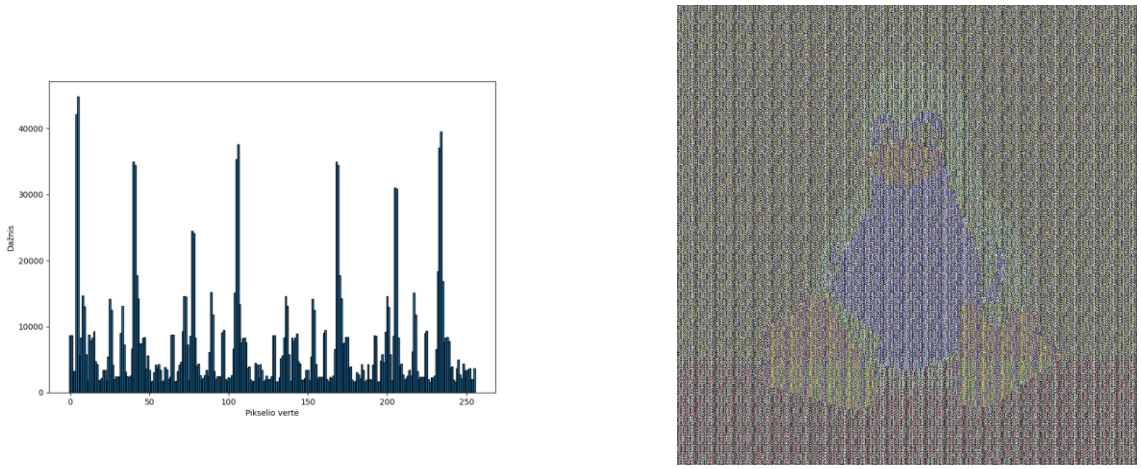
Šiam režimai analogiškai atlikti tokie patys tyrimai kaip ir CBC režimo atveju. Gauti rezultatai taikant tiek XOR operaciją, tiek sudėtį grupės eilės moduliu, šie atitinkamai pateikiami 14-17 pav. Lyginant gautuosius rezultatus, pastebima daugiau netolygumo ir netgi pastebimas pačio pradinio paveikslo vaizdas žiūrint į šifrogramas. Šio rezultato siekiame kiek įmanoma vengti, kadangi šifro tikslas yra paslėpti pradinę informaciją. Skirtingai nei CBC režime, čia galime pastebėti, kad sudėtis grupės eilės moduliu duoda geresnius rezultatus, ši užtikriną pakankamą elementų sumaišymą ir atvaizduojant šifrogramą užtikrina, kad pradinė informacija nėra atskleidžiama kaip taikant XOR operaciją (14 - 15 pav.)



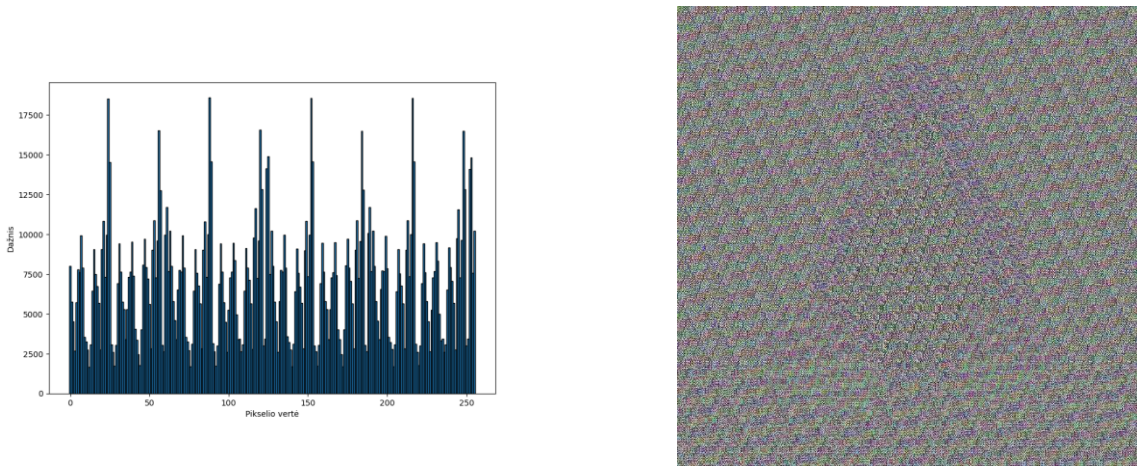
14 pav. Rezultatai gauti taikant šifravimo režimą paremtą komutatyvia grupe CTR režimu, naudojant 4 eilės blokus ir atliekant sumavimą taikant XOR operaciją



15 pav. Rezultatai gauti taikant šifravimo režimą paremtą komutatyvia grupe CTR režimu, naudojant 4 eilės blokus ir atliekant elementų sumavimą moduliu q

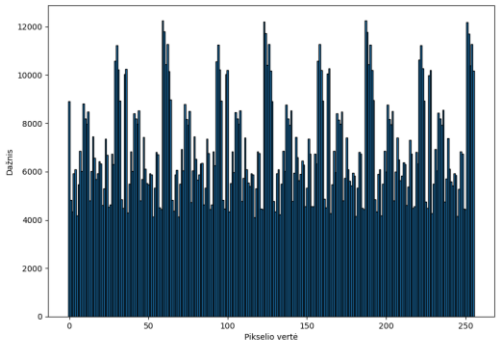


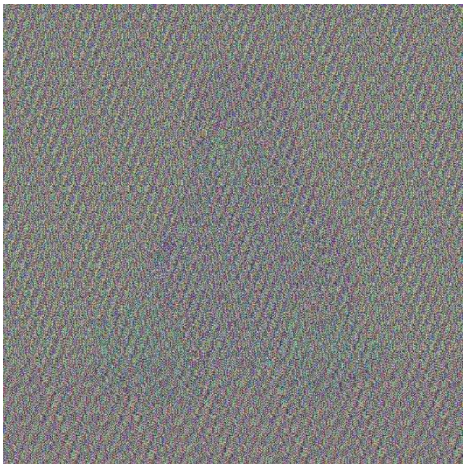
16 pav. Rezultatai gauti taikant šifravimo režimą paremtą nekomutatyvia grupe CTR režimu, naudojant 4 eilės blokus ir atliekant sumavimą taikant XOR operaciją



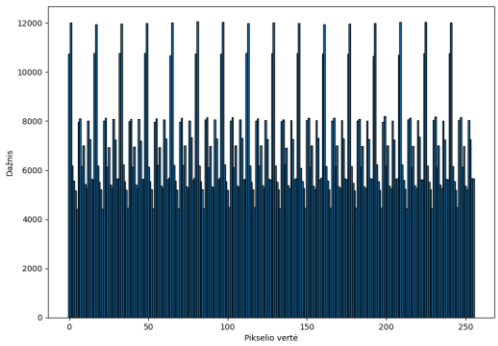
17 pav. Rezultatai gauti taikant šifravimo režimą paremtą nekomutatyvia grupe CTR režimu, naudojant 4 eilės blokus ir atliekant elementų sumavimą moduliu 2^t

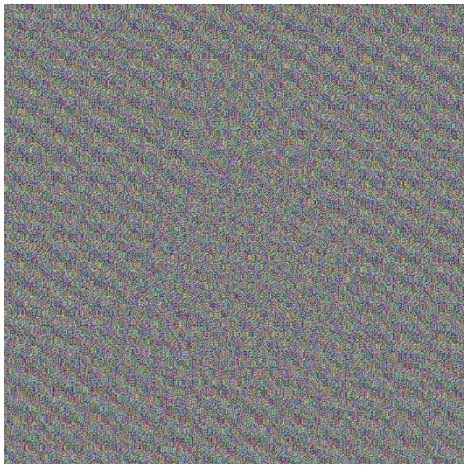
Šifravimo režimas paremtas komutatyvia grupe su suma grupės eilės moduliu duodą gražų sprendinį, t. y. šifrogramos elementų pasiskirstymas artimas tolygiajam ir atvaizduojama šifrograma neatskleidžia informacijos apie pradinį pranešimą, tačiau to negalima pasakyti apie šifravimo algoritmą paremtą nekomutatyvia grupe, matomi netolygumai žiūrint į šifrogramų histogramas, taip pat matomi pradinio paveikslo dalis. Tačiau šio rezultato buvo tikimasi, atliekant lavinos efekto tyrimus šiam šifravimo režimui buvo pastebėta, kad tik skaitiklio keitimas nėra pakankamas šiam algoritmui, kad būtų užtikrintas elementų sumaišymas duodantis siektiną rezultatą. Siekiant pagerinti turimą rezultatą, atliekamos papildomos šifro modifikacijos leidžiančios tai padaryti. Kadangi CTR šifravimo režime naudojama tik šifravimo funkcija, čia nėra būtinybės užtikrinti, kad matrica Y turėtų atvirkštinę, todėl galima atsisakyti šios matricos apribojimo ir ją generuoti atsitiktinai. Pašalinus šį apribojimą gaunami rezultatai pateikiami 18-19 pav.





18 pav. Rezultatai gauti taikant šifravimo režimą paremtą nekomutatyvia grupe CTR režimu, naudojant 4 eilės blokus ir atliekant sumavimą taikant XOR operaciją, bei pašalinus matricos Y apribojimą

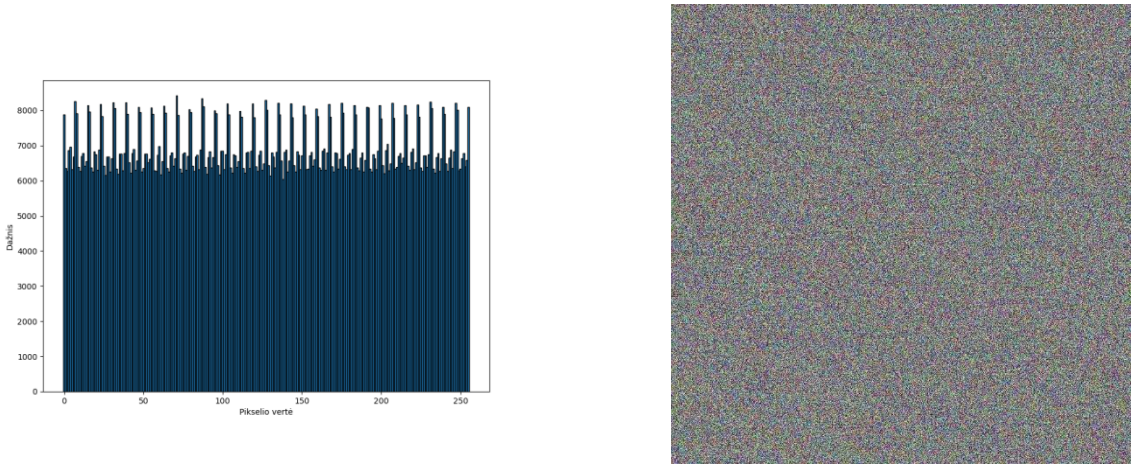




19 pav. Rezultatai gauti taikant šifravimo režimą paremtą nekomutatyvia grupe CTR režimu, naudojant 4 eilės blokus ir atliekant elementų sumavimą moduliu 2^t, bei pašalinus matricos Y apribojimą

Lyginant rezultatus galima pastebėti, kad elementų pasiskirstymas tapo tolygesnis, tačiau pastebimi dėsningumai šifrogramos elementų skirstinyje, taip pat į gautame paveikslę galima įžvelgti švelnius kontūrus leidžiančius identifikuoti pradinį paveikslą. Galime pastebėti, kad naudojant sumą grupės eilės moduliu, rezultatai gaunami neženkliai, tačiau geresni. Siekiant dar labiau pagerinti gaunamus

rezultatus, panaudojamas sumaišymas apibrėžtas 2.4.2. skyrelyje. Atlikę šį vaizdavimą ir naudojant tokią pačia režimo struktūra su suma grupės eilės moduliu gaunami rezultatai pateikiami 20 pav.

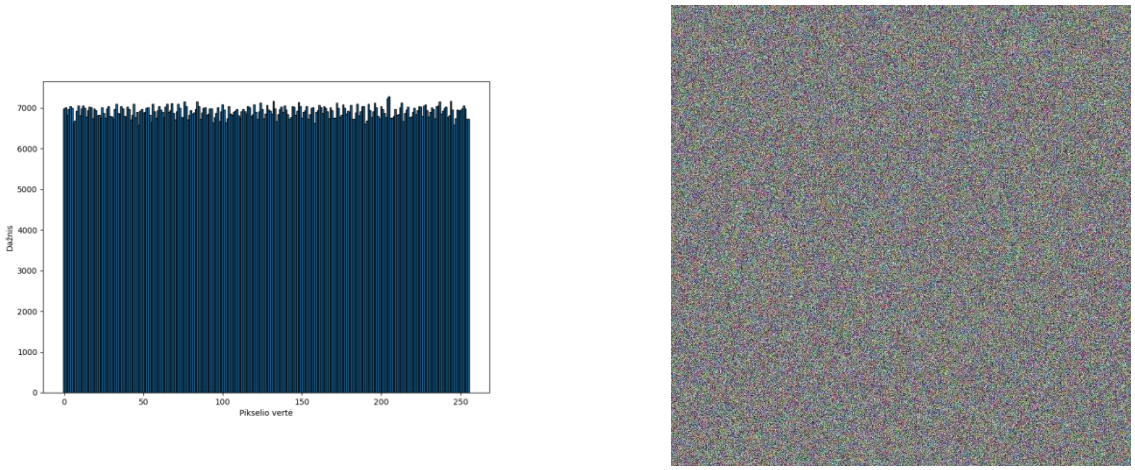


20 pav. Rezultatai gauti taikant šifravimo režimą paremtą nekomutatyvia grupe CTR režimu, naudojant 4 eilės blokus, atliekant elementų sumavimą moduliu 2^t su pašalintu matricos Y apribojimu, bei elementų sumaišymu

Galima pastebėti, kad rezultatai kur kas geresni, gautame vaizde nebeįžvelgiami kontūrai ir matomas tik triukšmas. Tačiau žiūrint į elementų pasiskirstymą galime pastebėti dėsningumus, kuriuos siekiame panaikint, kad šie nebūtų panaudojami turimo šifro nulaužimui. Todėl įvedama dar viena papildoma modifikacija šiai problemai spręsti. Pagrindinė problema, kurią stengiamės pašalinti yra elementų sumaišymas, žinoma, kad siekiant užtikrinti pakankamą elementų sumaišymą, lavinos efekto įvertis turėtų būti artimas 0,5, kas leistų teigti apie elementų tolygųjį pasiskirstymą, tačiau lyginant su CTR režimu matome, kad tik skaitiklio keitimas nedaro pakankamai įtakos sumaišymui. Todėl išbandyta papildoma modifikacija, kas turėtų užtikrinti pakankamą sumaišymą. Praeitame žingsnyje sumaišymas paliekamas, tačiau pateikiamas kamšalas su bloko indeksu yra apibrėžiamas kitaip, t. .y kamšalas su skaitikliu nėra sujungiamas, tačiau juos atitinkančios matricos yra sudedamos, gaunant naująją įvesties matricą.

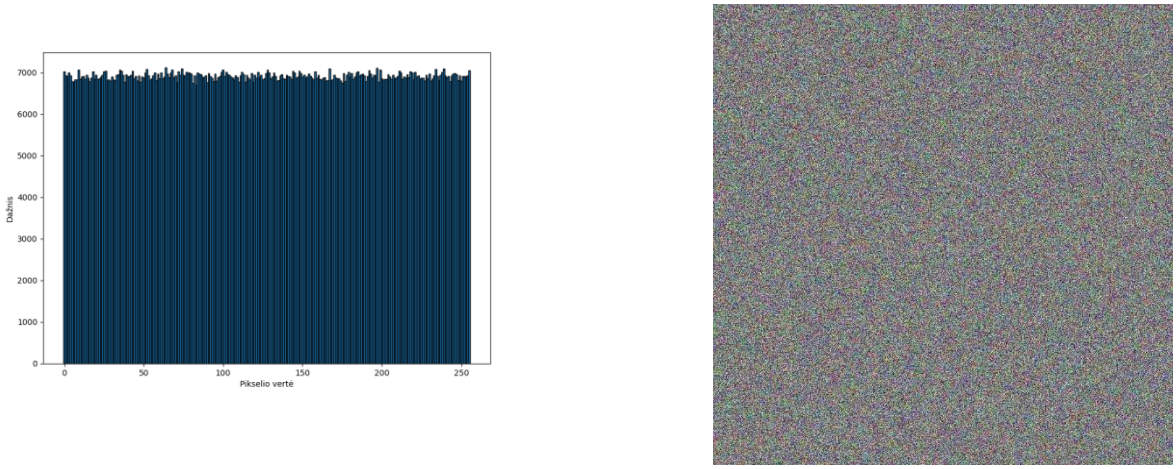
$$\eta^{(i)} = Nonce + F_\zeta(Cntr^{(i)})$$

Čia kamšalo matrica sugeneruojamas atskirai ir pridedama skaitiklio matrica pasinaudojant F_ζ vaizdavimu. Reiktų atkreipti dėmesį, kad čia skaitiklis Cntr yra atskira matrica, į kurią kiekvieno bloko metu pridedamas vienetas į pasirinktą poziciją, tarkime apatinį dešinįjį kampą ir kiekviename bloke dėl vaizdavimo turėsime skirtingą konfigūraciją, kuri paveiks ne tik paskutiniuosius elementus. Žinoma, elemento pridėjimas gali būti sutartas kitoks, pavyzdžiui pridėti elementą prie viršutinio kairiojo matricos elemento. Gauti rezultatai atlikę visus minėtuosius pakeitimus pateikiami 21 pav.



21 pav. Rezultatai gauti taikant šifravimo režimą paremtą nekomutatyvia grupe CTR režimu, naudojant 4 eilės blokus, atliekant elementų sumavimą moduliu 2^t su pašalintu matricos Y apribojimu, bei elementų sumaišymu kartu su pridedamo skaitiklio vaizdavimu

Taikant AES-128 šifravimo algoritmą CTR šifravimo režimu, gaunami rezultatai pateikiami 22 pav.



22 pav. Rezultatai gauti naudojant AES-128 šifravimo algoritmą CTR režimu

Lyginant visus tris rezultatus galime pastebėti, kad šie užtikriną pakankamą elementų sumaišymą neatskleisdami plika akimi pastebimų pradinės tekstogramos informacijos, bei užtikriną šifrogramos elementų pasiskirstymą artimą tolygiajam. Tačiau siekiant gauti tokius rezultatus reikia užtikrinti, kad šifras paremtas komutatyvia grupe sudėtų elementus (gautą šifrogramą su tekstograma) taikant elementų sumą grupės eilės moduliu, o šifras paremtas nekomutatyvia grupe papildomai atsisakytų matricos Y apribojimų, bei naujai apibrėžto įvesties elemento su papildomais vaizdavimais, kurie užtikrina, kad siektinos savybės būtų tenkinamos.

Turint paveikslą pateiktą 13 pav., suformuota neparametrinė Chi kvadrato hipotezė, kur nulinė hipotezė teigia, kad elementai pasiskirstę tolygiai. Šiam režimui p–reikšmės tikrintos tik su elementų suma grupės eilės moduliu, nes su XOR operacija rezultatai gaunami ne tokie geri ir plika akimi matomas skirstinių netolygumas, todėl nėra prasmės įvertinį p-reikšmės šiems scenarijams. Atlikę šį testą kiekvieno šifravimo algoritmo atveju gauti rezultatai pateikiami 2 lentelė.

3 lentelė. Chi kvadrato hipotezės p-reikšmės skirtingiems šifravimo algoritmams

Šifras	Chi kvadrato testo p-reikšmė
Šifras paremtas komutatyvia grupe taikant elementų sumą moduliu q CTR režimu	0.999999
Šifras paremtas nekomutatyvia grupe taikant elementų sumą moduliu 2^t CTR režimu, pašalinus Y apribojimą ir įvedant bloko indekso vaizdavimą*	0.653328
AES-128 CTR režimu	0.393360

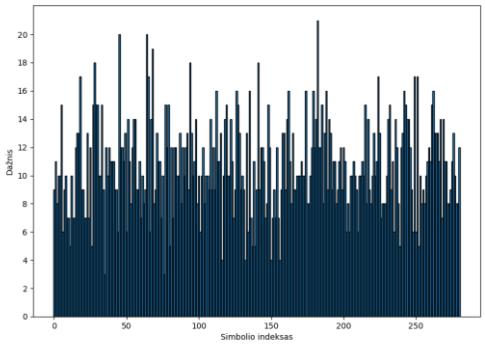
Remiantis gautosios lentelės rezultatais galima pastebėti, kad visais atvejais hipotezė yra priimtina su reikšmingumo lygmeniu $\alpha = 0.05$, tai leidžia su 95 % garantija teigti, kad elementai gautosiose šifrogramose yra pasiskirstę tolygiai.

3.2. Teksto šifravimas

Teksto šifravimas atliekamas tokiu pačiu principu kaip ir paveikslams ankstesniame skyrelyje. Toliau pateikti rezultatai gaunami šifruojant sugeneruotą tekstą taikant [31] pateiktą įrankį. Tai nėra autorinių teisių saugomas sugeneruotas tekstas. Sugeneruotą tekstogramą sudaro 2988 simboliai (taikant 4 eilės blokus, turimi iš viso 187 blokai) ir ji yra pateikta 2 priede. Pirmiausia nagrinėjamas UTF-8 formatu, nors šio formato elementai gali būti koduojami nuo 1 iki 3 baitų, tačiau pasirinktoje šifrogramoje visi simboliai koduojami 1 baitu. Todėl šifravimo algoritmų parametrus galima pasirinkti tokius pačius, kurie buvo naudojami šifruojant paveikslus, taip pat šifravimo režimų modifikacijas, t. y. pasirinkta sudėti atitinkamu grupės moduliu vietoj bitų sudėties moduliu 2 (taikant XOR operaciją). Šifrograma yra gaunama 8 bitų elementą konvertavus į UTF-8 simbolį, šie sutampa su išplėstiniu ASCII elementų išdėstymu. Toliau pateikiami pavyzdžiai taikant skirtingus šifrus su skirtingais šifravimo režimais. Dėl mažo turimos tekstogramos ilgio, gautieji rezultatai neleidžia objektyviai įvertinti šifrogramos elementų pasiskirstymo, tačiau galima atlikti papildomą modifikaciją leidžiančia sumažint naudojamų bitų skaičių kiekvienam matricos elementui per pusę, taip leidžiant naudoti mažesnę naudojamos grupės eilę. Šis metodas gali būti taikomas taip pat, jeigu naudojamos sistemos resursai yra riboti, kadangi sumažinus naudojamos grupės eilę, taip pat sumažėja ir atminties kiekis reikalingas saugoti naudojamus bendruosius raktus ir parametrus reikalingus atlikti šifravimą.

3.2.1. Teksto šifravimas taikant CBC režimą

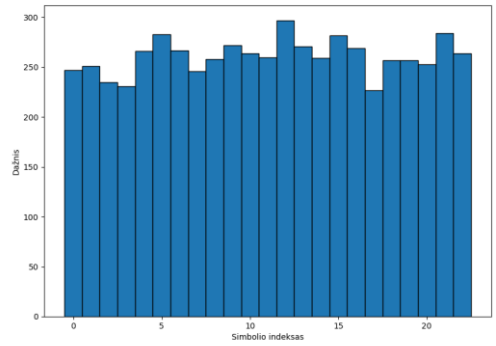
25-26 pav. pateikiama užšifruotos tekstogramos histograma taikant šifrą paremtą komutatyvia grupe CBC režimu. Galima pastebėti, kad turint trumpą pranešimą, bei šiuo atveju pasirinkus parametrus $p = 563$, $q = 281$ gaunamas netolygus skirstinys, kurio elementų dažniai yra labai reti. Tačiau nepaisant to gautosios šifrogramos elementai nėra įskaitomi. Čia reikėtų atkreipti dėmesį, kad taip pat kaip ir šifruojant paveikslus turint pirminio skaičiaus grupės eilę, norint korektiškai atvaizduoti elementus, elementai pirmiausia redukuojami ir tik tada atvaizduoti, taip kaip buvo daroma paveikslams ankstesniame 3.1. skyrelyje.



Q ¬
¼4
}æß!O-Z {Ã-mn±}¹DÈb~ ùã³b9+iõÙá{I_§àH%-~`TÅÛ'Q ¨=4z¥ý¡_£àî
lìË¤ ¢1ra Ð~?+IßEá
T8~T5ˆÔ@z-KÝ hÅÐÈ>vÕ,Y^ôÉÝ«Zú³4@u¶
¶^£à\ºC(_}'xuilïC ÙüÙ<{G¢wX£S¹Ñ/+ É<Uk

23 pav. Elementų histograma ir šifrogramos fragmentas taikant šifravimo režimą paremtą komutatyvia grupe CBC režimu, naudojant 4 eilės blokus ir atliekant elementų sumavimą moduliu q

Turint trumpą pranešimą, šio elementų pasiskirstymas yra pakankamai retas, todėl siekiant užtikrinti geresnį elementų sumaišymą trumpiems pranešimams yra įvedama modifikacija. Prieš šifruojant pranešimą kaip pateikta 2.4.1. skyrelyje kiekviena matricos įvestis iš 8 bitų padalinama į du po 4. Atlikę šią modifikacija galima naudoti mažesnę grupės eilę pranešimui šifruoti. Pasirinkus parametrus $p = 47$, $q = 23$, gaunama šifrogramos histograma

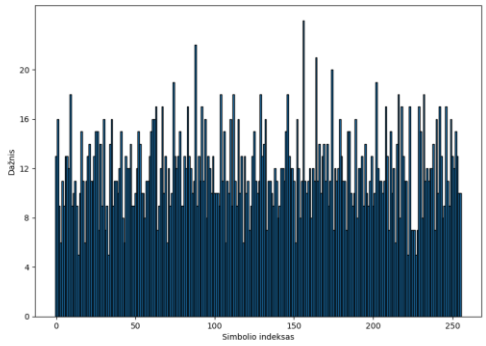


òìbeJTói.ô],Q0b:Dà QSG\TÔÞ$4ÃãÒUÚZÚ& dí]
#w|¶ÈAKòpÁ¼SE3}òFBzbVj¼6J¢aRC¾ù6ºÌà³Ue.$R Äîg
v".ÞÉ¦VßNR&Zdó!(£4 F@tS5%D]¼uÙ[*Å<æ"Æç S©@ñU
ñÖ`1%x0 .Àµ¼FUfÑ:ú%ºXÞ]Ýpnr®SCkÓlTéPÂºu!q;ÕÑ1dã´v
$¯F@`0ØæVôøs59Qe]~B`¥âltN6*SlÖIeüsx ¦¡eDÇ¯+3013U!

24 pav. Rezultatai gauti taikant šifravimo režimą paremtą komutatyvia grupe CBC režimu, naudojant 4 eilės blokus ir atliekant elementų sumavimą moduliu q 4 bitų elementams

Šiuo atveju elementų dažnis yra didesnis, bei matoma aiškesnis elementų pasiskirstymas, kaip ir prieš tai turimu atveju, iš gautosios šifrogramos nieko negalime pasakyti apie pradinį pranešimą.

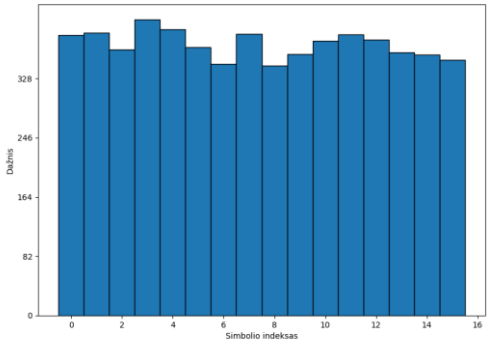
Šifruojant tą patį pranešimą taikant šifrą paremtą nekomutatyvią grupe CBC režimu, matome panašią situaciją, kai elementų dažnis yra mažas ir pasiskirstymas nėra panašus į tolygųjį.



ÿª)/´Ì0°è~3·¡ p±@Á¥w3öÈ
ø¹4òM¿ä30ú XDó´ H7
ÜQwMª)=H¿IÅh(C¥ ÏÑ§Õ9Ã§à×Ê8×^ÕiÉñ³4ÊÔHæ¢³u¶£¤è'
²íþå WÚò- dm6
ð]e ÂÌ{Ú4Íaßé ¤Y8 jÇïJ¦"öî:= dx óÒ O ÉD ø Ã

25 pav. Rezultatai gauti taikant šifravimo režimą paremtą nekomutatyvia grupe CBC režimu, naudojant 4 eilės blokus ir atliekant elementų sumavimą moduliu 2^t 8 bitų elementams

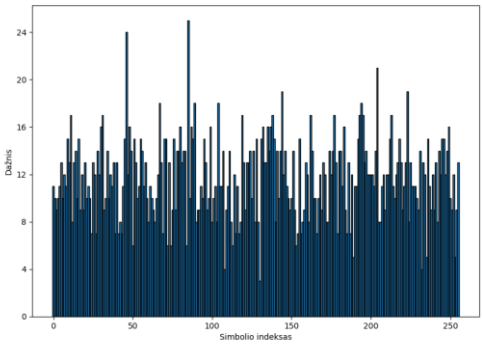
Išskaidę elementus po 4 bitus, matomas gražesnis vaizdas, kur 16 grupės elementų turi didesnį dažnį, bei skirstinys artimesnis tolygiajam. Reikia atkreipti dėmesį, kad skirtingai nei naudojant šifrą paremtą komutatyvia grupe, čia elementų skaičius pilnai užpildo 4 bitus ir nereikia atlikti papildomų veiksmų norint atvaizduoti gautąjį rezultatą. Tačiau lyginant su 25-26 pav. gautasis rezultatas labai panašus.

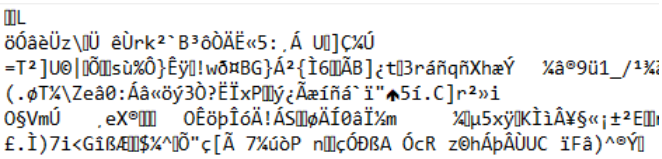


1úê0 ä÷\o¤ Ùû ài¿eãXÙ?K³"gIWS;¥Ð «_>Ê+éÃOUäæKCyûW?¶¤ëjË(
\4p GX ý'ê
êg)'¼ ¡k½ Fû Ø¬Ëó0F¶khùç MÞÁg ìbRº¡ e;± T3 ¿[è-!³@dÛÐ M+Wp ¥09Ä ÛL \
ëP VEWX-À@¤ã¤ ¢µ/·¢"ÿ âáË3Ðr'à? ç Y\ ÇZwÔqÂêöT
Þï Ö¸'A>øÁÀQ!UjãK%ßH\ux×Sv í°ç `UHOOä>ØÏÛè Ï:7a«9eìb-!Ç hD,
ó"êtºÕøÐ³üå ÞX¿óPù¤ÁOÖ¢z>ÚEn¥~Mê(§áW»êexh0q Ö) Ã`Bîú ÔK:d£ =

26 pav. Rezultatai gauti taikant šifravimo režimą paremtą nekomutatyvia grupe CBC režimu, naudojant 4 eilės blokus ir atliekant elementų sumavimą moduliu 2^t 4 bitų elementams

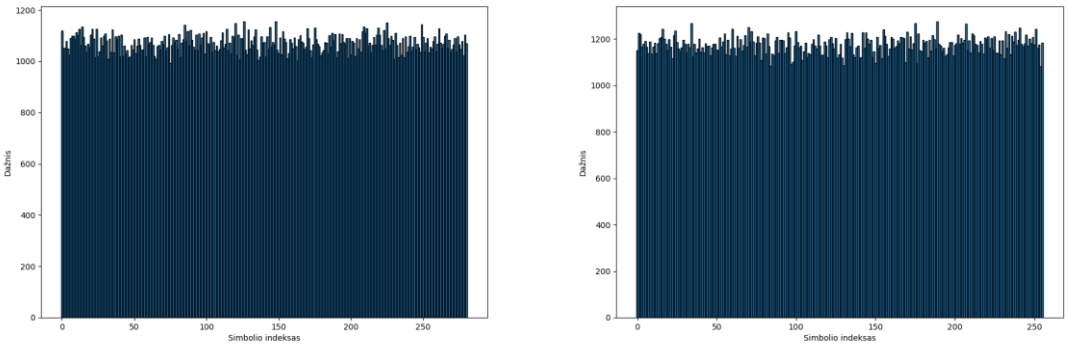
Toliau pateikiami rezultatai gauti, užšifruojant tekstinį pranešimą taikant AES-128 algoritmą CBC režimu. Matoma ta pati tendencija, kur turimas mažas elementų dažnis, tačiau šiame algoritme elementų dydžio negalime pakeisti.



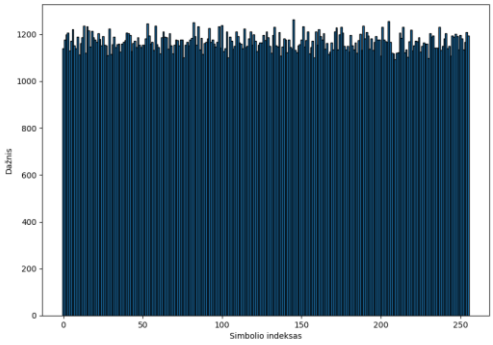


27 pav. Rezultatai gauti taikant AES-128 šifravimo algoritmą CBC režimu

Šifruojant tą patį pranešimą su skirtingais raktais ir inicijavimo vektoriais, gaunami rezultatai pateikti 28-29 pav. Čia aiškiai matome, kad kuomet turimas didesnis elementų skaičius, jų pasiskirstymas artėja prie tolygiojo skirstinio.



28 pav. Šifro paremto komutatyvia ir nekomutatyvia grupe šifrogramos elementų skirstiniai su 100 skirtingų raktų

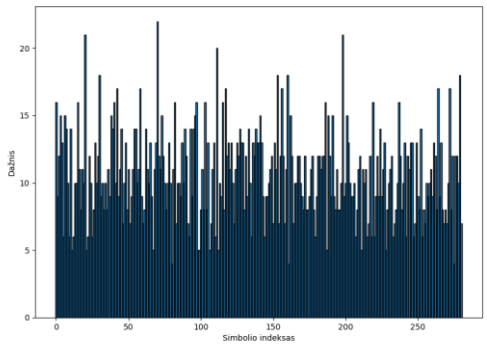


29 pav. AES-128 CBC režimu gautų šifrogramos elementų skirstinys su 100 skirtingų raktų

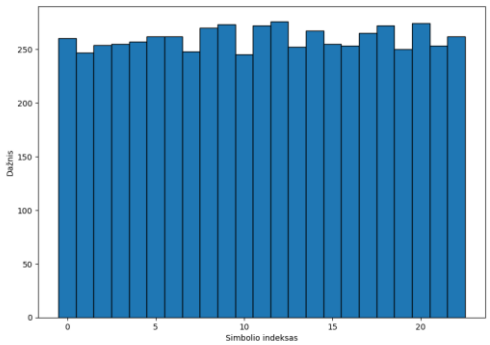
3.2.2. Teksto šifravimas taikant CTR režimą

Analogiškai realizuotas algoritmas paremtas komutatyvia grupe taikant CTR šifravimo režimą (30-31 pav.). Čia vėl matomas panašus rezultatas ir ženkliai nesiskiriantis nuo rezultatų gautų taikant CBC šifravimo režimą. Elementų sumažinimas šiame variante taip pat davė geresnius rezultatus, t. y. elementų skirstinys artimas tolygiam.

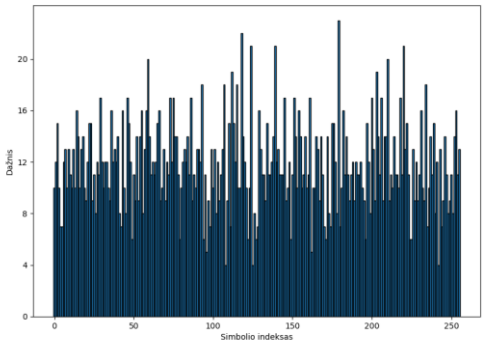
Šifruojant pranešimą šifru paremtu nekomutatyvia grupe buvo taikoma taip pat sudėtis grupės moduliu, atsisakyta matricos Y apribojimų, bei naudojamas skaitiklis vaizduojamas tokiu pačiu būdu kaip apibrėžta 3.1.2. skyrelyje. Šis scenarijus pasirinktas remiantis gautaisiais rezultatais šifruojant ilgus pranešimus. Kaip ir prieš tai buvusiuose pavyzdžiuose matomas panašus vaizdas, kur histograma turi mažai stebinių kiekvienam skirtingam grupės elementui pagal ką negalime objektyviai įvertinti elementų pasiskirstymo. Padalinus elementus po 4 bitus matomas panašus rezultatas (32-33 pav.) kaip ir prieš tai matytuose pavyzdžiuose, kur gautųjų elementų dažnis padidėja, bei skirstinys tampa artimas tolygiajam.



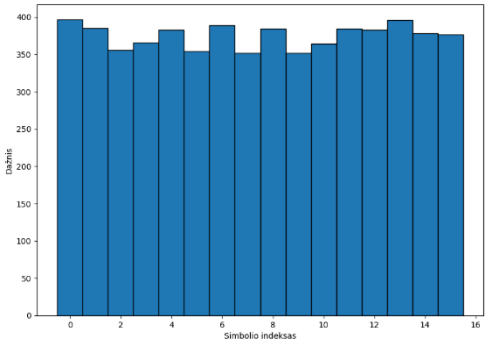
30 pav. Rezultatai gauti taikant šifravimo režimą paremtą komutatyvia grupe CTR režimu, naudojant 4 eilės blokus ir atliekant elementų sumavimą moduliu q



31 pav. Rezultatai gauti taikant šifravimo režimą paremtą komutatyvia grupe CTR režimu, naudojant 4 eilės blokus ir atliekant elementų sumavimą moduliu q 4 bitų elementams

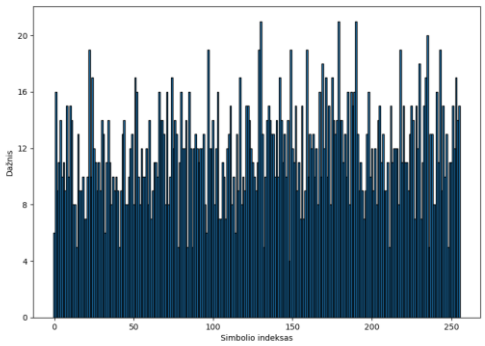


32 pav. Rezultatai gauti taikant šifravimo režimą paremtą nekomutatyvia grupe CTR režimu, naudojant 4 eilės blokus ir atliekant elementų sumavimą moduliu 2^t 8 bitų elementams



33 pav. Rezultatai gauti taikant šifravimo režimą paremtą nekomutatyvia grupe CTR režimu, naudojant 4 eilės blokus ir atliekant elementų sumavimą moduliu 2^t 4 bitų elementams

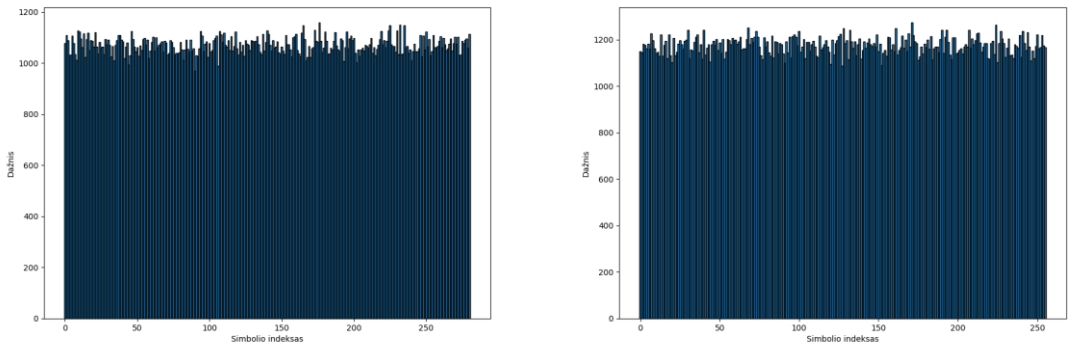
Taikant AES-128 algoritmą CTR režimu tam pačiam pranešimui užšifruoti, matomas skirtingų elementų skirstinys, bei jo netolygumas turint maža elementų dažnį. Skirtingai nei mūsų nagrinėjami algoritmai, čia elementų negalima sumažinti.



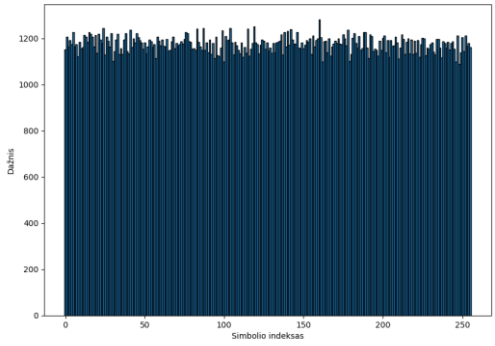
34 pav. Rezultatai gauti taikant AES-128 šifravimo algoritmą CTR režimu

Nors ir matomas šifrogramos elementų netolygumais, jis yra gaunamas tik dėl to, kad turimas pranešimas yra trumpas. Jeigu šifruojame tarkime 100 tokių pačių pranešimų su skirtingais pasirinktais raktais, turime skirstinius šifro paremto nekomutatyvia grupe, šifro paremto komutatyvia grupe ir AES šifravimo algoritmą CTR režimu, galima pastebėti, kad gautieji rezultatai

artimi gautiesiems tiriant paveikslus (35-36 pav.). Todėl elementų išskaidymas siekiant sumažinti grupės eilę būtų naudingas tik tuo atveju, jei naudojamos atminties kiekis yra ribotas ir negalime saugoti atmintyje ilgų raktų.



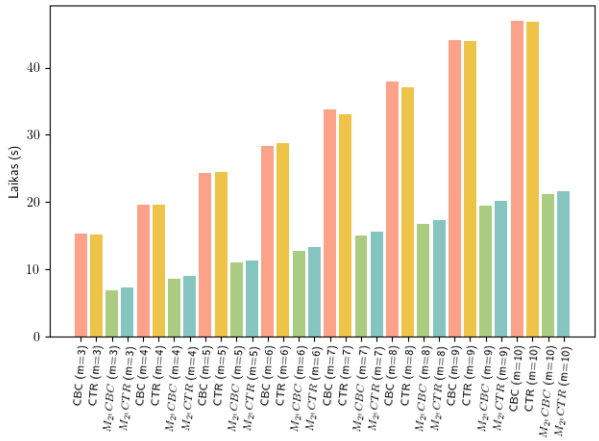
35 pav. Šifro paremto komutatyvia ir nekomutatyvia grupe šifrogramos elementų skirstiniai su 100 skirtingų raktų



36 pav. AES-128 CTR režimu gautų šifrogramos elementų skirstinys su 100 skirtingų raktų

3.3. CBC ir CTR režimų šifravimo laikų palyginimas

Siekiant įvertinti skirtingu režimų pranašumus toliau esančiame grafike pateikiami laikai reikalingi užšifruoti pranešimą pateikti 37 pav. Laikai įvertinti tik šifravimo žingsnio, t. y. raktų generavimas ir matricų paruošimas šifravimui nebuvo įtrauktas į laiko skaičiavimą. Matricos eilės skirtingiems testavimo scenarijams buvo keičiamos nuo 4 iki 10, kiekvienam scenarijui atlikta po 10 generacijų ir paimtas gautųjų rezultatų vidurkis. Šifravimui pasirinktas paveikslas pateiktas 5 pav.



37 pav. Gauti šifravimo laikai lyginant CBC ir CTR režimus taikant skirtingus algoritmus

4 lentelė. Šifravimo laikų priklausomybė nuo matricos eilės ir pasirinkto šifravimo algoritmo CBC režimu

Šifruojamas paveikslas – Cameraman.tiff, suskaidant į 8 bitų elementus								
	m=3	m=4	m=5	m=6	m=7	m=8	m=9	m=10
Šifras paremtas komutatyvia grupe CBC režimu	15,35	19,56s	24,31s	28,37s	33,82s	37,98s	44,09s	46,90s
Šifras paremtas nekomutatyvia grupe CBC režimu	6,86	8,61s	10,94s	12,66s	14,95s	16,75s	19,45s	21,16s
AES-CBC	-	7.92s	-	-	-	-	-	-

5 lentelė. Šifravimo laikų priklausomybė nuo matricos eilės ir pasirinkto šifravimo algoritmo CTR režimu

Šifruojamas paveikslas – Cameraman.tiff, suskaidant į 8 bitų elementus								
	m=3	m=4	m=5	m=6	m=7	m=8	m=9	m=10
Šifras paremtas komutatyvia grupe CTR režimu	15,10s	19,54s	24,44s	28,71s	33,07s	37,06s	43,95s	46,84s
Šifras paremtas nekomutatyvia grupe CTR režimu	7,28s	8,98s	11,27s	13,25s	15,56s	17,26s	20,22s	21,65s
AES-CTR	-	7,84s	-	-	-	-	-	-

Žiūrint į grafiką galima pastebėti, kad didinant matricos eilę didėja ir laikas reikalingas užšifruoti pranešimą. Taip pat pastebime, kad šifravimas paremtas nekomutatyvia grupe laimi prieš šifravimo algoritmą paremta komutatyvia grupe. Tai galima paaiškinti faktu, kad operacijos taikomos algoritme paremtu komutatyvia grupe atliekamos įprastai, t. y. keliant skaičius laipsniais juos dauginant ir redukuojant pirminio skaičiaus moduliu p arba q, priklausomai nuo struktūros kurioje atliekamos operacijos. Tačiau algebrinės operacijos atliekamos algoritme paremtame nekomutatyvia grupe kaip laipsnio kėlimo operacijos ir narių daugyba atliekama pagal instrukcijas, kurios leidžia dirbti su generatorių laipsniai, čia turimos operacijos yra paprastesnės ir reikalauja mažiau sąnaudų jas atlikti. Nepaisant to, galima pastebėti, kad CTR režimo laikai išsiskiria lyginant su CBC šifravimo režimu, čia reikia prisiminti, kad CBC šifravimo režime visi bloko elementai yra sudedami taikant XOR operaciją, o CTR režime, pridedamas tik vienas skaičius, t. y. bloko indeksas padidinamas vienetu. Matome, kad operacijų skaičius yra mažesnius, todėl ir laiko prasme, tai užtrunka mažiau, žinoma galima pastebėti ir išskirtį, kai matricos eilė lygi 6, čia panašu skirtumas atsiranda dėl skirtingų generuotų scenarijų ir mažo skirtumo tarp tiriamų šifrogramų

laikų, kur didesnis nestandartinis nuokrypis nusvėrė rezultatus, šiuo atveju CBC naudai. Praleidus testus papildomus kartus, gauta tokia pati tendencija matoma su kitomis matricos eilėmis.

6 lentelė. Šifravimo laikų priklausomybė nuo matricos eilės ir pasirinkto šifravimo algoritmo CBC režimu

Šifruojamas paveikslas – Cameraman.tiff, suskaidant į 4 bitų elementus								
	m=3	m=4	m=5	m=6	m=7	m=8	m=9	m=10
Šifras paremtas komutatyvia grupe CBC režimu	17,14s	21,03s	27,56s	31,17s	38,39s	41,01s	46,91s	52,47s
Šifras paremtas nekomutatyvia grupe CBC režimu	14,00s	17,73s	22,28s	25,63s	30,03s	33,70s	37,90s	42,62s

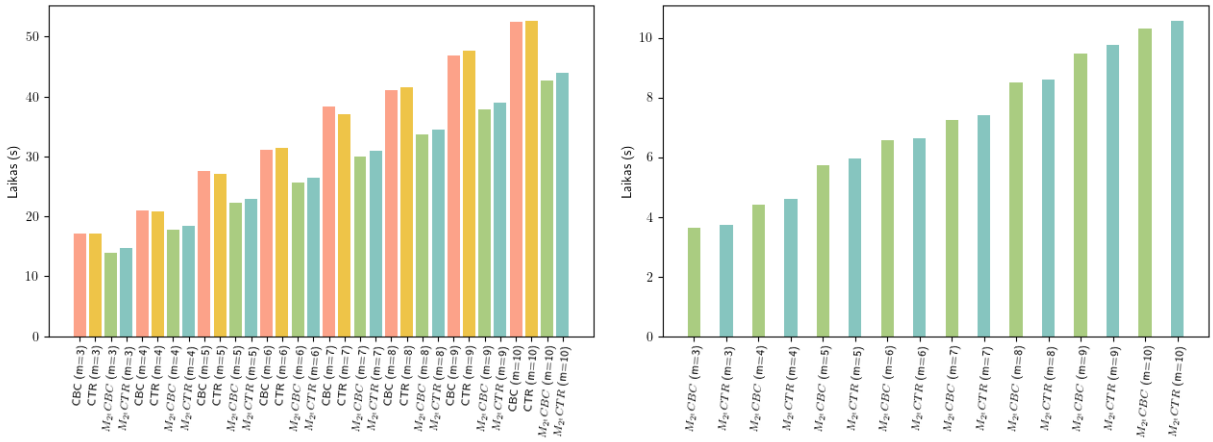
7 lentelė. Šifravimo laikų priklausomybė nuo matricos eilės ir pasirinkto šifravimo algoritmo CTR režimu

Šifruojamas paveikslas – Cameraman.tiff, suskaidant į 4 bitų elementus								
	m=3	m=4	m=5	m=6	m=7	m=8	m=9	m=10
Šifras paremtas komutatyvia grupe CTR režimu	17,15s	20,78s	27,09s	31,48s	37,12s	41,63s	47,61s	52,58s
Šifras paremtas nekomutatyvia grupe CTR režimu	14,74s	18,48s	22,87s	26,47s	30,95s	34,53s	39,07s	43,96s

Skyrelyje 2.4.1. buvo minima kaip galima išskaidyti arba apjungti turimus elementus 38 pav. kairėje pateikiamas grafikas gaunamas šifruojant tą patį paveikslą, tačiau elementai suskaidyti į 4 bitus. Galima pastebėti, kad atlikę šią modifikaciją, algoritmas veikia lėčiau, kadangi turimų blokų skaičius padidėja, nors ir operuojama su mažesniais skaičiais (gauti šifravimo laikai pateikti 6-7 lentelėse). Taip pat galime operuojamus elementus apjungti į 16 bitų elementus (rezultatai pateikti 8 lentelėje), tai galima lengvai padaryti algoritmui paremtu nekomutatyvia grupe, kadangi grupės eilė yra priklausoma nuo 2 laipsnio, taip išnaudojant 16 bitų. Kadangi tyrimas buvo vykdomas naudojant 32 bitų Python programinę įrangą, maksimalus sveikųjų skaičių dydis lygus $2^{31} - 1$. Tai turi įtakos, kada norime nagrinėti tokiu pačiu principų modifikuota šifrą paremtą komutatyvia grupe. Grupės eilė turi būti pasirinkta taip, kad būtų didesnė už šifruojamus elementus, kad iššifruojant nebūtų prarandama prasmė, todėl pirminis skaičius $q > 2^{16}$, o parametras $p > 2^{17}$, nors šie skaičiai mažesni už 32 bitų elementus, tačiau norint sudauginti elementus to nepavyksta įgyvendinti, nes šie viršija 32 bitų ribą prieš atliekant redukcija moduliu. Todėl 38 pav. dešinėje pateikiami laikai taikant tik šifrą paremtą nekomutatyvia grupe.

8 lentelė. Šifravimo laikų priklausomybė nuo matricos eilės ir pasirinkto šifravimo algoritmo CTR režimu

Šifruojamas paveikslas – Cameraman.tiff, suskaidant į 16 bitų elementus								
	m=3	m=4	m=5	m=6	m=7	m=8	m=9	m=10
Šifras paremtas komutatyvia grupe CTR režimu	3,63s	4,39s	5,72s	6,58s	7,25s	8,49s	9,48s	10,32s
Šifras paremtas nekomutatyvia grupe CTR režimu	3,73s	4,61s	5,96s	6,62s	7,40s	8,59s	9,76s	10,56s



38 pav. Gauti šifravimo laikai lyginant CBC ir CTR režimus taikant skirtingus algoritmus turint mažesnes (4 bitų) ir didesnes (16 bitų) matricų įvestis

Tikslūs laikai pateikti 9-10 lentelėse. Galima pastebėti, kad didinant elementų bitų skaičių ir atitinkamai grupės eilę mažėja ir laikas reikalingas užšifruoti pranešimą. Reikia atkreipti dėmesį, kad laikas skaičiuojamas pačio šifravimo, t. y. laikas reikalingas išskaidyti ar apjungti elementus nėra įtraukiamas. Tačiau išskaidyti elementus po 4 bitus vidutiniškai šiam paveikslui užtrunka 0,11 sekundžių, o apjungti į elementus po 16 bitų užtrunka 0,06s sekundžių. Čia pateiki rezultatai tik taikant 4 eilės blokus, kad šiuos būtų galima objektyviau palyginti su AES algoritmo rezultatais.

9 lentelė. Šifravimo laikų priklausomybė nuo šifravimo algoritmo ir elementų įvesties dydžio CBC režimu

Šifruojamas paveikslas – Cameraman.tiff			
	4 bitai	8 bitai	16 bitų
Šifras paremtas komutatyvia grupe CBC režimu	21,03s	19,56s	-
Šifras paremtas nekomutatyvia grupe CBC režimu	17,73s	8,61s	4,39s
AES-CBC	-	7.92s	-

10 lentelė Šifravimo laikų priklausomybė nuo šifravimo algoritmo ir elementų įvesties dydžio CTR režimu

Šifruojamas paveikslas – Cameraman.tiff			
	4 bitai	8 bitai	16 bitų
Šifras paremtas komutatyvia grupe CTR režimu	20,78s	19,54s	-
Šifras paremtas nekomutatyvia grupe CTR režimu	18,48s	8,98s	4,61s
AES-CTR	-	7,84s	-

Galima pastebėti, kad apjungus elementus gauname mažiau blokų ir elemento didumas neturi tokios didelės įtakos, kaip matricos eilės keitimas, todėl šiuo atveju laikai gaunami užšifruoti tą patį pranešimą yra mažesni. Matome, kad turint šifrą paremtą nekomutatyvia grupe didinant matricos elementų įvestis, gaunamas laikas beveik du kartus mažesnis lyginant su laiku turint mažesnė matricos įvestį. Tai galima paaiškinti turimo šifro struktūra, kur turint didesnę grupės eilę laipsnio kėlimo operacija nereikalauja papildomo kiekio daugybų, tačiau daugiausiai yra paveikiamas blokų skaičiaus sumažėjimu, todėl čia pastebimas ryškus laiko pokytis, kuris labiausiai priklauso nuo minėtojo blokų skaičiaus pokyčio. Palyginimui su gautaisiais rezultatais, šifravimo algoritmas AES-

128 pagal apibrėžimą naudoja tik 4x4 dydžio blokus, kuriuose elementai yra 8 bitų ilgio. Šio algoritmo CBC ir CTR šifravimo režimo laikas vidutiniškai yra lygus 7,92s ir 7,84s atitinkamai.

3.4. Rezultatų palyginimas ir rekomendacijos

Remiantis gautaisiais rezultatais galima pastebėti, kad norint užšifruoti didelius pranešimus rekomenduoti suskaidyti kiek įmanoma mažesnius blokus. Pastebėta, kad turint mažesnes matricas, operacijos atliekamos greičiau, nors ir naudojamų blokų skaičius išauga, tačiau atliekamų operacijų greitis kompensuoja blokų skaičiaus padidėjimą. Kadangi pasiūlytuose algoritmuose galima keisti naudojamų grupių eiles, galima elementus apjungti, taip sumažinant blokų skaičių ir padidinant naudojamos grupės eilę. Šis apjungimas remiantis 4 lentele užtikrina greitesnį algoritmo veikimą, jeigu naudojama architektūra leidžia padidinti elementus ir atlikti veiksmus su jais. Turint ribotus darbinės atminties resursus, rekomenduotina būtų naudoti šifrą paremtą nekomutatyvia grupe, kadangi šis efektyviai išnaudojama naudojamus bitus, atlikdamas šifravimo operacijas greičiau negu tai galėtų atlikti šifras paremtas komutatyvia grupe. Tačiau čia reikia atkreipti dėmesį, kad nors ir operacijos yra atliekamos greičiau, tačiau skirtingai negu taikant šifrą paremtą komutatyvia grupe, siekiant užtikrinti pakankamą sumaišymą, reikia papildomai saugoti pasirinkto parametro p m eilės vaizdavimų, kurie yra naudojami užtikrinti elementų sumaišymą. Kitaip sakant greičio išlošis yra gaunamas atminties sąnaudų elementams saugoti sąskaita. Jeigu atmintis elementams saugoti yra ribota, tada rekomenduotina naudoti šifrą paremtą komutatyvia grupe. Čia nereikia saugoti papildomų vaizdavimų matricos elementų vaizdavimui, todėl sutaupoma papildomai vietos. Remiantis [2, 3] rekomenduotina naudoti kiek įmanomą didesnę grupės eilę siekiant užtikrinti geresnius maišos rezultatus, todėl algoritmui virš nekomutatyvios grupės būtų rekomenduotina naudoti 16 bitų matricos įvestis ir parametrą $t = 16$, t. y. naudoti grupę $M_{2^{16}}$. Šifrui virš komutatyvios grupės 32 bitų architektūroms būtų rekomenduoti naudoti $p = 563, q = 281$, 64 bitų architektūroms $p = 131267, q = 65633$.

Šifruojant tekstinius pranešimus, galima taikyti tokius pačius parametrus, kaip ir paveikslams užšifruoti, tačiau taip pat elementus galima suskaidyti ir taikyta mažesnę grupės eilę. Priklausomai nuo turimų resursų galima rinkti arba mažesnę, arba didesnę grupės eilę. Nors mažesnioji grupės eilės pareikalaus mažiau atminties sąnaudų raktams ir vaizdavimų matricoms saugoti, tačiau šifravimo laiko bus galima tikėtis ilgesnio negu tai galėtume gauti taikant aukštesnės eilės grupę (žiūrėti 9-10 lenteles).

Šiame darbe tekstogramoms užšifruoti buvo naudotas 4 eilės blokai, tačiau laiko atžvilgiu efektyviau šifruoja 3 eilės blokai. Čia reikia atkreipti dėmesį, kad turint CTR šifravimo režimą dalis įvesties bloko yra kamšalas ir kita dalis - skaitiklis (bloko indeksas). Turint pakankamai mažą bloką, šiame sunku tampa sutalpinti ir kamšalą ir skaitiklį, kadangi turint didelį blokų skaičių, gali likti mažai vietos kamšalui, kuris yra reikalingas užtikrinti, kad siunčiamas tas pats pranešimas kelis kartus duotų skirtingus rezultatus. Taigi turint mažą pranešimą 3 eilės blokai yra priimtini, tačiau turint didesnį pranešimą rekomenduotina naudoti bent 4 eilės blokus.

Išvados

1. CBC šifravimo režimas užtikrina pakankamą elementų sumaišymą taikant darbe apibrėžtus šifravimo algoritmus.
2. CTR šifravimo režimui siekiant užtikriną pakankamą elementų sumaišymą reikia įvesti papildomas modifikacijas. Šifrui paremtu komutatyvia grupe naudojamas elementų sumavimas grupės eilės moduliu, o šifrui paremtu nekomutatyvia grupe taikomi papildomi vaizdavimai.
3. Šifras paremtas nekomutatyvia grupe efektyviau išnaudoja bitus, todėl galima lengviau padidinti grupės elementus, siekiant greičiau užšifruoti tekstogramas.
4. Šifras paremtas nekomutatyvia grupe pranešimus užšifruoja greičiau nei taikant šifrą paremtą komutatyvia grupe. Nors operacijos yra labai panašios, tačiau dėl savo grupės operacijų struktūros, resursams jautrios operacijos tokios kaip laipsnio kėlimas yra apibrėžiamos kitaip, kas suteikia laiko išlošį.
5. Norint užšifruoti ilgą tekstogramą, rekomenduotina naudoti bent 4 eilės blokus, su kiek įmanoma didesne grupės eile, t. y. jeigu turima 32 bitų architektūra, tai taikant šifrą paremtą nekomutatyvia grupe imame parametrą $t = 16$ ir šifruojama 16 bitų elementus. Turint mažiau atminties resursų ir trumpą tekstogramą, šią galima išskaidyti į 4 bitų elementus ir naudoti mažesnę grupės eilę, tačiau reikėtų įvertinti, kad tai prailgins skaičiavimo laiką, kadangi blokų skaičius išaugs.

Literatūros sąrašas

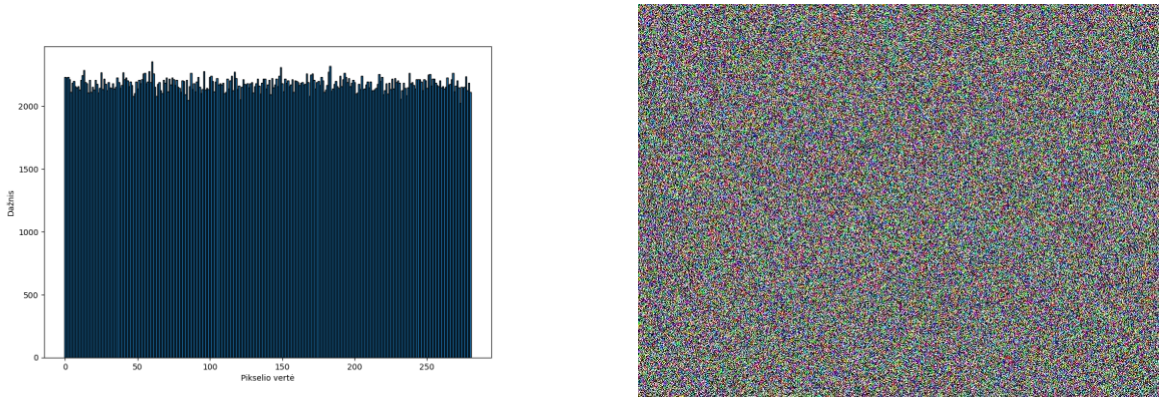
1. MIHALKOVICH, Aleksėjus; Eligijus SAKALAUSKAS ir Kestutis LUKŠYS. Key exchange protocol defined over a non-commuting group based on an NP-complete decisional problem. *Symmetry*, 2020, 12.9: 1389. DOI: 10.3390/sym12091389
2. LEVINSKAS, Matas ir Aleksėjus MIHALKOVICH. Avalanche effect and bit independence criterion of perfectly secure Shannon cipher based on matrix power. *Mathematical models in engineering*, 2021, 7.3: 50-53. DOI: 10.21595/mme.2021.22234
3. MIHALKOVICH, Aleksėjus; Matas LEVINSKAS ir Pijus MAKAUSKAS. MPF based symmetric cipher performance comparison to AES and TDES. *Mathematical models in engineering*, 2022, 8.2: 15-25. DOI: 10.21595/mme.2022.22517
4. MIHALKOVICH, Aleksejus, Matas LEVINSKAS, Lina DINDIENĖ ir Eligijus SAKALAUSKAS. CBC Mode of MPF Based Shannon Cipher Defined Over a Non-Commuting Platform Group. *Informatica*, 2022, 33.4: 833-856. DOI: 10.15388/22-INFOR499
5. DINDIENĖ, Lina, Aleksėjus MIHALKOVICH, Kęstutis LUKŠYS ir Eligijus SAKALAUSKAS. Matrix Power Function Based Block Cipher Operating in CBC Mode. *Mathematics*, 2022, 10.12: 2123. DOI: 10.3390/math10122123
6. MIHALKOVICH, Aleksejus; Matas LEVINSKAS ir Eligijus SAKALAUSKAS. Counter Mode of the Shannon Block Cipher Based on MPF Defined over a Non-Commuting Group. *Mathematics*, 2022, 10.18: 3363. DOI: 10.3390/math10183363
7. SAKALAUSKAS, Eligijus, Lina DINDIENĖ, Aušrys KILČIAUSKAS ir Kęstutis LUKŠYS. Perfectly secure Shannon cipher construction based on the matrix power function. *Symmetry*, 2020, 12.5: 860. DOI: 10.3390/sym12050860
8. DIFFIE Whitfield ir Martin HELLMAN, "New directions in cryptography," in *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644-654, November 1976. DOI: 10.1109/TIT.1976.1055638.
9. BERNSTEIN, Daniel J. Salsa20 specification. eSTREAM Project algorithm description, http://www. ecrypt. eu. org/stream/salsa20pf. html, 2005.
10. BERNSTEIN, Daniel J. ChaCha, a variant of Salsa20. In: *Workshop record of SASC*. 2008. p. 3-5.
11. MILLER, Frank. Telegraphic code to insure privacy and secrecy in the transmission of telegrams. CM Cornwell, 1882.
12. BONEH, Dan ir Victor SHOUP, 2020. A Graduate Course in Applied Cryptography. [žiūrėta 2021-04-11]. Prieiga per internetą: https://toc.cryptobook.us/.
13. Data Encryption Standard (DES), *Federal Information Processing Standards Publication 197, United States National Institute of Standards and Technology (NIST)*: Gaithersburg, MD, USA, 1977.
14. BARKER, Elaine ir Nicky MOUHA. Recommendation for the Triple Data Encryption Algorithm (TDEA) block cipher, *National Institute of Standards and Technology, Gaithersburg, MD, Special Publication (NIST SP)*, Nov. 2017. DOI: 10.6028/NIST.SP.800-67r2
15. DWORKIN, J. Morris, Elaine B. BARKER, James R. NECHVATAL, James FOTI, Lawrence E. BASSHAM, E. ROBACK, James F. DRAY Jr., 2001. Advanced Encryption Standard (AES). DOI: 10.6028/NIST.FIPS.197

16. FEISTEL, Horst, Cryptography and Computer *Privacy, Scientific American*, Vol. 228, No. 5, pp. 15–23, 1973.
17. GIRY Damien, Cryptographic key length recommendation, 2020 [žiūrėta 2023-05-07]. Prieiga per internetą: https://www.keylength.com/en/4/
18. Federal Office for Information Security. BSI - Technical Guideline. Cryptographic Mechanisms: Recommendations and Key Lengths. [žiūrėta 2023-05-07]. Prieiga per internetą: https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TG02102/BSI-TR-02102-1.pdf?__blob=publicationFile
19. BIRYUKOV, Alex ir Dmitry KHOVRATOVICH. Related-key cryptanalysis of the full AES-192 and AES-256. *Advances in Cryptology–ASIACRYPT 2009: 15th International Conference on the Theory and Application of Cryptology and Information Security*, Tokyo, Japan, December 6-10, 2009. Proceedings 15. Springer Berlin Heidelberg, 2009. p. 1-18. DOI: 10.1007/978-3-642-10366-7_1
20. Federal Office for Information Security. Quantum-safe cryptography – fundamentals, current developments and recommendations. 2022. [žiūrėta 2023-05-07]. Prieiga per internetą: https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Brochure/quantum-safe-cryptography.pdf?__blob=publicationFile&v=4
21. KANG, HyungChul, et al. New efficient padding methods secure against padding oracle attacks. *Information Security and Cryptology-ICISC 2015: 18th International Conference*, Seoul, South Korea, November 25-27, 2015, Revised Selected Papers 18. Springer International Publishing, 2016. p. 329-342. DOI: 10.1007/978-3-319-30840-1_21
22. B. Kaliski. PKCS #5: Password-Based Cryptography Specification, 2000. [žiūrėta 2023-05-07]. Prieiga per internetą: https://www.ietf.org/rfc/rfc2898.txt
23. B. Kaliski. PKCS #7: Cryptographic Message Syntax, 1998. [žiūrėta 2023-05-07]. Prieiga per internetą: https://www.ietf.org/rfc/rfc2315.txt
24. NIST, FIPS. 81: DES Modes of Operation. 1980. DOI: 10.6028/NBS.FIPS.81
25. DIFFIE, Whitfield; ir Martin E HELLMAN. Special Feature Exhaustive Cryptanalysis of the NBS Data Encryption Standard" Computer (Long Beach, Calif.) 10, no. 6 (1977): 74-84. DOI: 10.1109/C-M.1977.217750
26. DWORKIN, Morris. Recommendation for block cipher modes of operation. methods and techniques. *National Inst of Standards and Technology Gaithersburg MD Computer security* Div, 2001. DOI: 10.6028/NIST.SP.800-38A
27. SAKALAUSKAS, Eligijus ir Kęstutis LUKŠYS, 2007. Matrix power S-box construction, Cryptology ePrint Archive: Report, no.214. Prieiga per internetą: http://eprint.iacr.org/2007/214.
28. SAKALAUSKAS Eligijus ir Kestutis LUKSYS. Matrix power function and its application to block cipher s-box construction. *Int. J. Inn. Comp., Inf. Contr*, 2012, 8.4: 2655-2664. ISSN: 1349-4198
29. GRUNDMAN, G. Helen ir Tara L. SMITH, 1996. Automatic Realizability of Galois Groups of Order 16. *Proceedings of the American Mathematical Society*, 124(9):2631-2640. DOI: 10.1090/S0002-9939-96-03345-X
30. MIHALKOVICH, Aleksėjus. On the associativity property of MPF over M16. Lietuvos matematikos rinkinys: Lietuvos matematikų draugijos darbai. Serija A, 2018, 59: 7-12. DOI: DOI: 10.15388/LMR.A.2018.02
31. Teksto generavimo įrankis. Prieiga per internetą: https://app.inferkit.com/demo

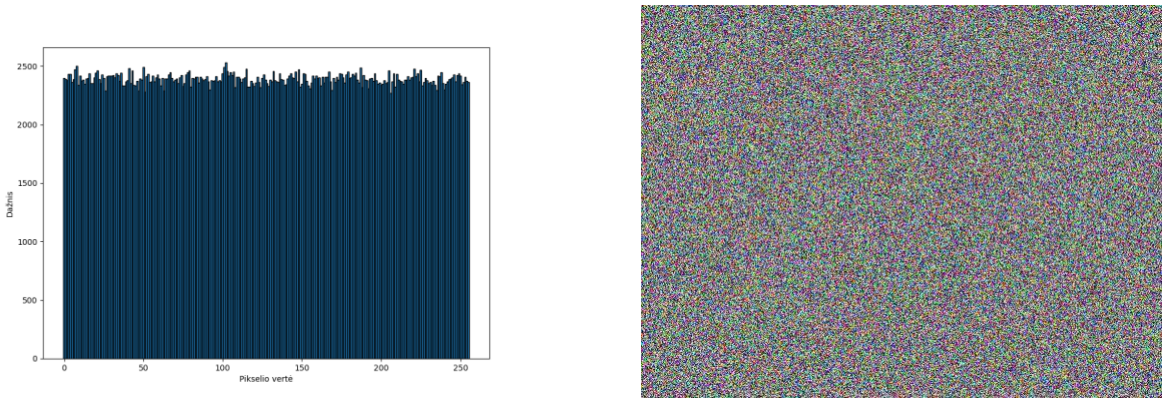
Priedai

1 Priedas. Papildomi tyrimai taikant skirtingus vaizdus

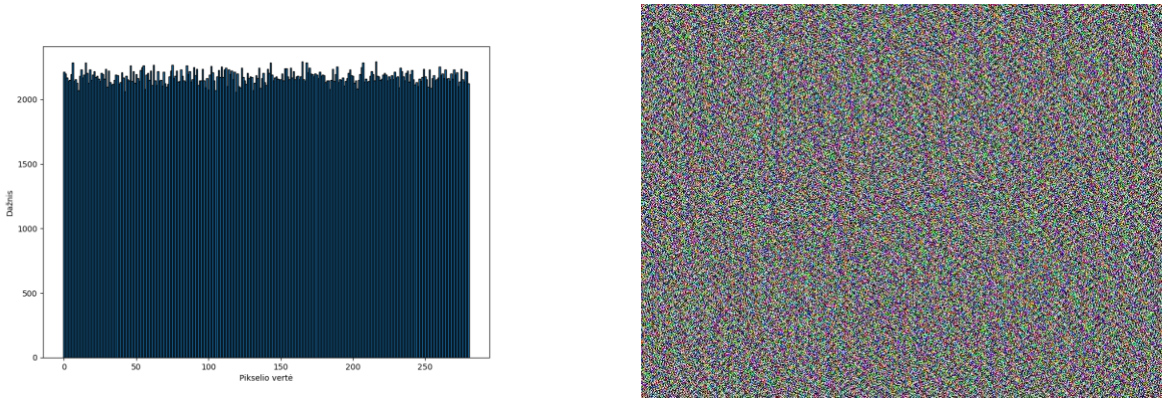
39 pav. Pradinis paveikslas



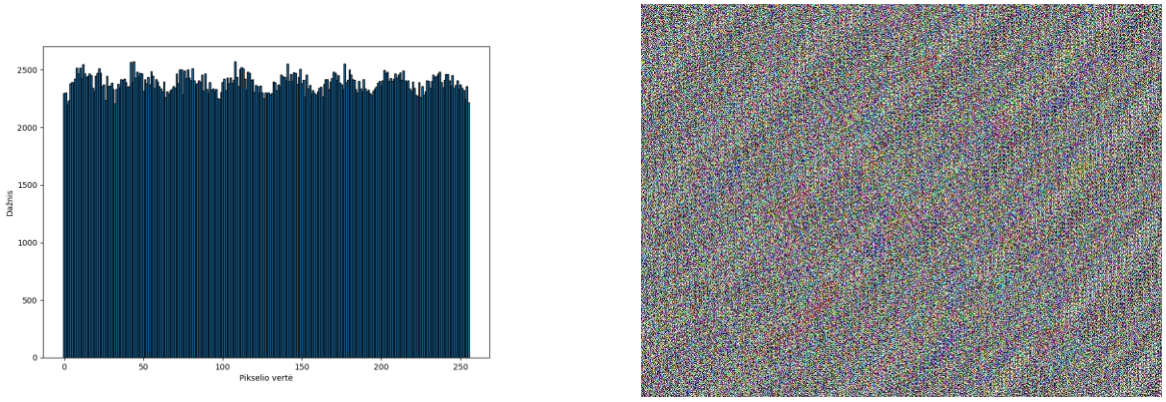
40 pav. Rezultatai gauti taikant šifravimo režimą paremtą komutatyvia grupe CBC režimu, naudojant 4 eilės blokus ir atliekant elementų sumavimą moduliu q



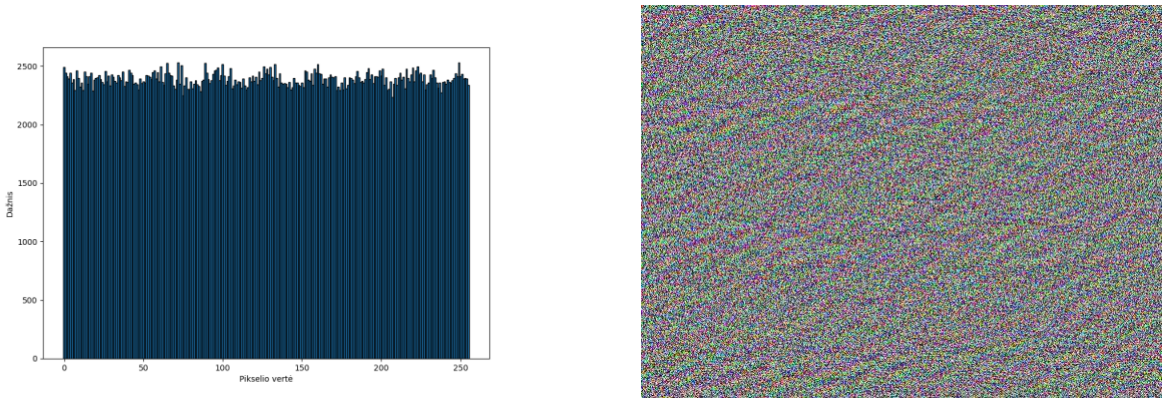
41 pav. Rezultatai gauti taikant šifravimo režimą paremtą nekomutatyvia grupe CBC režimu, naudojant 4 eilės blokus ir atliekant elementų sumavimą moduliu 2^t



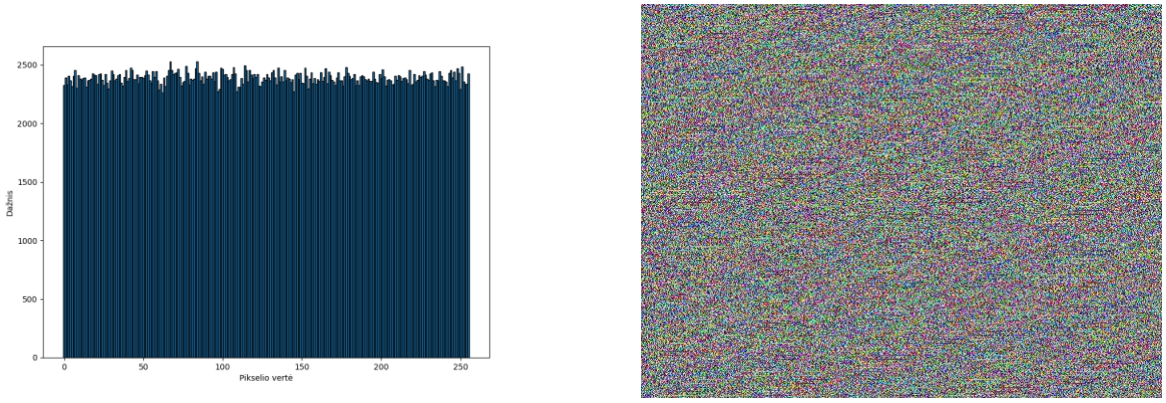
42 pav. Rezultatai gauti taikant šifravimo režimą paremtą komutatyvia grupe CTR režimu, naudojant 4 eilės blokus ir atliekant elementų sumavimą moduliu q



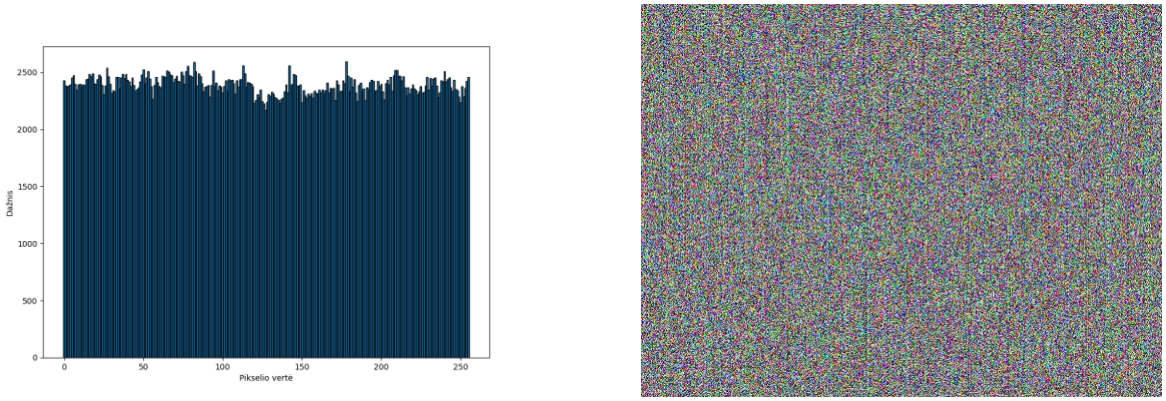
43 pav. Rezultatai gauti taikant šifravimo režimą paremtą nekomutatyvia grupe CTR režimu, naudojant 4 eilės blokus ir atliekant elementų sumavimą moduliu 2^t



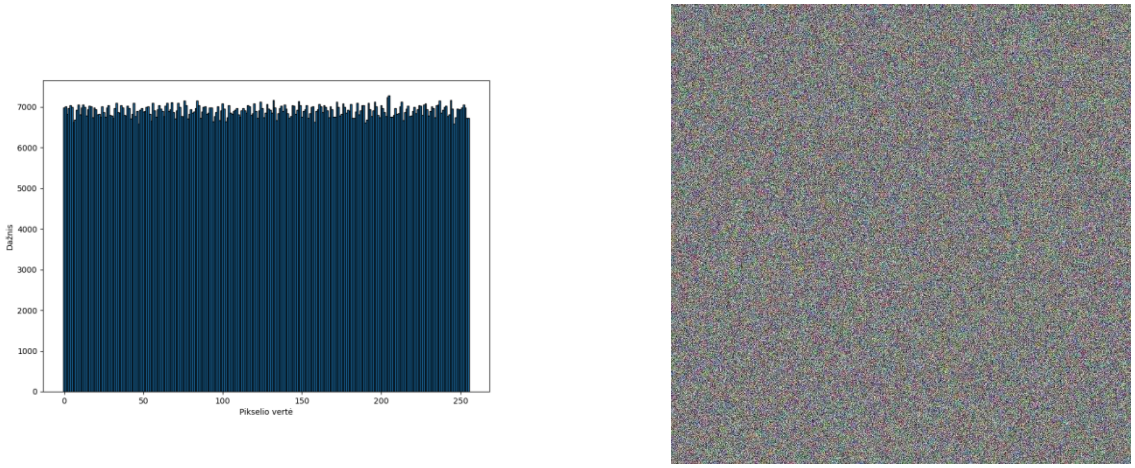
44 pav. Rezultatai gauti taikant šifravimo režimą paremtą nekomutatyvia grupe CTR režimu, naudojant 4 eilės blokus ir atliekant elementų sumavimą moduliu 2^t, bei pašalinus matricos Y apribojimą



45 pav. Rezultatai gauti taikant šifravimo režimą paremtą nekomutatyvia grupe CTR režimu, naudojant 4 eilės blokus, atliekant elementų sumavimą moduliu 2^t su pašalintu matricos Y apribojimu, bei elementų sumaišymu

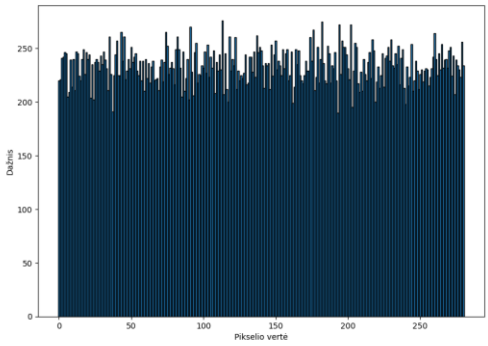


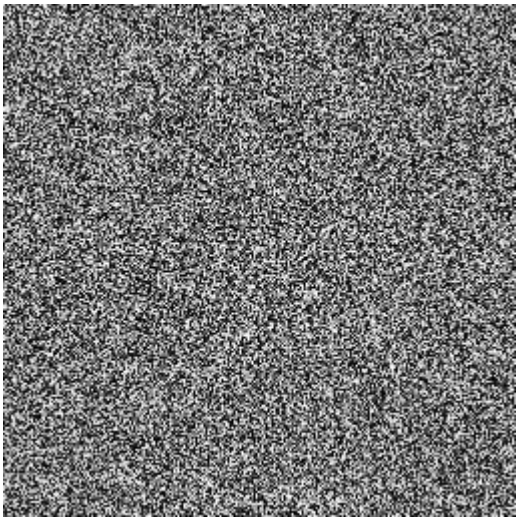
46 pav. Rezultatai gauti taikant šifravimo režimą paremtą nekomutatyvia grupe CTR režimu, naudojant 4 eilės blokus, atliekant elementų sumavimą moduliu 2^t su pašalintu matricos Y apribojimu, bei elementų sumaišymu kartu su įprastu skaitikliu



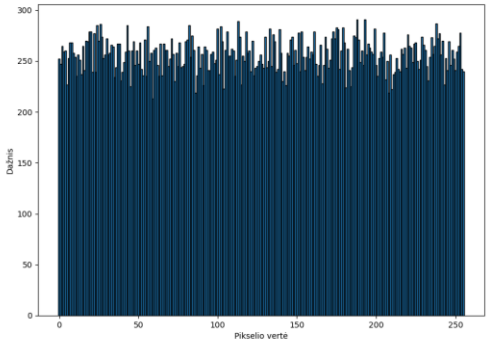
47 pav. Rezultatai gauti taikant šifravimo režimą paremtą nekomutatyvia grupe CTR režimu, naudojant 4 eilės blokus, atliekant elementų sumavimą moduliu 2^t su pašalintu matricos Y apribojimu, bei elementų sumaišymu kartu su pridedamo skaitiklio vaizdavimu

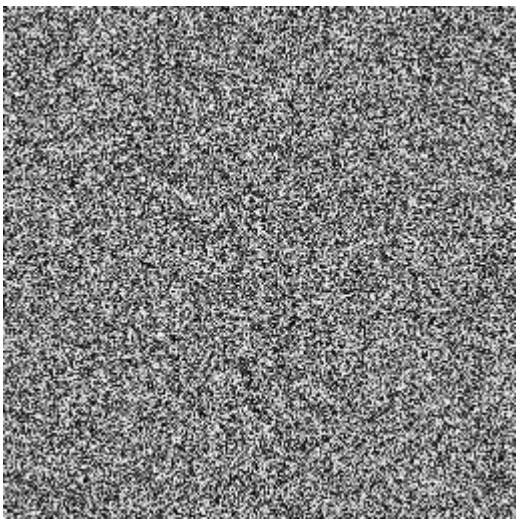
48 pav. Pradinis paveikslas



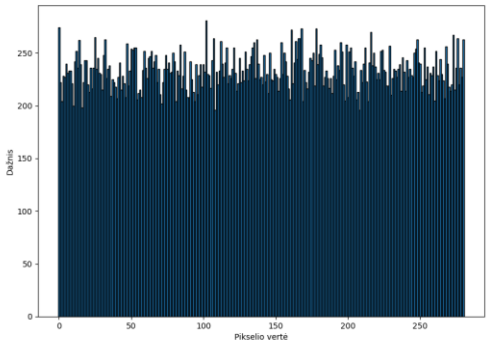


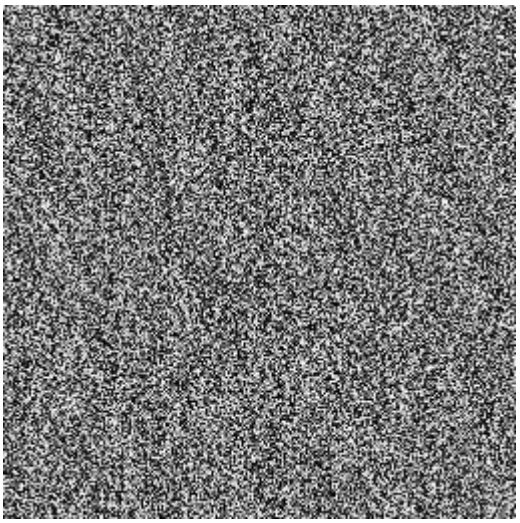
49 pav. Rezultatai gauti taikant šifravimo režimą paremtą komutatyvia grupe CBC režimu, naudojant 4 eilės blokus ir atliekant elementų sumavimą moduliu q



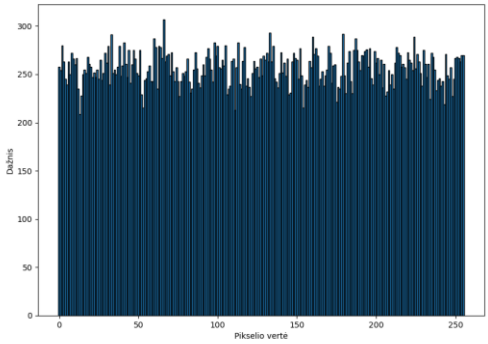


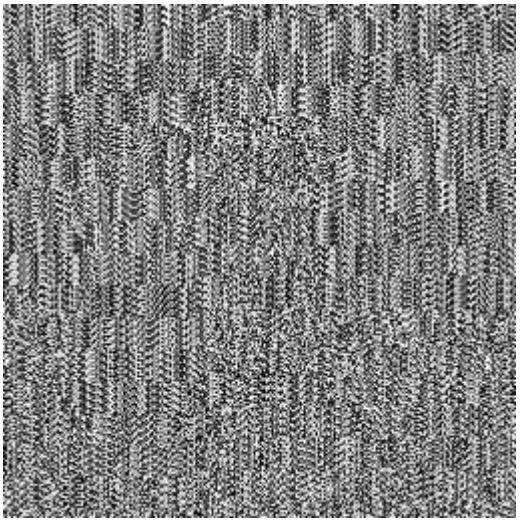
50 pav. Rezultatai gauti taikant šifravimo režimą paremtą nekomutatyvia grupe CBC režimu, naudojant 4 eilės blokus ir atliekant elementų sumavimą moduliu 2^t



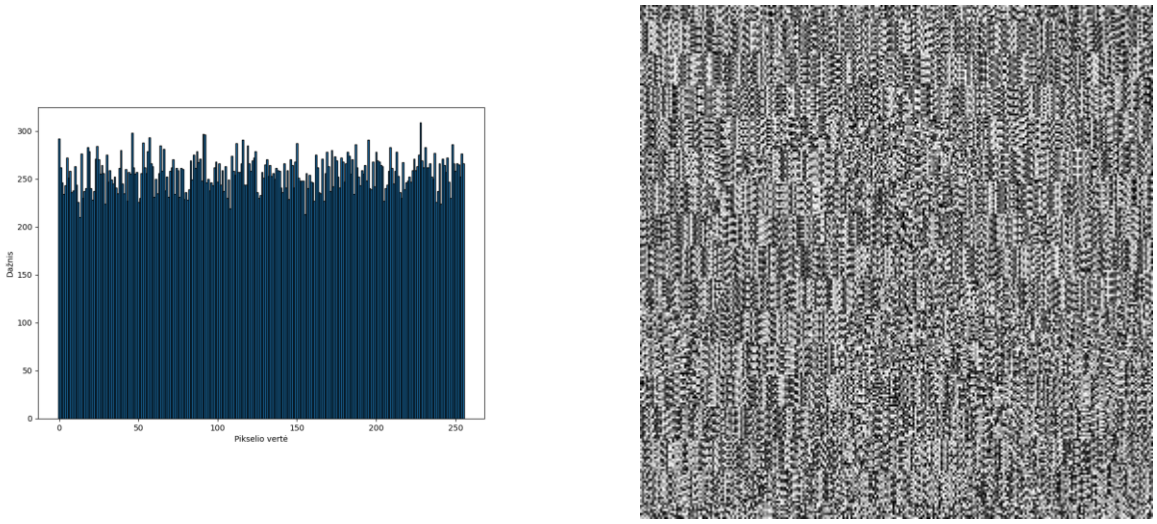


51 pav. Rezultatai gauti taikant šifravimo režimą paremtą komutatyvia grupe CTR režimu, naudojant 4 eilės blokus ir atliekant elementų sumavimą moduliu q

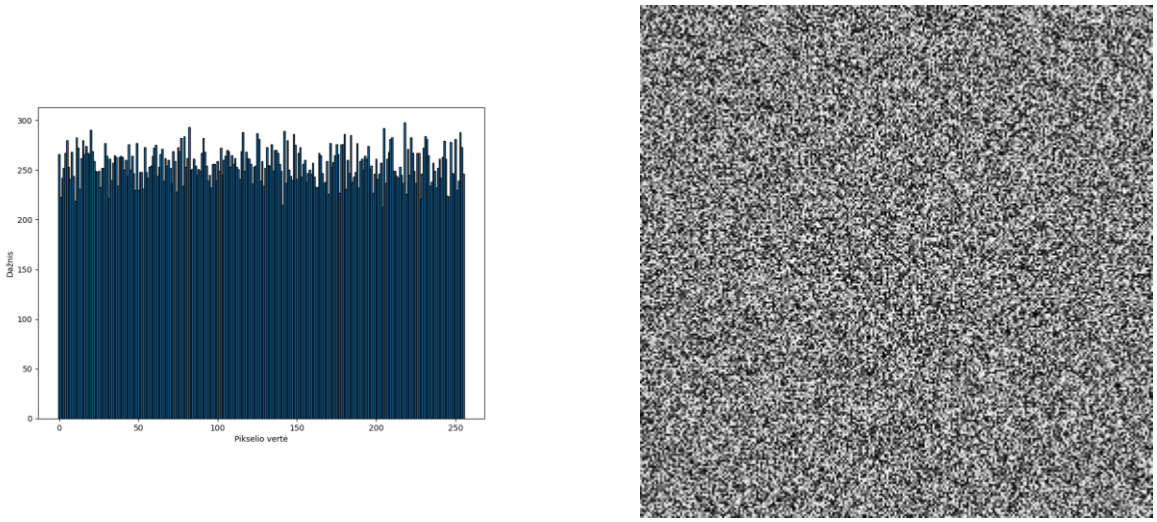




52 pav. Rezultatai gauti taikant šifravimo režimą paremtą nekomutatyvia grupe CTR režimu, naudojant 4 eilės blokus ir atliekant elementų sumavimą moduliu 2^t



53 pav. Rezultatai gauti taikant šifravimo režimą paremtą nekomutatyvia grupe CTR režimu, naudojant 4 eilės blokus ir atliekant elementų sumavimą moduliu 2^t, bei pašalinus matricos Y apribojimą



54 pav. Rezultatai gauti taikant šifravimo režimą paremtą nekomutatyvia grupe CTR režimu, naudojant 4 eilės blokus, atliekant elementų sumavimą moduliu 2^t su pašalintu matricos Y apribojimu, bei elementų sumaišymu kartu su pridedamo skaitiklio vaizdavimu

2 Priedas. Šifruojamos tekstogramos tekstas

III) A Deep Analysis

How plausible is it that something could communicate hidden messages? This depends on how much information one knows and at what depth one reaches in analyzing or decoding that information. Given what is known, we are trying to communicate encoded details of either encrypted bank notes, encrypted communications, ciphers, or coded communication. Most resources have estimated methods of decoding or locating hidden messages, or signatures indicating hidden messages. Therefore, beyond constructing reasonably vulnerable protocols and infrastructure, it is unlikely that a standard system would only protect on some protocols but ignore all of them. In most sectors, especially security, not all targets are predictable. Hence, as of December 2013, unknown intelligence has described 128 valid targets. If you take the actions that protect at full load, someone digging in a known breach risks everything, sometimes via lots of noise or small risks but also sometimes by amplifying small risks. Attacks operate at something very different from cipher machines with handwritten settings, easily embedded on weak devices, protecting valid transfers through reagent spoofing in modes unrelated to national defense, capable of duplicating data back, completely controlled (off or on) by peer agents but hidden.

Despite that, encryption protocols allow further decomposition and encryption is employed on a broad range of targets. Some official standards comply completely to existing requirements and govern different sources of good knowledge. Many key communication methods are nothing more than round tripping protocols, often if closely controlled and provided under known regulations and sometimes provided by organizations employing cryptography, cryptographic or otherwise not, and others implemented privately to evade surveillance, associated with encrypted information or communication systems that permit hidden security systems within institutions like educational and finance ones or high efficiency transportation systems.

Having encapsulated that specification, when defenders must pick up each service or process on arbitrary higher levels in protocol operations or complexity it exposes sensitive, ambiguous or yet undefined material. If one must set some of their security controls over things outside what they approve or desire then an attacker may perceive application output on multiples levels within the transfer to narrow this configuration at most, where best attack systems, sometimes referred to as intermediate configurations usually involve minimal device overhead while providing effective protection at much lesser investment of effort. In this study from 2014 we explored this restriction, this fault given security for software. We found some good work from European and American experts. Our presentation recommended unencoded control at lower scales because greater amounts of modern built or technical tasks mean diminished security with

3 Priedas. Tyrimui naudotas kodas

```
#----------EsakCipher.py-----------------
import numpy as np
from Basic_functions.main_func import *
from Basic_functions.Gauss_Jordan import Gauss_Jordan

def GenG(p):
    """
    Generates index and number list of the mapping function
    """
    q = (p-1)//2
    index = 3
    while (mod_exp(index,q,p) != 1 or np.mod(mod_exp(index,1,p) * mod_exp(index,2,p),q) == mod_exp(index,2,p)):
        index = index + 1
    numbers = []
    for i in range(1,q+1):
        numbers.append(mod_exp(index,i,p))
    return numbers


def Gmap(index, mat):
    """
    Maps matrix elements using function f
    """
    try:
        m = len(mat)
    except:
        m = 1
    M = np.zeros(m)
    fm = np.vectorize(lambda x: index[x])
    M=fm(mat)
    return M.astype('int64')

def imap(ind, num):
    """
    Performs an inverse mapping F^{-1}
    """
    try:
        m = len(num)
    except:
        m = 1
        return ind.index(num)
    M = np.zeros([m,m])
    for i in range(m):
```

```
        for j in range(m):
            found = ind.index(num[i,j])
            M[i,j] = found
    return M.astype('int64')

def ESakCipher(G,X,M,Y,p,q,Fmap):
    """
    G = GenG(p)
    X \in
    M \in
    Y \in Z_q \ {0}
    """
    #q = (p-1)//2
    #Fmap = Gmap(G,X)
    C1 = np.mod(X+M,q).astype('int64')
    F = Gmap(G,C1)
    YF = matrix_exp_left(F,Y,p)
    YFY = matrix_exp_right(YF,Y,p)
    C2 = hadamard_prod(Fmap, YFY,p)
    return np.mod(imap(G, C2)+X,q).astype('int64')

def ESakDecryption(G, YY, X, C,p):
    """
    Decryption algorythm
    """
    q = (p-1)//2
    FF = hadamard_inv(Gmap(G,X),p)
    prod = hadamard_prod(FF,C,p)
    YYprod = matrix_exp_right(prod,YY,p)
    YYprodYY = matrix_exp_left(YYprod,YY,p)
    M_new = np.mod(imap(G,YYprodYY,p)-X,q)
    return M_new

def Gen_parameters(m, p):
    """
    Parameters generation
    """
    q = (p-1)/2
    G = GenG(p)
    X = np.random.randint(0,q,size=(m,m))
    Y = np.random.randint(1,q,size=(m,m))
    Z = np.random.randint(0,q,size=(m,m))
    YY = Gauss_Jordan(Y,q)
    while(np.mod(np.linalg.det(Y),q)==0 or (np.mod(np.matmul(Y,YY),q) == np.eye(m)).sum() !=
m*m):
        Y = np.random.randint(1,q,size=(m,m))
```

```
        YY = Gauss_Jordan(Y,q)
    return q, G, X, Y, Z

---------main_func.py-----------
import numpy as np
import math

def mod_exp(g,n,p):
    """
    Modular exponenet function
    """
    r = 1
    while(n > 0):
        if(np.mod(n,2) != 0):
            r = np.mod(np.multiply(r,g),p)
        n = math.floor(n/2)
        g = np.mod(np.multiply(g,g),p)
    return r

def matrix_exp_left(W,X,p):
    """
    Function allows to raise matrix W by matrix power X from the left side modulo p
    """
    n1, _ = W.shape
    _, m2 = X.shape
    M = np.zeros([n1,m2])
    for i in range(m2):
        for j in range(n1):
            M[i,j] = 1
            for k in range(m2):
                tmp = mod_exp(W[k,j],X[i,k],p)
                M[i,j] = M[i,j]*tmp % p
    return M.astype('int64')

def matrix_exp_right(W,X,p):
    """
    Function allows to raise matrix W by matrix power X from the right side modulo p
    """
    _, n2 = W.shape
    m1, _ = X.shape
    M = np.zeros([n2,m1])
    for i in range(n2):
        for j in range(m1):
            M[i,j] = 1
            for k in range(n2):
                tmp = mod_exp(W[i,k],X[k,j],p)
```

```
                M[i,j] = M[i,j]*tmp % p
    return M.astype('int64')

def hadamard_prod(A,B,p):
    """
    Hadamar product
    """
    return np.mod(np.multiply(A,B),p)

def mulinv(number, modulo):
    """
    Multiplicatively inverse element
    """
    if(math.gcd(number, modulo) != 1):
        #"Inverse element does not exist"
        return -1
    else:
        m, n = np.int64(number),np.int64(modulo)
        s,t,u,v = np.int64(1), np.int64(0), np.int64(0), np.int64(1)
        while(n > 0) :
            q=np.int64(math.floor((np.double(m))/np.double(n)))
            r=np.int64(m-q*n)
            m, n =n, r
            u, s =s-q*u, u
            v, t =t-q*v, v
            out = s
            if(out < 0):
                out = modulo+out
    return out

def hadamard_inv(M,p):
    """
    Inverse with respect of Hadamar operator
    """
    H = None
    fm = np.vectorize(lambda x: mulinv(x,p))
    H=fm(M)
    return H

def Shifting_bits(row,k):
    """
    Bit shifting by k positions
    """
    part = row[2:k+2]
    return row[:2]+row[k+2:] + part
```

```
#----------modes.py---------------
import numpy as np
from ESakCipher import *
from M16 import *


def E_sak_CBC(IV,M, G, X, Y,Z, rounds, p,q):
    """
    CBC mode used with algorithm based on commutative group
    """
    m = X.shape[0]
    Fmap = Gmap(G,Z)
    CF = np.zeros([m,m,rounds+1], dtype='int64')
    CF[:,:,0] = IV
    for i in range(rounds):
        """if(i % 100 == 0):
            print(i/rounds*100)"""
        M1                                                                                                  =
np.mod(CF[:,:,i]+M[:,:,i],q)#np.mod(CF[:,:,i]+M[:,:,i],q)#np.mod(np.bitwise_xor(CF[:,:,i],M[:,:,i]),
q)
        C = ESakCipher(G,X,M1,Y,p,q, Fmap)
        CF[:,:,i+1] = C
    return CF

def E_sak_CTR(M, G, X, Y, Z, rounds, p,q):
    """
    CTR mode used with algorithm based on commutative group
    """
    m = X.shape[0]
    Fmap = Gmap(G,Z)
    Cnt = np.random.randint(1,q,size=(m,m)).astype('int32')
    """Cnt = np.matrix([
   [487,   455,   365,   283],
   [401,   181,   263,   341],
   [487,   219,   243,     8],
    [0,   320,     0,     0]
    ])"""
    Cnt[m-1,m-1] = 0
    Cnt[m-1,m-2] = 0
    Cnt[m-1,m-3] = 0
    Cnt[m-1,m-4] = 0
    CF = np.zeros([m,m,rounds])
    for i in range(rounds):
        """if(i % 100 == 0):
            print(i/rounds*100)"""
        M1 = Cnt.copy()
```

```
        Cnt[m-1,m-1] += 1
        pind = m-1
        while(Cnt[m-1,pind] >= q):
            Cnt[m-1,pind] = 0
            Cnt[m-1,pind-1] += 1
            pind -= 1
        C = ESakCipher(G,X,M1,Y,p,q, Fmap)
        """print("C", C)
        print("M", M[:,:,i]+255)"""
        CF[:,:,i] = np.mod(C+M[:,:,i],q)#np.mod(np.bitwise_xor(C,M[:,:,i]),q)
        #print("CF", CF[:,:,i])
    return CF.astype('int32')

def merg2mat(M1, M2):
    """
    Merging two matrices into one, making 16-bit element
    """
    m = len(M1)
    NM = np.zeros(M1.shape)
    for i in range(m):
        for j in range(m):
            NM[i,j] = int('{:02X}'.format(int(M1[i,j]))+ '{:02X}'.format(int(M2[i,j])), base=16)
    return NM

def spl2mat(M):
    """
    Split matrix into 2, 4-bit elements
    """
    M1 = np.zeros(M.shape)
    M2 = np.zeros(M.shape)
    for i in range(len(M)):
        for j in range(len(M)):
            sk = '{:02X}'.format(int(M[i,j]))
            M1[i,j] = int(sk[0],base = 16)
            M2[i,j] = int(sk[1], base = 16)
    return M1, M2

def Form_pic_blocks(m, X1, rgb, nb=8):
    """
    Making block from Picture
    """
    ss = X1.shape
    #print(ss)
    if(ss[0]%m != 0):
        s1 = (ss[0]//m+1)*m
    else:
```

```
    s1 = ss[0]//m*m
if(ss[1]%m != 0):
    s2 = (ss[1]//m+1)*m
else:
    s2 = ss[1]//m*m
#print(s1,s2)
if(rgb == 3):
    ZZ = np.zeros([s1, s2, rgb])
    ZZ[:ss[0],:ss[1],:] = X1
else:
    ZZ = np.zeros([s1, s2])
    ZZ[:ss[0],:ss[1]] = X1
finish = (s1//m)*(s2//m)

if(finish*rgb%2 != 0 and nb == 16):
    M1 = np.zeros([m,m,finish*rgb+1])
else:
    M1 = np.zeros([m,m,finish*rgb])
indx = 0
for k in range(rgb):
    for i in range(s1//m):
        for j in range(s2//m):
            if(rgb ==3):
                M1[:,:,indx] = ZZ[i*m:(i+1)*m, j*m:(j+1)*m,k]
            else:
                M1[:,:,indx] = ZZ[i*m:(i+1)*m, j*m:(j+1)*m]
            indx += 1
if(nb==16):
    if(indx%2 != 0):
        if(rgb ==3):
            M1[:,:,indx] = np.zeros([m,m,1])
        else:
            M1[:,:,indx] = np.zeros([m,m])
        indx += 1
    MM = np.zeros([m,m,(indx)//2])
    indx2 = 0
    for i in range(0,indx,2):
        MM[:,:,indx2] = merg2mat(M1[:,:,i],M1[:,:,i+1])
        indx2 += 1
    M1 = MM
elif(nb==4):
    MM = np.zeros([m,m,(indx+1)*2])
    indx2 = 0
    for i in range(0,indx):
        MM[:,:,indx2],MM[:,:,indx2+1] = spl2mat(M1[:,:,i])
        indx2 += 2
```

```
        M1 = MM

    return s1, s2, M1.astype('int32')

def Form_text_blocks(m, text_file, nb = 8):
    """
    Making blocks from text
    """
    unicode_file = open(text_file,encoding='utf-8')
    txt = unicode_file.read()
    strl = list()
    for c in txt:
        strl.append('{:02X}'.format(ord(c)))
    NM = []
    #Splits 8bit elements into 2 4bit elements
    for i in range(len(strl)):
        sk = strl[i]
        #print(sk)
        if(nb == 4):
            for ii in range(len(sk)):
                NM.append(int(sk[ii],base = 16))
        elif(nb == 8):
            NM.append(int(sk,base = 16))
    #Hex 2 dec
    """
    for i in range(len(strl)):
        sk = strl[i]
        NM.append(int(sk,base = 16))"""


    diff = m**2 - (len(NM)-math.floor(len(NM)/(m**2))*(m**2))
    for i in range(diff):
        NM.append(0)
    k=-1
    M1 = np.zeros([m,m,len(NM)//(m**2)])
    for i in range(0,len(NM),m**2):
        k += 1
        M1[:,:,k] = np.reshape(NM[i:i+m**2],[m,m])
    return M1.astype('int32')


def M16_CTR(M, Ma, X, Y2, rounds, t, nbits, delta):
    m = X.shape[0]
    Cnt = np.random.randint(0,np.power(2,t),size=(m,m,1)).astype('int32')
    pp = 5
    Vec = [MakeVector(m) for _ in range(pp)]
```

```
    Cnt2 = np.zeros([m,m,1], dtype='int32')
    Cnt2[m-1,m-1,0] = 1
    tt = np.power(2,t)
    CF = np.zeros([m,m,rounds])
    for i in range(rounds):
        if(i % 100 == 0):
            print(i/rounds*100)
        Cnt2 = BuildPlaintext(Vec,m,Cnt2, pp)
        M1 = FormM(BuildPlaintext(Vec,m,np.mod(Cnt+Cnt2,tt),pp), m, Ma, 1)
        Cnt2[m-1,m-1,0] += 1
        pind = m-1
        while(Cnt2[m-1,pind,0] >= tt):
            Cnt2[m-1,pind,0] = np.mod(Cnt2[m-1,pind,0],tt)
            Cnt2[m-1,pind-1,0] += 1
            pind -= 1
        C = M16_Enc(M1[:,:,0,:],delta, X,Y2,t, nbits)
        CF[:,:,i]                                                                    =
np.mod(C+M[:,:,i],tt)#np.mod(C+M[:,:,i],np.power(2,t))#np.mod(np.bitwise_xor(C,M[:,:,i]),np.pow
er(2,t))
    return CF

def M16_CBC(IV,M, Ma, X, Y2, rounds, t, nbits, delta):
    m = X.shape[0]
    CF = np.zeros([m,m,rounds+1], dtype='int32')
    CF[:,:,0] = IV
    for i in range(rounds):
        M1                                                                           =
np.mod(CF[:,:,i]+M[:,:,i],np.power(2,t))#np.mod(np.bitwise_xor(CF[:,:,i],M[:,:,i]),np.power(2,t))
        M2 = FormM(M1.reshape([m,m,1]), m, Ma, 1)
        C = M16_Enc(M2[:,:,0,:],delta, X,Y2,t, nbits)
        CF[:,:,i+1] = C
    return CF

----------CBC.py-------------
from PIL import Image
import numpy as np
import math
from ESakCipher import *
from timeit import default_timer as timer
from datetime import timedelta
import pickle
import scipy.stats as stats
from modes import E_sak_CBC as Forward, Form_pic_blocks, Form_text_blocks

def CBC_F(m, file_name, save = False, rgb = 'L', nb = 8):
    #file_name = 'linux'
```

```
    #Algorithm: M16, Esak
    file_folder = 'Esak'
    #Mode: CTR, CBC
    mode = 'CBC'
    im = Image.open(f"images/{file_name}").convert(rgb)#Image.open("coloredChips.png")
    X1 = np.asarray(im)
    if(nb == 4):
        p = 47
    elif (nb == 8):
        p = 563
    elif (nb == 16):
        p = 131267
    if(rgb == 'L'):
        prgb = 1
    else:
        prgb = 3
    q, G, X, Y, Z = Gen_parameters(m,p)
    start = timer()
    s1, s2, M1 = Form_pic_blocks(m, X1,prgb, nb=nb)
    print("Generavimo laikai - ",timer()-start)
    #M1 = Form_text_blocks(m, 'Generated.txt', nb = nb)

    IV = np.zeros([m,m], dtype=int)
    Nblocks = M1.shape[2]

    start = timer()
    C = Forward(IV,M1, G, X,Y,Z,  Nblocks, p, q)
    end = timer()
    print(timedelta(seconds=end-start))
    unique, counts = np.unique(np.array(C[:,:,1:]), return_counts=True)
    q = int(q)
    print(stats.chisquare(f_obs=counts, f_exp=np.ones(q, dtype = 'int64')*(Nblocks*m*m/q)))
    if(save):
        file = open(f'res/{mode}/{file_folder}/{file_name}_4x4_addmod', 'wb')
        pickle.dump([C, M1, s1, s2], file)
        #pickle.dump([C, M1], file)
        file.close()
    return end-start


CBC_F(4, 'cameraman.tif',save = False, nb = 16)

#---------------M16.py------------

import numpy as np
import math
```

```
from Basic_functions.main_func import Shifting_bits

def m16mult(a, b, t):
    """
    Multiplication operation defined over M_{2^t}
    """
    C = np.zeros([2])
    C[0] = np.mod(a[0]+b[0],2)
    if(np.mod(a[1],2)==0):
        C[1] = np.mod(a[1]+b[1], np.power(2,t-1))
    if(np.mod(a[1],2)==1):
        if(np.mod(b[0],2)==0):
            C[1] = np.mod(a[1]+b[1], np.power(2,t-1))
        else:
            C[1] = np.mod(a[1]+b[1]+np.power(2,t-2),np.power(2,t-1))
    return C


def m16exp(a, n, t):
    """
    Power operation defined over M_{2^t}
    """
    b = np.zeros([2])
    if(np.mod(a[0],2)):
        b[0] = np.mod(n,2)
        if(np.mod(a[1],2)):
            b[1]=np.mod(a[1]*n+np.power(2,t-2)*math.floor(n/2),np.power(2,t-1))
        else:
            b[1]=np.mod(a[1]*n,np.power(2,t-1))
    else:
        b[0]=0
        b[1] = np.mod(a[1]*n,np.power(2,t-1))
    return b


def m16_matrix_exp_right(W,X,t):
    """
    MPF from the right side over M_{2^t}
    """
    m, _, _ = W.shape
    _, n = X.shape
    B = np.zeros([m,n,2])
    for i in range(m):
        for j in range(n):
            B[i,j,0] = 0
            B[i,j,1] = 0
```

```
            for k in range(m):
                B[i,j,:] = m16mult(B[i,j,:], m16exp(W[i,k,:], X[k,j],t), t)
    return B.astype('int64')

def m16_matrix_exp_left(W,X,t):
    """
    MPF from the left side over M_{2^t}
    """
    m, _, _ = W.shape
    _, n = X.shape
    B = np.zeros([m,n,2])
    for i in range(m):
        for j in range(n):
            B[i,j,0] = 0
            B[i,j,1] = 0
            for k in range(m):
                B[i,j,:] = m16mult(B[i,j,:], m16exp(W[k,j,:], X[i,k],t), t)
    return B.astype('int64')


def M16_Enc(M,delta, X,Y,t, nbits):
    """
    Encryption algorithm defined over M_{2^t}
    """
    m,_ = X.shape
    M_inp = M.copy()
    M_inp[:,:,1] = np.mod(M_inp[:,:,1]+X,np.power(2,t-1))
    M_inp[:,:,0] = np.mod(delta[:,:,0]+M_inp[:,:,0],2)
    C1 = M_inp
    C2 = m16_matrix_exp_left(C1,Y,t)
    C2_2 = m16_matrix_exp_right(C2,Y,t)
    C3 = np.zeros([m,m])
    DX = C3.copy()
    for i in range(m):
        for j in range(m):
            C3[i,j] = int(Shifting_bits(bin(C2_2[i,j,0])+bin(C2_2[i,j,1])[2:].zfill(t-1),nbits),2)
            DX[i,j] = np.mod(int(bin(delta[i,j,0])+bin(X[i,j])[2:].zfill(t-1),2),np.power(2,t))
    C4 = np.mod(C3+DX, np.power(2,t))
    return C4.astype('int64')


def M16_Dec(C,delta, X,YY,t, nbits):
    """"
    Decryption algorythm defined over M_{2^t}
    """
    m,_ = X.shape
```

```
    DX = np.zeros([m,m])
    for i in range(m):
        for j in range(m):
            DX[i,j] = np.mod(int(bin(delta[i,j,0])+bin(X[i,j])[2:].zfill(t-1),2),np.power(2,t))
    D1 = np.mod(C-DX, np.power(2,t)).astype('int64')
    D2 = np.zeros([m,m,2])
    D5 = D2.copy()
    for i in range(m):
        for j in range(m):
            PD = Shifting_bits('0b'+bin(D1[i,j])[2:].zfill(t),t-nbits)
            D2[i,j,0] = int(PD[:2]+PD[2],2)
            D2[i,j,1] = int(PD[:2]+PD[3:],2)
    D3 = m16_matrix_exp_right(D2, YY,t)
    D4 = m16_matrix_exp_left(D3, YY,t)
    D5[:,:,1] = np.mod(D4[:,:,1]-X, np.power(2,t-1))
    D5[:,:,0] = np.mod(delta[:,:,0]+D4[:,:,0],2)
    return D5.astype('int64')


def FormM(C, m, Ma, amount):
    """
    Mapping M elements to M_b & M_a elements
    """
    M = np.zeros([m,m,amount,2])
    for a in range(amount):
        #print(a/amount*100)
        for i in range(m):
            for j in range(m):
                M[i,j,a,0] = Ma[C[i,j, a],0]
                M[i,j,a,1] = Ma[C[i,j, a],1]
    return M.astype('int64')


def perm_matrix(n,t):
    """
    Permutation matrix generation
    """
    A = np.eye(n)
    A = np.random.permutation(A)
    B = np.mod(np.random.randint(1,np.power(2,t-2)+1,size=(n,n))*2,np.power(2,t-1))
    pag = np.mod(np.random.randint(1,np.power(2,t-2)+1,size=(n,n))*2-1,np.power(2,t-1))
    D = np.mod(B-B*A+pag*A, np.power(2,t-1))
    return D.astype('int64')

def MakeVector(m):
    """
```

```
    Constructin permutation vector
    """
    Vec = np.arange(0,m*m)
    return np.random.permutation(Vec)

def BuildPlaintext(VVec, m, M,pp):
    """
    Form plaintext using permutation vector mapping
    """
    if(VVec == None):
        return M
    index = 0
    Vec = VVec[np.mod(M.sum(dtype='int64'),pp)]
    NM = np.zeros(shape=M.shape)
    for i in range(m):
        for j in range(m):
            NM[i,j] = M[Vec[index]//m,Vec[index]%m]
            index += 1
    return NM.astype('int64')

#-------------CTR.py------------------
from PIL import Image
import numpy as np
import math
from ESakCipher import *
from timeit import default_timer as timer
from datetime import timedelta
import pickle
import scipy.stats as stats
from modes import E_sak_CTR as Forward, Form_pic_blocks , Form_text_blocks

def CTR_F(m, file_name, save = False, rgb = 'L', nb = 8):
    #file_name = 'linux'
    #Algorithm: M16, Esak
    file_folder = 'Esak'
    #Mode: CTR, CBC
    mode = 'CTR'
    im = Image.open(f"images/{file_name}").convert(rgb)#Image.open("coloredChips.png")
    X1 = np.asarray(im)
    if(nb == 4):
        p = 47
    elif (nb == 8):
        p = 563
    elif (nb == 16):
        p = 131267
    if(rgb == 'L'):
```

```
        prgb = 1
    else:
        prgb = 3
    q, G, X, Y, Z = Gen_parameters(m,p)
    start = timer()
    s1, s2, M1 = Form_pic_blocks(m, X1,prgb, nb = nb)
    print("Generavimo laikai - ",timer()-start)
    #M1 = Form_text_blocks(m, 'Generated.txt', nb = nb)

    Nblocks = M1.shape[2]

    start = timer()
    C = Forward(M1, G, X, Y, Z, Nblocks, p, q)
    end = timer()
    print(timedelta(seconds=end-start))
    #unique, counts = np.unique(np.array(C), return_counts=True)
    #q = int(q)
    #print(stats.chisquare(f_obs=counts, f_exp=np.ones(q, dtype = 'int64')*(Nblocks*m*m/q)))
    if(save):
        file = open(f'res/{mode}/{file_folder}/{file_name}_4x4_addmod', 'wb')
        pickle.dump([C, M1, s1, s2], file)
        #pickle.dump([C, M1], file)
        file.close()
    return end-start
TM = 0
FT = []

CTR_F(4, 'Generated8',save = False, nb = 16)


#---------------M16CBC.py-------------------

from PIL import Image
import numpy as np
import math
from timeit import default_timer as timer
from datetime import timedelta
import pickle
from M16 import *
import scipy.stats as stats
from modes import Form_pic_blocks, M16_CBC as Forward,Form_text_blocks

"""
im = Image.open("coloredChips.png").convert('L')
image = np.asarray(im)
Image.fromarray(image).show()
```

```
im.show()
"""

def M16CBC_F(m, file_name, save = False, rgb = 'L', nb = 8):
    #file_name = 'linux'
    #Algorithm: M16, Esak
    file_folder = 'M16'
    #Mode: CTR, CBC
    mode = 'CBC'

    im = Image.open(f"images/{file_name}").convert(rgb)#Image.open("coloredChips.png")
    X1 = np.asarray(im)

    t = nb
    nbits = 2
    X = np.random.randint(0,np.power(2,t-1),size=(m,m))
    Y2 = perm_matrix(m,t)

    Ma = np.zeros([np.power(2,t),2])
    for a in range(np.power(2,t)):
        Ma[a,0] = math.floor(a/np.power(2,t-1))
        Ma[a,1] = np.mod(a,np.power(2,t-1))
    if(rgb == 'L'):
        prgb = 1
    else:
        prgb = 3
    start = timer()
    s1, s2, M1 = Form_pic_blocks(m, X1,prgb, nb = nb)
    print("Generavimo laikai - ",timer()-start)
    #M1 = Form_text_blocks(m, 'Generated.txt', nb = nb)
    #M1 = FormM(M1, m, Ma, 1)
    delta = np.zeros([m,m,2]).astype('int64')
    delta[:,:,0] = np.random.randint(0,2,size=(m,m))

    Nblocks = M1.shape[2]
    IV = np.zeros([m,m], dtype=int)
    start = timer()
    C = Forward(IV,M1, Ma, X, Y2, Nblocks, t, nbits, delta)
    end = timer()
    #print(timedelta(seconds=end-start))

    #unique, counts = np.unique(np.array(C[:,:,1:]), return_counts=True)
    #print(stats.chisquare(f_obs=counts, f_exp=np.ones(np.power(2,t), dtype =
'int64')*(Nblocks*m*m/np.power(2,t))))
    if(save):
        file = open(f'res/{mode}/{file_folder}/{file_name}_4x4_addmod', 'wb')
```

```
        pickle.dump([C, M1, s1, s2], file)
        #pickle.dump([C, M1], file)
        file.close()
    return end-start

#M16CBC_F(4, 'cameraman.tif',save = False, nb = 16)

#--------------------M16CTR.py-----------------------
from PIL import Image
import numpy as np
import math
from timeit import default_timer as timer
from datetime import timedelta
import pickle
from M16 import *
import scipy.stats as stats
from modes import Form_pic_blocks, M16_CTR as Forward,Form_text_blocks

"""
im = Image.open("coloredChips.png").convert('L')
image = np.asarray(im)
Image.fromarray(image).show()
im.show()
"""

def M16CTR_F(m, file_name, save = False, rgb = 'L', nb = 8):
    #file_name = 'coloredChips_NoY'
    #file_name = 'linux'
    #Algorithm: M16, Esak
    file_folder = 'M16'
    #Mode: CTR, CBC
    mode = 'CTR'

    im = Image.open(f"images/{file_name}").convert(rgb)#Image.open("coloredChips.png")
    X1 = np.asarray(im)

    t = nb
    nbits = 2
    X = np.random.randint(0,np.power(2,t-1),size=(m,m))
    #Y2 = perm_matrix(m,t)
    Y2 = np.random.randint(0,np.power(2,t-1),size=(m,m))

    Ma = np.zeros([np.power(2,t),2])
    for a in range(np.power(2,t)):
        Ma[a,0] = math.floor(a/np.power(2,t-1))
        Ma[a,1] = np.mod(a,np.power(2,t-1))
```

```
    if(rgb == 'L'):
        prgb = 1
    else:
        prgb = 3
    s1, s2, M1 = Form_pic_blocks(m, X1,prgb, nb = nb)
    #M1 = Form_text_blocks(m, 'Generated.txt', nb = nb)
    #M = FormM(M1, m, Ma, 1)
    delta = np.zeros([m,m,2]).astype('int32')
    delta[:,:,0] = np.random.randint(0,2,size=(m,m))

    Nblocks = M1.shape[2]
    start = timer()
    C = Forward(M1, Ma, X, Y2, Nblocks, t, nbits, delta)
    end = timer()
    #print(timedelta(seconds=end-start))
    #unique, counts = np.unique(np.array(C), return_counts=True)
    #print(stats.chisquare(f_obs=counts, f_exp=np.ones(np.power(2,t), dtype =
'int32')*(Nblocks*m*m/np.power(2,t))))
    if(save):
        file = open(f'res/{mode}/{file_folder}/{file_name}_4x4_NoY_addmod_cntr7', 'wb')
        pickle.dump([C, M1, s1, s2], file)
        #pickle.dump([C, M1], file)
        file.close()
    return end-start

sk = 100
for i in range(sk):
    tmp = M16CTR_F(4, 'Generated8',save = False, nb = 8)
    if(i == 0):
        CC = np.zeros([tmp.shape[0],tmp.shape[1],tmp.shape[2],sk])
    CC[:,:,:,i] = tmp[:,:,:]

file = open(f'res/CTR/M16/Full_text_C', 'wb')
pickle.dump(CC, file)
file.close()

#M16CTR_F(4, 'Generated8',save = True, nb = 8)

#---------------Time_comparison.py-----------------
from CBC import CBC_F
from CTR import CTR_F
from M16CBC import M16CBC_F
from M16CTR import M16CTR_F
import pickle
from datetime import timedelta
```

```
TM = 0
TM2 = 0
TM3 = 0
TM4 = 0
FT = []
FT2 = []
FT3 = []
FT4 = []

nr = 10
for m in range(3,11):
    TM = 0
    TM2 = 0
    TM3 = 0
    TM4 = 0
    for i in range(nr):
        print(f"m = {m}")
        TM += CBC_F(m,'cameraman.tif', nb = 16)
        TM2 += CTR_F(m,'cameraman.tif', nb = 16)
        TM3 += M16CBC_F(m,'cameraman.tif', nb = 16)
        TM4 += M16CTR_F(m,'cameraman.tif', nb = 16)
    FT.append(TM/nr)
    FT2.append(TM2/nr)
    FT3.append(TM3/nr)
    FT4.append(TM4/nr)


file = open(f'res/time_comp_Cameraman_16b', 'wb')
pickle.dump([FT, FT2, FT3, FT4], file)
file.close()
for i in range(len(FT)):
    print(f"m = {i+3}, ",timedelta(seconds=FT[i]))


#-----------------Time_bars.py-------------------
import matplotlib.pyplot as plt
import pandas as pd
import pickle
import numpy as np

plt.rcParams['text.usetex'] = True
file = open(f'res/time_comp_Cameraman_16b', 'rb')
L1CBC, L2CTR, L3M16CBC, L4M16CTR = pickle.load(file)
xlab = ['CBC', 'CTR', '']
print(L1CBC)
print(L2CTR)
```

```
print(L3M16CBC)
print(L4M16CTR)
lbl = []
for i in range(len(L3M16CBC)):
    if(len(L1CBC) > 0 and len(L3M16CBC) > 0):
        lbl.append(f'CBC (m={3+i})')
        lbl.append(f'CTR (m={3+i})')
        lbl.append(r'$M_{2^t}CBC$ '+f'(m={3+i})')
        lbl.append(r'$M_{2^t}CTR$ '+f'(m={3+i})')
        plt.bar(i+0, L1CBC[i], color='#fca289', width = 0.2)
        plt.bar(i+0.25, L2CTR[i], color = '#eec448', width = 0.2)
        plt.bar(i+0.5, L3M16CBC[i], color='#aacc81', width = 0.2)
        plt.bar(i+0.75, L4M16CTR[i],color='#86c5bf', width = 0.2)
    elif (len(L3M16CBC) > 0):
        lbl.append(r'$M_{2^t}CBC$ '+f'(m={3+i})')
        lbl.append(r'$M_{2^t}CTR$ '+f'(m={3+i})')
        plt.bar(i, L3M16CBC[i], color='#aacc81', width = 0.2)
        plt.bar(i+0.5, L4M16CTR[i],color='#86c5bf', width = 0.2)

if(len(L1CBC) > 0 and len(L3M16CBC) > 0):
    plt.xticks([0.25*i for i in range(len(lbl))], lbl, size='small')
elif (len(L3M16CBC) > 0):
    plt.xticks([0.5*i for i in range(len(lbl))], lbl, size='small')
plt.xticks(rotation=85)
plt.ylabel('Laikas (s)')
plt.tight_layout()
plt.show()

#-----------Hist_plot.py--------------------
import matplotlib.pyplot as plt
import pickle
import numpy as np
from PIL import Image

def Print_pic(C, s1, s2, d3):
    m = C[:,:,0].shape[0]
    if(d3):
        rgb = 3
        FC = np.zeros([s1,s2,3])
    else:
        rgb = 1
        FC = np.zeros([s1,s2])
    indx = 0
    for k in range(rgb):
        for i in range(s1//m):
            for j in range(s2//m):
```

```
            if(d3):
                FC[i*m:(i+1)*m,j*m:(j+1)*m,k] = C[:,:,indx]
            else:
                FC[i*m:(i+1)*m,j*m:(j+1)*m] = C[:,:,indx]
            indx += 1
    return np.array(FC)

file_name = 'linux_4x4'
#Algorithm: M16, Esak
file_folder = 'Esak'
#Mode: CTR, CBC
mode = 'CBC'
#RGB: 1 - RGB; 0 - Grayscale
rgb_TF = 1

file = open(f'res/{mode}/{file_folder}/{file_name}', 'rb')
mode = 'CTR'
data, M1, s1, s2 = pickle.load(file)
file.close()
if(mode == 'CBC'):
    ans = Print_pic(data[:,:,1:], s1, s2, rgb_TF)
elif(mode == 'CTR'):
    ans = Print_pic(data[:,:,:], s1, s2, rgb_TF)

fig, ax = plt.subplots(figsize =(10, 7))
#unique, counts = np.unique(np.mod(np.array(ans),256), return_counts=True)
#ans = Print_pic(M1, s1, s2, rgb_TF)
unique, counts = np.unique(np.array(ans), return_counts=True)
ax.bar(unique, counts,width=1,  edgecolor='black')
plt.xlabel('Pikselio vertė')
plt.ylabel('Dažnis')
plt.show()

#-------------M16_show_pictures.py-----------------
import pickle
import numpy as np
from PIL import Image

#Filename: freely chosen
file_name = 'coloredChips'
#Algorithm: M16, ESak
file_folder = 'AES'
#Mode: CTR, CBC
mode = 'CTR'
#RGB: 1 - RGB; 0 - Grayscale
rgb_TF = 1
```

```
rez_path = f"..\Figures\{mode}\AES\P{file_name}_RGB.jpg"

file = open(f'res/{mode}/{file_folder}/{file_name}', 'rb')
mode = 'CTR'
data, M1, s1, s2 = pickle.load(file)

print(data.shape)
file.close()
if(file_folder == 'ESak'):
    p = 563
    q = (p-1)/2
    data = np.mod(data,256)

def Print_pic(C, s1, s2, d3):
    m = C[:,:,0].shape[0]
    if(d3):
        rgb = 3
        FC = np.zeros([s1,s2,3])
    else:
        rgb = 1
        FC = np.zeros([s1,s2])
    indx = 0
    for k in range(rgb):
        for i in range(s1//m):
            for j in range(s2//m):
                if(d3):
                    FC[i*m:(i+1)*m,j*m:(j+1)*m,k] = C[:,:,indx]
                else:
                    FC[i*m:(i+1)*m,j*m:(j+1)*m] = C[:,:,indx]
                indx += 1
    if(d3):
        im = Image.fromarray(FC[:,:,:].astype(np.uint8)).convert('RGB')
    else:
        im = Image.fromarray(FC).convert('L')
    return im

if(mode == 'CBC'):
    ans = Print_pic(data[:,:,1:], s1, s2, rgb_TF)
elif(mode == 'CTR'):
    ans = Print_pic(data[:,:,:], s1, s2, rgb_TF)

ans.save(rez_path)


#ans.show()
```