KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS

Marius Geležinis

NUTOLUSIOS FAILŲ SISTEMOS ŠIFRAVIMO METODO SUDARYMAS IR TYRIMAS

Baigiamasis magistro darbas

Vadovas
doc. dr. Nerijus Morkevičius

KAUNAS, 2017

KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

KOMPIUTERIŲ KATEDRA

NUTOLUSIOS FAILŲ SISTEMOS ŠIFRAVIMO METODO SUDARYMAS IR TYRIMAS

Baigiamasis magistro darbas

Informacijos ir informacinių technologijų sauga (kodas 621E10003)

Vadovas

doc. dr. Nerijus Morkevičius

Recenzentas

dr. Kęstutis Lukšys

Projektą atliko

Marius Geležinis

KAUNAS, 2017

KAUNO TECHNOLOGIJOS UNIVERSITETAS

(Fakultetas)

(Studento vardas, pavardė)

(Studijų programos pavadinimas, kodas)

„Nutolusios failų sistemos šifravimo metodo sudarymas ir tyrimas“

AKADEMINIO SĄŽININGUMO DEKLARACIJA

20 17 m. gegužės 22 d.

Kaunas

Patvirtinu, kad mano **Mariaus Geležinio** baigiamasis projektas tema „Nutolusios failų sistemos šifravimo metodo sudarymas ir tyrimas“ yra parašytas visiškai savarankiškai, o visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

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

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

Geležinis, M. „Nutolusios failų sistemos šifravimo metodo sudarymas ir tyrimas“. Magistro baigiamasis projektas / vadovas doc. dr. Nerijus Morkevičius; Kauno technologijos universitetas, informatikos fakultetas, kompiuterių katedra.

Kaunas, 2017. 61 p.

SANTRAUKA

Populiarėjant debesies kompiuterijai vis daugiau žmonių ir organizacijų naudojasi jų teikiamomis failų talpinimo paslaugomis. Taip atsiranda naujų informacijos saugumo problemų. Saugant duomenis nutolusiose failų sistemose kyla jų konfidencialumo problema – vartotojui nežinant pašaliniai asmenys gali gauti prieigą prie jų.

Išanalizavus failų šifravimo metodus buvo išsiaiškinta, kad didžiausia problema yra didelis perduodamų duomenų kiekis. Labiausiai paplitusios tinklinės failų sistemos nesuteikia galimybės atlikti dalinio failų išsaugojimo, t. y. įvykus failo pakitimui, visas jo turinys yra siunčiamas į nutolusį serverį. Taip pat nustatyta, kad net ir kelių baitų pridėjimas ar ištrynimas gali visiškai pakeisti šifruotą failą.

Išanalizavus failų šifravimo metodų ir programų privalumus bei trūkumus sudarytas naujas failų šifravimo metodas, sprendžiantis nustatytus trūkumus. Šifravimo metu failai yra skaidomi į mažesnius failus, kurie saugomi serveryje. Šie failai gali būti įvairaus dydžio, fiksuotas tik maksimalus. Tokiu būdu sprendžiama nustatyta problema su tinklinėmis failų sistemomis, nes pakitus failui, į serverį perduodamas tik maža dalis, kuri keitėsi.

Sudarytas naujas šifravimo metodas realizuotas ir eksperimentiškai palygintas su kitais sprendimais. Atlikti šifravimo bei perduodamų duomenų kiekio tyrimai keičiant, pridedant ir ištrinant baitus įvairiose failo vietose. Nustatyta, kad priklausomai nuo atliekamo pakeitimo ir failo dydžio, vienais atvejais šifruotas failas mažiau pakinta naudojant sukurtą prototipą, kitais atvejais – vieną iš esamų sprendimų. Tačiau kai failo dydis didesnis, negu 4 KB ir faile atlikti nedideli pakeitimai, šifruojant pasiūlytu metodu į serverį perduodama kur kas mažiau duomenų. Kitais atvejais perduodamas duomenų kiekis sutampa.

Geležinis, Marius. *Creation and investigation of remote file system encryption method: Master's* thesis in Information and Information Technology Security / supervisor assoc. prof. Nerijus Morkevičius. The Faculty of informatics, Kaunas University of Technology.

Research area and field: file encryption

Key words: file encryption, remote file system

Kaunas, 2017. 61 p.

SUMMARY

As cloud computing are getting more and more popular, number of people and organizations using their provided data storage services are growing every day. This leads to increased risk of data confidentiality as an unauthorized person can get access to someone's files without owner's knowledge.

File encryption methods analysis results shows that biggest problem is huge data bandwidth. Most widely used network file systems does not support partial file update, that means every time file changes all of its content is sent to server, even if only one byte was changed. Also during the analysis was found that by adding or removing few bytes encrypted file can change dramatically.

New remote file encryption method was composed after analyzing advantages and disadvantages of existing solutions. This new method resolves main disadvantages found during analysis. During encryption file is split into smaller dynamic size files and encrypted separately. Only maximum file size is fixed. This method is used in order to solve problem with network file systems. After file changes, only the small file that changed is sent to server.

According to composed method prototype was created and compared to existing solutions. Investigation included encryption and transferred data to server while changing, adding and removing bytes in different locations of the file. Results shows that depending on the change that was made and file size, in some cases encrypted file changes less when encrypted using prototype, in other cases – using one of the existing solutions. However, when file size is more than 4 KB and small changes made to the file, a lot less data transferred to server when encrypted with prototype. In other cases, transferred data is equal.

TURINYS

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

TERMINŲ IR SANTRUMPŲ ŽODYNAS

IV (Inicializacijos vektorius) – atsitiktinė (bet kokio turinio) duomenų seka, pradedanti užšifruotą tekstą.

MAC (angl. *Message Authentication Code*) – pranešimo autentifikavimo kodas.

OS (angl. *Operation System*) – operacinė sistema.

SSD (angl. *Solid State Drive*) – puslaidininkinis diskas.

NIST (angl. *National Institute of Standards and Technology*) – Nacionalinis Standartų ir Technologijos Institutas.

Data at rest – duomenys fiziškai saugomi kurioje nors skaitmeninėje formoje.

Data in transit – duomenys perduodami tinklu.

Pagrindinis užkrovimo sektorius (angl. *Master boot sector*) – specialus sektorius kompiuterio kietajame diske, kuriame saugoma operacinės sistemos paleidimo instrukcijos.

FDE (angl. *Full Disc Encryption*) – viso disko šifravimas.

Demonas (angl. *daemon*) – specialios paskirties programa, dirbanti, kaip foninis procesas.

ĮVADAS

Darbas priklauso Informacijos ir informacinių sistemų saugos studijų programai.

Darbo problematika ir aktualumas

Populiarėjant debesies kompiuterijai vis daugiau žmonių ir organizacijų naudojasi jų teikiamomis failų talpinimo paslaugomis. Tai palengvina failų dalinimąsi tarpusavyje, sinchronizavimą tarp skirtingų vartotojo įrenginių, bei atsarginių duomenų kopijų saugojimą. Tačiau tuo pačiu atsiranda ir naujų informacijos saugumo problemų. Saugant duomenis nutolusiose failų sistemose kyla jų konfidencialumo problema – vartotojui nežinant pašaliniai asmenys gali gauti prieigą prie jų.

Darbo tikslas ir uždaviniai

Pagrindinis darbo tikslas yra pasiūlyti naują failų šifravimo metodą, kuris, lyginant su egzistuojančiais metodais, sumažintų perduodamų duomenų kiekį į serverį po failo pakitimo, kai failai saugomi serveryje ir vartotojo kompiuteryje pasiekiami naudojant tinklinę failų sistemą.

Šiam tikslui pasiekti yra sprendžiami tokie uždaviniai:

1. išanalizuoti esamus failų šifravimo metodus ir nustatyti jų trūkumus;
2. pasiūlyti tinkamą failų šifravimo metodą iškeltam tikslui pasiekti;
3. praktiškai realizuoti pasiūlytą metodą;
4. ištirti ir eksperimentiškai įvertinti metodo efektyvumą.

Darbo rezultatai ir jų svarba

Šiame darbe pasiūlytas naujas failų sistemos šifravimo metodas, kuris, lyginant su egzistuojančiais metodais, sumažina perduodamų duomenų kiekį į serverį pakitus failui, kai failai yra saugomi serveryje. Pasiūlyto metodo pagrindu realizuotas prototipas, kurio efektyvumas patvirtintas eksperimentiškai.

Darbo struktūra

Darbas susideda iš 4 skyrių:

1. analizės dalyje yra pateikiama skirtingų failų šifravimo metodų ir esamų sprendimų analizė;
2. projektavimas dalyje aprašomas siūlomas naujas failų šifravimo metodas;
3. tyrimo dalyje aprašomi atlikti eksperimentai ir pateikiami rezultatai;
4. išvadose yra pateikiamos darbo išvados.

1. FAILŲ ŠIFRAVIMO METODŲ ANALIZĖ

Šios analizės tikslas yra išanalizuoti esamus duomenų šifravimo ir failų sinchronizavimo metodus, išsiaiškinti failų šifravimo problematiką ir išanalizuoti jau egzistuojančius sprendimus.

Saugant failus nutolusiose serveriuose kyla informacijos konfidencialumo pažeidimo rizika. Vartotojas tiksliai nežino, kur iš tiesų stovi serveriai, kuriuose yra saugomi jo failai, ir kas turi prieigą prie jų.

1.1. Analizės tikslas

Šios analizės tikslas yra išanalizuoti esamus duomenų šifravimo ir failų sinchronizavimo būdus, išsiaiškinti failų šifravimo problematiką ir išanalizuoti jau egzistuojančius sprendimus.

1.2. Tyrimo objektas, sritis ir problema

Šio darbo tyrimo sritis yra duomenų šifravimas bei šifruotų failų sinchronizavimas debesyje.

Tyrimo objektas – failų šifravimo metodai.

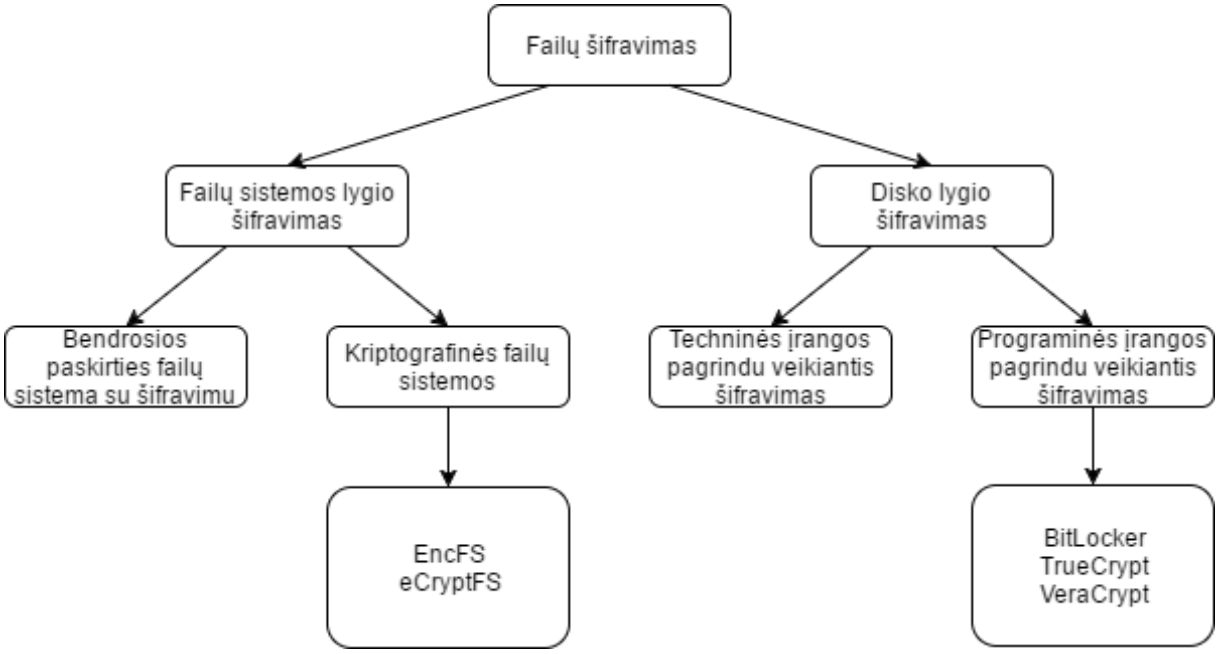
Sprendžiama šifruotų failų efektyvaus sinchronizavimo nutolusiame serveryje problema.

1.3. Failų saugojimo nutolusiame serveryje problemos

Failo išsaugojimo nutolusiame serveryje procese galima išskirti du esminius etapus, kada kyla informacijos konfidencialumo pažeidimo rizika: duomenų persiuntimo metu (angl. *data in transit*) ir duomenų saugojimo metu (angl. *data at rest*). Duomenys yra persiunčiami viešu, nesaugiu tinklu, prie kurio priėjimą turi pašaliniai asmenys. Yra įvairių metodų, kurių pagalba, vartotojui nežinant, piktavaliai gali perimti viešu tinklu perduodamus duomenis. Kriptografijos pagalba galima apsisaugoti nuo tokių atakų šifruojant failus perdavimo metu (angl. *data in transit*). Tokiu atveju visi duomenys prieš iškeliaujant iš kompiuterio į tinklą yra užšifruojami ir pasiekę galutinį tašką – iššifruojami.

1.4. Failų šifravimas

Analizės metu išnagrinėti skirtingi failų šifravimo metodai ir kai kurių metodų esami sprendimai, kurie pateikti 1 pav. Nustatytos dvi failų šifravimo metodų grupės: failų sistemos lygio ir disko lygio. Failų sistemos lygio šifravimas dar skaidomas į bendrosios paskirties failų sistemą su šifravimu, kurios šifruoja tik failų turinį, ir kriptografines failų sistemas, kurios šifruoja ir turinį ir metaduomenis. Disko lygio šifravimas dar skaidomas į veikiančius techninės įrangos pagrindu ir programinės įrangos pagrindu.



1 pav. Failų šifravimas

Išanalizuoti kriptografinės failų sistemos bei programinės įrangos pagrindu veikiančio disko lygio šifravimo esami produktai, nes jiems nereikia specialios techninės įrangos ir šifruoja ne tik failą, bet ir jo metaduomenis.

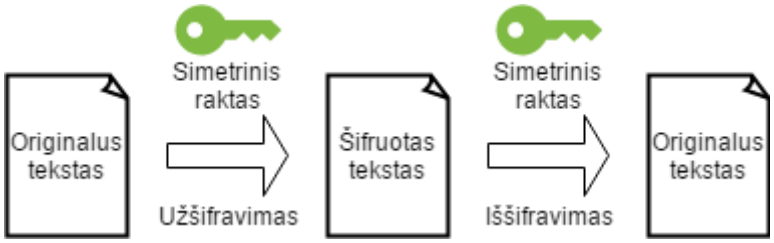
1.5. Šifravimo algoritmai

Kriptografija – mokslas naudojantis matematinius algoritmus duomenims užšifruoti ir iššifruoti. Tai leidžia apsaugoti informaciją ir tik autorizuoti asmenys gali ją perskaityti. Tam yra naudojami tam tikri raktai, kurie dažniausiai pasirenkami atsitiktinai. Yra išskiriamos dvi šifravimo metodų grupės:

a. simetrinis šifravimas;
b. asimetrinis šifravimas.

1.5.1. Simetrinis šifravimas

Simetrinio šifravimo algoritmuose, arba kitaip vadinamoje privačiojo rakto kriptografijoje, informacijai užšifruoti ir iššifruoti yra naudojamas tas pats slaptas raktas. Šio metodo pavyzdys pateiktas 2 pav. [1] Palyginti su asimetriniu šifravimu, simetrinio rakto kriptografija yra greitesnė ir paprastesnė, o raktai – trumpesni. [2]



2 pav. Simetrinis šifravimas

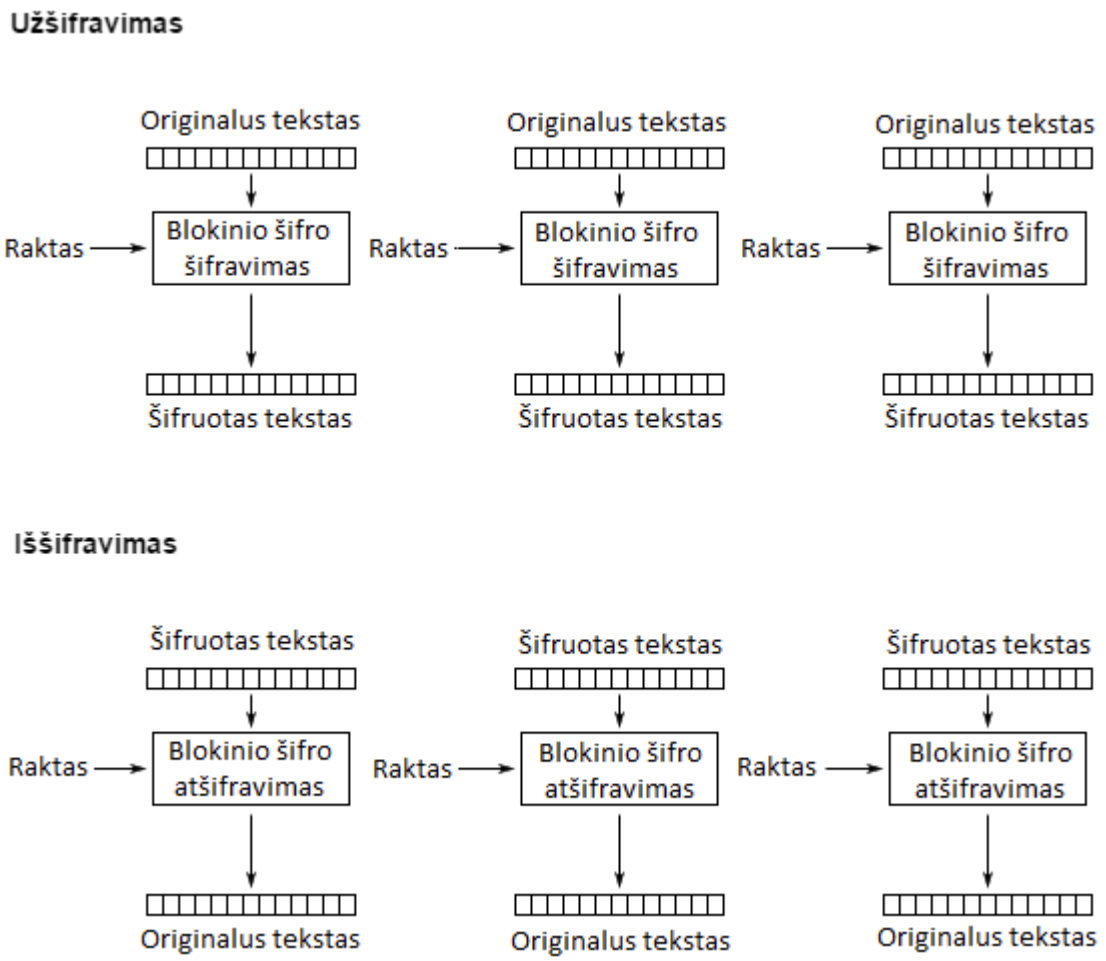
Šios grupės algoritmai dar skirstomi į srautinius ir blokinius. Srautinio tipo algoritmuose duomenys šifruojami imant po vieną bitą ir sudedama su rakto bitu naudojant loginę operaciją *XOR*. Po to, pagal tam tikras, iš anksto apibrėžtas matematines funkcijas, sugeneruojamas naujas raktas ir sudedamas *XOR* operacija su sekančiu informacijos bitu ir t. t. Srautinio tipo šifrai savo ruožtu dar skirstomi į sinchroninius (angl. *synchronous*) ir savaime susisinchronizuojančius (angl. *self-synchronizing*). Skirtumas tarp jų yra tas, kad sinchroninių šifrų raktų generacija priklauso nuo prieš tai buvusio šifruojamojo (arba iššifruojamojo) bito, o savaime susisinchronizuojančių šifrų raktų generacija nepriklauso. Šiuo atveju, jeigu yra naudojamas sinchroninis šifras ir iššifruojant failo duomenis, jame yra išimta ar pridėta arba pakeista keletas bitų, sugeneruojami raktai taps kitokie negu kad buvo užšifruojant šį bitą ir daugiau jo nebebus galima iššifruoti. Tuo tarpu savaime susisinchronizuojantys šifrai dekoduos tuos failo bitus, kurie nebus pakeisti (tarsi iššifruotų teisingai dalį failo).

Blokiniuose šifruose informacijos duomenys šifruojami įvairaus dydžio blokais po keletą bitų (pvz.: 56, 128) ir tuomet jie perskaičiuojami pasitelkus matematines funkcijas bei loginę operaciją *XOR* su raktu. Blokiniai šifrai taip pat turi keletą skirtingų modifikacijų, kaip užšifruoti informaciją. Pirmiausia reiktų pastebėti, kad naudojant blokinius šifrus dažnai galima sutikti terminą pradinis vektorius (angl. *initialization vector*) arba *IV*. Jis yra naudojamas pačioje šifravimo pradžioje ir turi būti tokio dydžio, kokio yra šifruojamieji blokai (pvz.: 128 bitai). Šiuo atveju prasidėjus šifravimo algoritmui, jis pirmiausia pasiims ne bitų bloką iš failo, bet *IV* ir nuo jo pradės šifravimo procesą. Jis naudojamas saugumui padidinti. Pagrindiniai blokinio šifravimo režimai:

a. elektroninės šifrų knygos režimas (*ECB*);
b. šifro bloko grandininis režimas (*CBC*);

c. grįžtamojo ryšio šifro režimas (*CFB*);
d. grįžtamojo ryšio išvesties režimas (*OFB*);
e. skaitiklio režimas (*CTR*).

Elektroninės šifrų knygos režimas yra pats paprasčiausias iš visų. Pranešimas skaidomas į blokus ir kiekvienas jų šifruojamas bei iššifruojamas atskirai (žr. 3 pav.). Tai leidžia visus blokus apdoroti lygiagrečiai. [3]



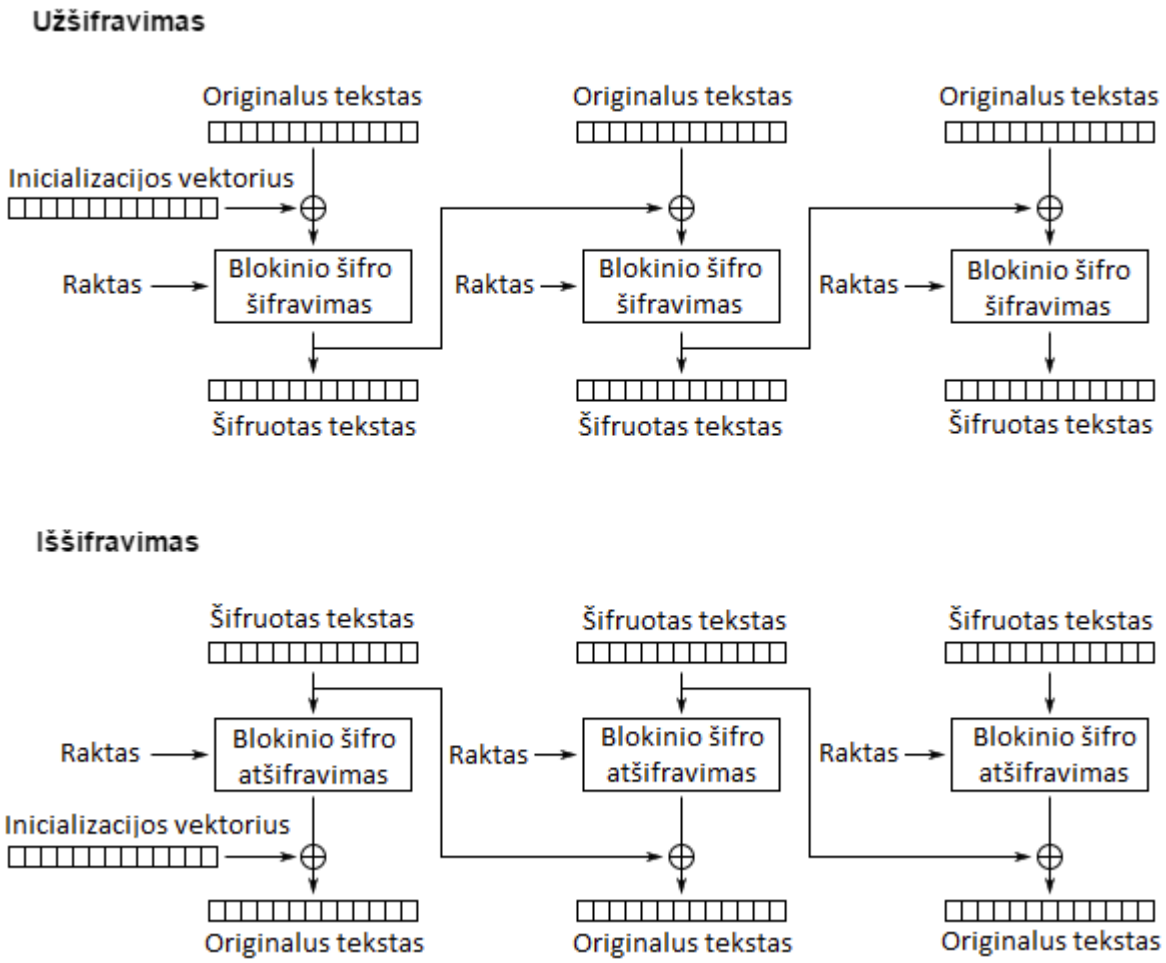
3 pav. Elektroninės šifrų knygos šifravimas ir iššifravimas

Didžiausi *ECB* režimo privalumai – paprastumas ir tai, kad šiuo režimu pranešimas neišplečiamas ir neperduodamos klaidos. Taip pat pakeitus 1 baitą, keičiasi tik blokas, kuriame yra pakeistas baitas. Tačiau šis režimas turi ir keletą rimtų trūkumų:

a. vienodi tekstogramos blokai priskiriami vienodiems šifrogramos blokams (kai naudojamas tas pats raktas). Tai nenaudinga, nes iš daugelio blokų sudaryta šifrograma gali atskleisti statistinę atitinkamos tekstogramos informaciją, net jei neįmanoma iššifruoti visos šifrogramos. Ši statistinė informacija yra būtent tai, ko įprastai ieško šifrogramų analitikai ir ką vienu ar kitu būdu bando išnaudoti.

b. *ECB* režimas neapsaugo šifrogramos blokų sekos. Gali keisti ilgą pranešimą paprasčiausiai ištrinant ar įrašant į jį pavienius blokus. Turint šifrogramos blokus, užšifruotus naudojant tą patį raktą, juos galima įterpti į šifrogramą. Nė vienu iš šių atvejų nereikia gebėti iššifruoti šifrogramos blokų, naudojamų atakai, tačiau galima sujaukti šifrogramą.

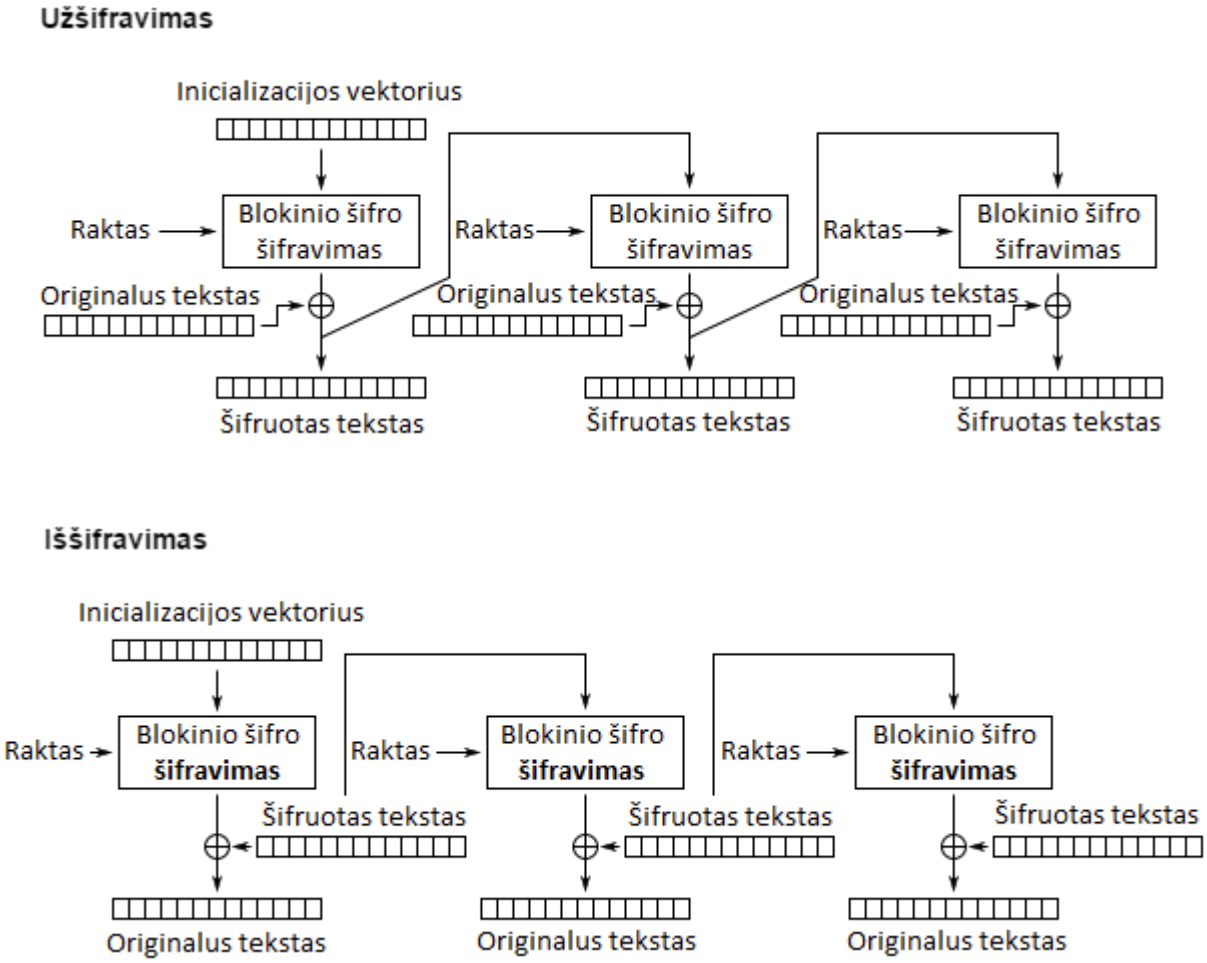
Šifro bloko grandininis šifravimo režimas buvo sukurtas siekiant panaikinti kai kuriuos *ECB* režimo trūkumus. Naudojant *XOR* operaciją prie šifruojamo teksto pridedamas prieš jį esantis užšifruotas blokas, taip užtikrinant, kad net ir vienodų pranešimo blokų užšifruotas tekstas bus skirtingas (žr. 4 pav.). [3]



4 pav. Šifro bloko grandininis šifravimas ir iššifravimas

Didžiausias *CBC* režimo privalumas tas, kad jis panaikina anksčiau minėtus *ECB* režimo trūkumus, tačiau turi keletą kitų trūkumų. Vienas jų – tai, kad šifruojant šiuo režimu pranešimas išplečiamas vienu bloku. Svarbesnis trūkumas yra tai, kad šifrogramos blokai yra susieti. Tai reiškia, kad įvykus šifravimo ar perdavimo klaidai kuriame nors bloke, klaida bus perduota ir į sekantį bloką. Taip pat vieno baito pakeitimas nešifruotam tekste pakeis visus sekančius blokus.

Grįžtamojo ryšio šifro režimas blokinį šifrą paverčia srautiniu. Blokinį šriftą jis pasitelkia tam, kad sugeneruotų pseudoatsitiktinių bitų seką. Toliau tie bitai pridedami prie tekstogramos bitų naudojant loginę operaciją *XOR* ir taip gaunami šifrogramos bitai. (žr. 5 pav.) [3]



5 pav. Grįžtamojo ryšio šifro šifravimas ir iššifravimas

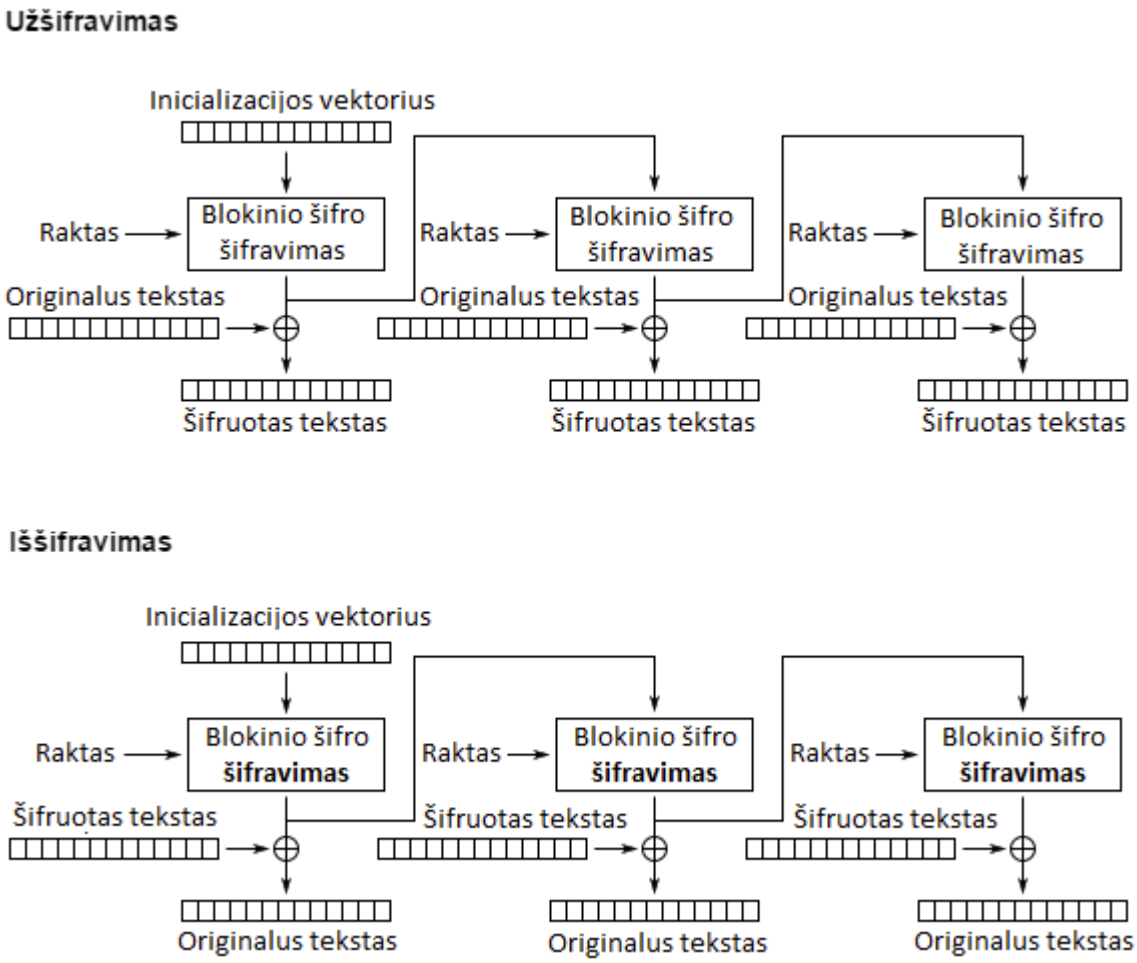
Didžiausias *CFB* režimo privalumas tas, kad jis blokinį šifrą paverčia srautiniu. Taip pat šiuo režimu galima užšifruoti blokus, mažesnius už blokinio šifro bloko ilgį. Tai praverčia tada, kai nereikia perduoti didelių pranešimų. Tačiau *CFB* turi keletą trūkumų:

a. didžiausias trūkumas yra sparta. Tam, kad būtų užšifruoti tik r bitai, reikia užšifruoti visus n bitus. Sparta priklauso ne tik nuo naudojamo simetrinio šifro, bet ir nuo parametro r dydžio ir jo santykio su bloko ilgiu n. Pavyzdžiui, jeigu *CFB* režimu naudojamas *AES* ir r yra 8 bitai, tuomet sparta yra n/r = 128/8 = 16 kartų lėtesnė nei *AES*, jį naudojat *ECB* ar *CBC* režimais.
b. parametro r dydis taip pat turi įtakos užšifravimo arba perdavimo klaidų plitimui kitiems šifrogramos blokams.
c. kaip jau minėta, užšifravimas yra paprasta sudėtis moduliu 2, todėl dėka gerų kriptografinių šios funkcijos savybių dažniausiai *CFB* režimu dirbantis blokinis šifras raktų srautui

sugeneruoti srautiniame šifre. Kadangi šis generavimas priklauso nuo šifrogramos bitų, kurie grąžinami į įvesties registrą, šis režimas ir vadinamas šifro grįžtamojo ryšio šifru.

Neturint šifrogramos neįmanoma iš anksto apskaičiuoti rakto bitų srauto suskirstytų r-bitų blokais, kurie naudojami šifravimo procese. Tai yra didelis *CFB* režimo privalumas.

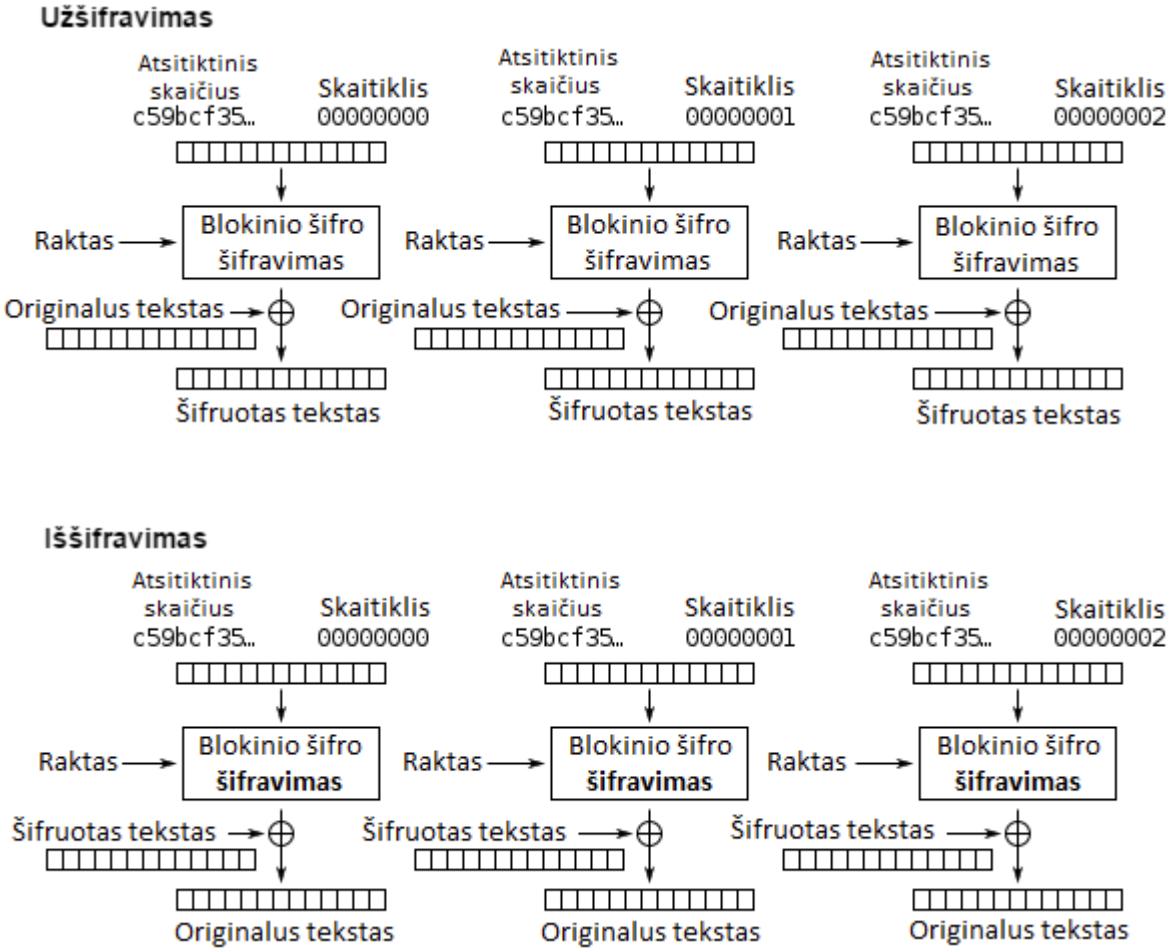
Grįžtamojo ryšio išvesties režimas panašus į *CFB*, tačiau skiriasi tuo, kad raktų srautas sugeneruojamas nepriklausomai nuo šifrogramos. Tai reiškia, kad *OFB* režimu neperduodamos klaidos. Be to *OFB* režimas pasižymi didesne sparta nei *CFB*, nes raktų srautas gali būti sugeneruotas nepriklausomai nuo tekstogramos ar šifrogramos. Tai suteikia galimybę raktų srautą apskaičiuoti iš anksto tuo būdu ženkliai padidinant užšifravimo spartą, kurią gali riboti nebent ryšio kanalo pralaidumas. (žr. 6 pav.) [3]



6 pav. Grįžtamojo ryšio išvesties šifravimas ir iššifravimas

Skaitiklio režimas, kaip ir *CFB* bei *OFB*, naudoja blokinį šifrą tam, kad sugeneruotų pseudoatsitiktinių bitų seką. Gauti bitai pridedami prie tekstogramos bitų panaudojant loginę operaciją

XOR ir taip gaunami šifrogramos bitai, ir atvirškčiai, gautus bitus pridedami prie šifrogramos bitų gaunama pradinė tekstograma. (žr. 7 pav.) [3]



7 pav. Skaitiklio šifravimas ir iššifravimas

Tam, kad skirtingų tekstogramų šifravimas tuo pačiu raktu k vyktų skirtingai, paprastai prie užšifruojamo bloko numerio pridedamas fiksuotas atsitiktinis skaičius, kuris generuojamas panaudojant pseudoatsitiktinių skaičių generatorių (*PRNG*). [4]

Skirtingi režimai skirtingai toleruoja atsiradusias klaidas duomenų blokuose. 1 lentelėje pateikiamas šių režimų sugadinamų duomenų kiekis, kai užšifravimo arba duomenų perdavimo metu susigadina vienas baitas. [5]

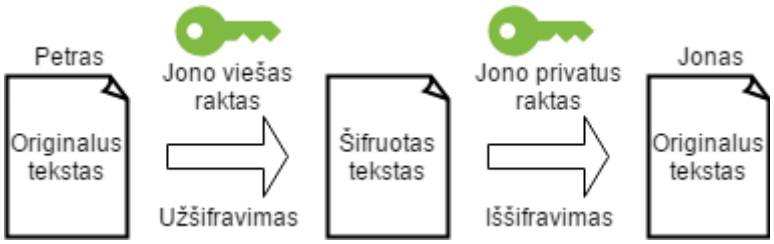
1 lentelė. Sugadintų duomenų kiekis atsiradus vieno bito klaidai duomenų bloke

Klaidos atsiradimo sritis	**Sugadintų duomenų kiekis, sugadinus vieną bitą**				
	ECB	**CBC**	**OFB**	**CFB**	**CTR**
Užšifravimas	Vienas blokas	Vienas blokas	Visi duomenys po klaidos atsiradimo	Vienas blokas	Vienas blokas
Duomenų perdavimas	Vienas blokas	Du blokai	Klaida neperduodama	Du blokai	Klaida neperduodama

Lentelėje matosi, kad vienintelis režimas, kuris perduoda klaidas į sekančius blokus yra grįžtamojo ryšio išvesties režimas. Visuose kituose susigadina tik tas blokas, kuriame įvyko klaida ir visi kiti blokai lieka nesugadinti.

1.5.2. Asimetrinis šifravimas

Asimetrinis šifravimas dar vadinamas viešojo rakto kriptografija, užšifravimui ir iššifravimui naudoja du skirtingus raktus: viešąjį ir privatųjį. Pavyzdys pateiktas 8 pav.



8 pav. Asimetrinis šifravimas

Ši kriptografijos forma eliminuoja simetrinės kriptografijos silpnybę kalbant apie užšifruotų duomenų dalinimąsi su kitais asmenimis. Simetrinėje kriptografijoje siuntėjas ir gavėjas naudoją tą patį raktą, kas sukelia saugumo problemas, nes siuntėjas turi kaip nors perduoti raktą gavėjui. Asimetrinio šifravimo atveju informacija yra užšifruojama naudojant gavėjo viešąjį raktą ir iššifruojama naudojant jo privatųjį raktą. Viešieji raktai nesuteikia galimybės iššifruoti duomenis, todėl gali būti viešai prieinami nesukeliant saugumo problemų. [6] [7]

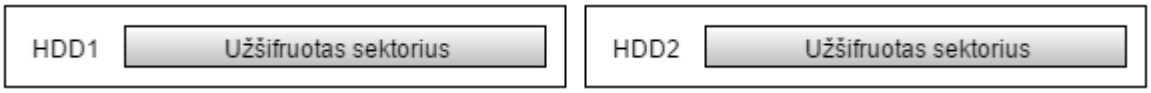
1.6. Failų šifravimas

Duomenų šifravimas naudojamas ne tik saugiam duomenų dalinimuisi, bet ir jų apsaugojimui įsilaužimo ar vagystės atveju. Galima išskirti du metodus, skirtus visų failų šifravimui:

a. disko lygio šifravimą;
b. failų sistemos lygio šifravimą;

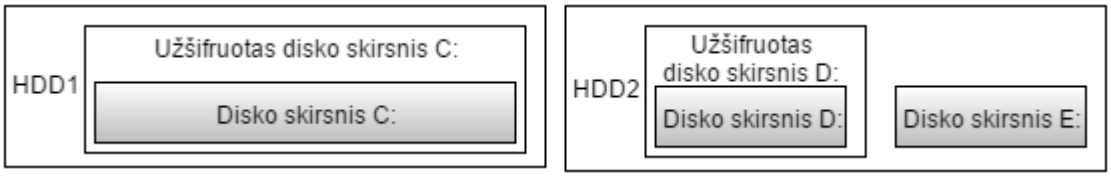
1.6.1. Disko lygio šifravimas

Disko lygio šifravimas (angl. *Full Disk Encryption*) yra viso disko arba skirsnio šifravimas. Viso disko šifravimo atveju programos užšifruoja visus kietojo disko sektorius, dažniausiai kiekvienas diskas užšifruojamas naudojant skirtingus raktus. (žr. 9 pav.) [8]



9 pav. Viso disko šifravimas

Skirsnio lygio šifravimas šifruoja atskiras disko skirsnius. (žr. 10 pav.)

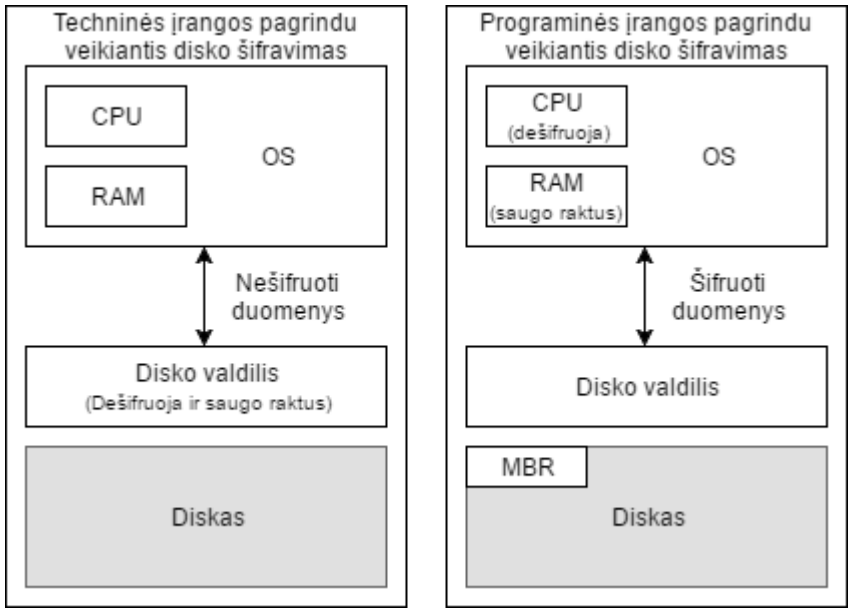


10 pav. Skirsnio lygio šifravimas

Disko lygio šifravimo metodai skirstomas į dvi kategorijas:

a. veikiančius techninės įrangos pagrindu (angl. *hardware-based*);
b. veikiančius programinės įrangos pagrindu (angl. *software-based*).

Kategorijų palyginimas pavaizduotas 11 pav. Techninės įrangos pagrindu veikiančiose šifravimo sistemose visą darbą atlieka disko valdiklis. Jis užšifruoja ir iššifruoja duomenis, bei saugo raktus. Operacinei sistemai apie šifravimą nėra žinoma, ji gauna ir perduoda kietajam diskui nešifruotus duomenis. Programinės įrangos pagrindu veikiančiose sistemose duomenų šifravimą ir iššifravimą atlieka kompiuterio procesorius, o raktai yra saugomi darbinėje atmintyje. Operacinė sistema iš disko gauna ir jam perduoda užšifruotus duomenis, o disko valdiklis šifravime nedalyvauja.



11 pav. Techninės ir programinės įrangos pagrindu veikiančių disko lygio šifravimų palyginimas

Techninės įrangos šifravimo atveju yra užšifruotas visas diskas, įskaitant ir pagrindinį užkrovimo sektorių (angl. *master boot record*). Programinės įrangos atveju šis sektorius lieka nešifruotas, kad operacinė sistema galėtų užsikrauti kompiuterio paleidimo metu.

Disko lygio šifravimas pirmiausia išpopuliarėjo su sistemomis veikiančiomis programinės įrangos pagrindu: *BitLocker* (*Windows*), *FileVault* (*Mac*), ir *dm-crypt* (*Linux/Android*). Tik išpopuliarėjus *SSD* kietiesiems diskams, juose buvo pradėta diegti techninės įrangos pagrindu veikiančios šifravimo sistemos. [9]

1.6.2. Failų sistemos lygio šifravimas

Failų sistemos lygio šifravimas dar vadinamas failų/aplankų šifravimu. Šio metodo principas yra atskirų failų ir aplankų šifravimas, atskirus failus šifruojant skirtingais raktais. Išskiriami du failų sistemos šifravimo tipai:

a. bendros paskirties (angl. *general-purpose*) failų sistema su šifravimu;
b. kriptografinės failų sistemos sluoksnis.

Bendrosios paskirties tipo šifravimas dažniausiai nešifruoja metaduomenų, tokių kaip direktorijų struktūros, failų pavadinimų, dydžio ar failo modifikavimo datos. Tai sukelia saugumo problemų, nes bet kas, kas turi priėjimą prie fizinio disko, gali matyti kokie dokumentai yra saugomi tam diske, tačiau jų turinio nemato. Kriptografinės failų sistemos yra suprojektuotos didelį dėmesį skiriant šifravimui ir saugumui. Tokios sistemos dažniausiai užšifruoja visą turimą informaciją, įskaitant ir metaduomenis.

1.7. Failų sinchronizavimas

Failų sinchronizavimas – tai procesas, kuris užtikrina, kad skirtingose vietose esantys failai ir atlikti pakeitimai yra atnaujinami pagal tam tikras taisykles. Yra du sinchronizavimo tipai:

a. vienos krypties;
b. dviejų krypčių.

Vienos krypties sinchronizavimo atveju failai yra kopijuojami iš šaltinio į vieną ar daugiau vietų, tačiau atlikti pakeitimai kitur nėra kopijuojami į šaltinį. Tokio tipo sinchronizavimas tinka atsarginių kopijų saugojimui. Dviejų krypčių sinchronizavimo atveju pakeitimai kopijuojami abiem kryptimis, siekiant abi vietas išlaikyti identiškas.

Duomenų saugojimas nutolusiuose serveriuose sukelia įvairias saugumo problemas. Vartotojas niekada negali būti tikras, kad pašaliniai asmenys negaus priėjimo prie jo failų. Taip pat siunčiant nešifruotus failus į serverį yra rizika, kad perkėlimo metu pašaliniai asmenys gali juos perimti. Siekiant užtikrinti duomenų saugumą būtina juos užšifruoti savo kompiuteryje ir tik tada juos talpinti serveryje. Tokiu būdu vartotojas užsitikrina, įvykus failų konfidencialumo pažeidimui, jų nebus galima peržiūrėti neturint reikiamų raktų skirtų iššifravimui.

Viena iš problemų yra didelių failų sinchronizavimas. Priklausomai nuo naudojamo protokolo, failo pakeitimas gali iššaukti viso failo siuntimą į serverį ir didelių failų keitimas sunaudoja

didelius kiekius duomenų srauto. Siekiant sumažinti sunaudojamą duomenų srautą, paprastai yra taikomi du algoritmai. Pirmasis paremtas kompiuteryje veikiančia programa, kuri suranda pasikeitusius bitus tarp modifikuoto failo ir to, kuris patalpintas serveryje, ir siunčia tik pakeistus bitus. Tam tikslui pasiekti programa suskaido failus į blokus ir naudoja maišos (angl. *hash*) funkcijas suskaičiuoti maišos reikšmėms. Tada šios reikšmės yra nusiunčiamos į serverį, kuris jas lygina su savo turimais failais. Supaprastintas algoritmo veikimas:

a. kliento kompiuteryje:
    1. išskaidyti failą f_c į b dydžio blokus $B_i = f_c[ib,(i + 1)b - 1]$;
    2. kiekvienam blokui B_i apskaičiuoti maišos reikmes $h_i = h(B_i)$ ir nusiųsti į serverį.
b. serveryje:
    1. su kiekviena gauta maišos reikšme B_i įrašyti įrašą (h_i, i) į žodyną naudojant h_i kaip raktą;
    2. pereiti per failo f_s blokus pradedant nuo pozicijos $j = 0$ ir atliekant veiksmus:
        i. apskaičiuoti bloko maišos reikšmę $h(f_s[j, j + b - 1])$ pradedant nuo j ir patikrinti žodyne, ar yra atitikmuo;
        ii. jei atitikmuo rastas – nusiųsti atitinkančio bloko indeksą i kliento kompiuteriui, padidinti j per b;
        iii. jei atitikmuo nerastas – nusiųsti simbolį $f_s[j]$ kliento kompiuteriui ir padidinti j per vieną.

Antrasis prieš siunčiant pakeitimus į serverį juos suspaudžia, taip sumažindamas jų apimtį. Pavyzdžiui, pakoregavus jau egzistuojantį failą prirašant 100 MB identiškų simbolių (pvz.: „a“), skirtumas tarp pakeisto ir debesyje esančio failo būtų 100 MB, bet suglaudinus šiuos pakeitimus sunaudotas duomenų srautas būtų apie 40 KB. Tai tik šiek tiek daugiau už reikalingą interneto srautą pridėjus tik vieną simbolį – apie 38 KB įskaitant metaduomenis. [10] [11]

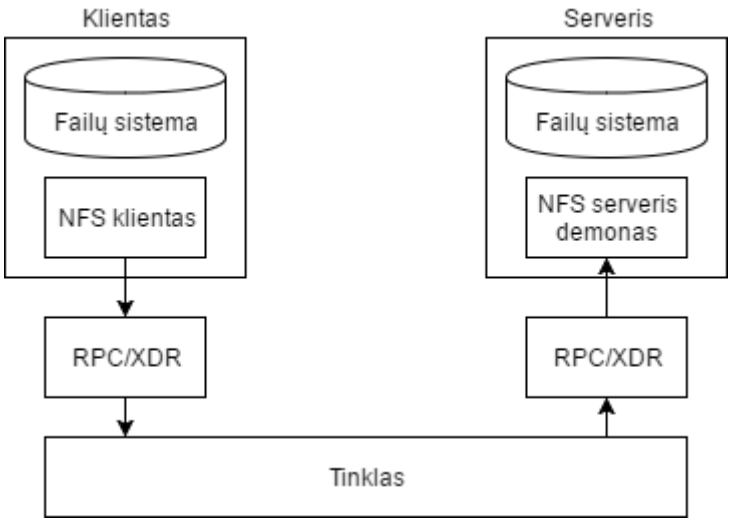
1.8. Tinklinės failų sistemos

Tinklinės failų sistemos – tai failų sistemos, kurios vartotojui suteikia galimybę savo kompiuteryje dirbti su failais, kurie fiziškai yra kitame įrenginyje, sujungtame per tinklą. Įprastai failai yra saugomi serveryje ir vienas ar keli vartotojai turi priėjimą prie jų iš skirtingų įrenginių. Pareikalavus tam tikro failo, jis yra parsiunčiamas į įrenginį ir jam pakitus – nauja versija nusiunčiama į serverį ir išsaugoma. Darbas su tokio tipo failų sistema nesiskiria nuo darbo su vietine failų sistema.

Populiariausios tokio tipo sistemos yra *NFS* (angl. *Network File System*) veikianti *UNIX* šeimos operacinėse sistemose ir *CIFS/SMB* (angl. *Server Message Block*) veikianti *Windows* operacinėse sistemose. Abi sistemos turi tik elementarias komandas skirtas manipuliuoti failais: failo atidarymas, ištrynimas, nusiuntimas. Sudėtingesnės operacijos, tokios kaip duomenų įterpimas nėra

palaikomos. Tai reiškia, kad atlikus minimalų failo pakeitimą, jis visas yra siunčiamas iš vartotojo kompiuterio į serverį. [12] [13]

NFS sukurta siekiant supaprastinti failų dalijimąsi per nehomogeninių įrenginių tinklą. Pagrindinis tikslas - pasiūlyti sprendimą programoms pasiekti nutolusius failus jų nemodifikuojant. Taip pat siekiamas nepriklausomumas nuo įrenginio ir operacinės sistemos, priimtinas veikimo greitis, atsistatymas po kritinių klaidų. Sistema susideda iš trijų pagrindinių procesų: protokolo, serverio pusės ir kliento pusės (12 pav.).



12 pav. Network file system

Serveryje veikia *NFS* demonas (angl. *daemon*), tam, kad failai visada būtų prieinami klientams. Administratorius nusprendžia, kuriuos failus ir aplankus padaryti pasiekiamus. Kliento įrenginyje esanti programa kreipiasi į serveryje veikiantį demoną, kai norima gauti failus. [14] [15]

1.9. Esamų problemos sprendimo metodų analizė

Rinkoje egzistuoja įvairių sprendimų, skirtų failų šifravimui. Siekiant užšifruoti failus ir juos sinchronizuoti su serveriu yra naudojama kelių programų kombinacija. Vieni populiariausių failo šifravimo produktų yra *TrueCrypt, EncFS, eCryptfs*. *TrueCrypt* veikia skirsnio šifravimo principu, tuo tarpu *EncFS* ir *eCryptfs* – failų sistemos šifravimo. Norint užšifruotus failus laikyti serveryje, prie kompiuterio yra primontuojamas nutolusio serverio katalogas, kuriame ir bus laikomi duomenys. Šiam tikslui naudojama tinklinė failų sistemą. Kompiuteryje yra sukuriamas naujas katalogas, kuriame vartotojas matys neužšifruotus failus ir pasinaudojus anksčiau minėtomis programomis šie du katalogai yra sujungiami.

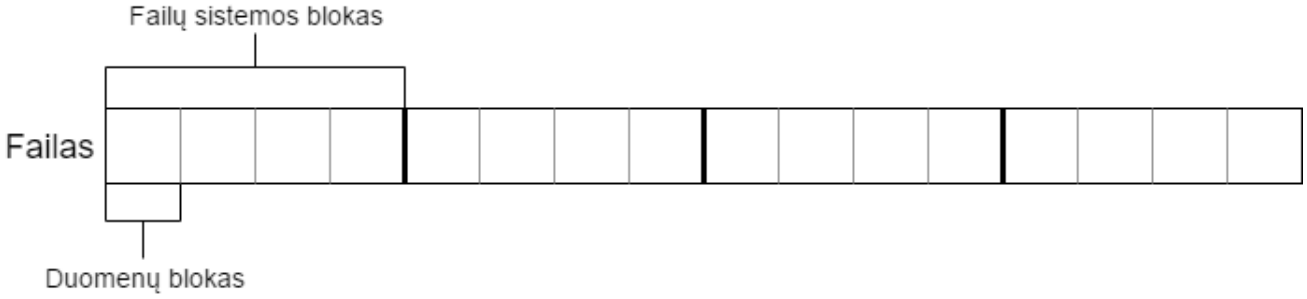
1.9.1. TrueCrypt ir VeraCrypt

TrueCrypt ir *VeraCrypt* yra nemokamos programos, skirtos failų šifravimui naudojant virtualius diskus. *TrueCrypt* kūrėjai 2014 metų gegužės 28 dieną internetiniame puslapyje paskelbė, kad sustabdo projekto palaikymą ir rekomenduoja vartotojams pradėti naudoti alternatyvius sprendimus. Nors tolimesnis šio produkto plėtojimas sustabdytas, atliktas nepriklausomas auditas paskelbė, kad jokių reikšmingų trūkumų nerasta. [16] *VeraCrypt* sukurta panaudojant *TrueCrypt* išeitis kodą ir yra viena iš atviro kodo alternatyvų. Nuo pirmosios versijos išleidimo ištaisyta nemažai saugumo spragų ir pagerintas programos saugumas. [17]

Šios programos veikia skirsnio šifravimo principu – yra sukuriamas pasirinkto dydžio virtualaus skirsnio failas, kuriame yra saugomi visi užšifruoti failai. Viena iš šio metodo problemų yra failo sinchronizavimas. Pasikeitus vienam failui (kad ir 1 baitas) yra vykdomas viso konteinerio sinchronizavimas ir tai gali sunaudoti didelį kiekį interneto srauto. Šiai problemai išspręsti kartais yra atliekamas failo bitų palyginimas ir sinchronizuojama tik ta dalis, kuri pasikeitė, tačiau dėl šifravimo, net ir menkiausias pakeitimas gali kardinaliai pakeisti užšifruotą konteinerį.

1.9.2. EncFs

EncFS yra vartotojo erdvės šifruota failų sistema. Yra naudojami du katalogai failų sistemai prijungti: šaltinio katalogas ir prijungimo taškas. Kiekvienas failas prijungimo taške turi jam atitinkantį failą šaltinio kataloge. [18] Failai yra šifruojami naudojant raktą, kuris laikomas užšifruotas šaltinio direktorijoje ir naudoja *AES* arba *Blowfish* šifravimo algoritmą, kurį vartotojas pasirenka failų sistemos kūrimo metu. 13 pav. pavaizduota, kaip failas yra skaidomas šifravimo metu. Pirmiausia yra suskaidomas į didesnius blokus, vadinamus failų sistemos blokais. Tada šie blokai yra suskaidomi į smulkesnius blokus, vadinamus duomenų arba šifro blokais. Jų dydis priklauso nuo pasirinkto algoritmo: *AES* atveju 16 baitų, *Blowfish* – 8 baitai. Failų sistemos bloko dydį gali pasirinkti vartotojas sistemos kūrimo metu. Galima rinktis nuo 64 iki 4096 baitų ir dydis privalo būti duomenų bloko dydžio kartotinis. Tada kiekvieno failų sistemos bloko duomenų blokai šifruojami atskirai naudojant *CBC* režimą.



13 pav. EncFs failų skaidymas

Paskutinis failo blokas yra šifruojamas skirtingai nuo visų kitų blokų. Kadangi dažniausiai šis blokas nėra pilnas, norint jį šifruoti tokiu pat metodu, kaip visus kitus, reiktų prie jo pridėti papildomų

baitų, dėl ko padidėtų pačio failo dydis. Siekiant to išvengti, paskutinis blokas yra kelis kartus šifruojamas naudojant *CFB* režimą.

2014 metų sausio 14 dieną atliktas saugumo auditas atskleidė šio įrankio saugumo spragas [19]:

1. *EncFS* naudoja vieną raktą duomenų šifravimui ir *MAC* skaičiavimui. Tai yra laikoma bloga praktika ir patartina naudoti skirtingus raktus. Šios spragos išnaudojimas ir poveikis saugumui maži;
2. srauto šifras naudojamas užšifruoti paskutiniam failo blokui. Tai turi didelę įtaką saugumui;
3. bloko *IV* generuojamas iš bloko numerio naudojant *XOR*. Tai taip pat turi įtaką saugumui;
4. nėra atpažįstamos failų skylės. Failų skylės – tai dideli failai, kurių dalį sudaro nuliniai baitai ir kurie nėra saugomi diske. Jei visas blokas yra nuliai, tokie blokai nėra iššifruojami ir nėra tikrinamas jų *MAC*. Šios spragos išnaudojimo galimybės yra gana didelės, nors poveikis saugumui – mažas;
5. naudojami 64 bitų *MAC*. Jie yra per trumpi ir tai turi didelį poveikį saugumo;
6. redaguojant konfigūracijos failą galima atjungti *MAC*.

Nors *EncFS* yra naudingas įrankis, jis ignoruoja daug gerųjų kriptografijos praktikų. Viena to priežasčių yra įrankio amžius. Nepaisant to, jis iki šiol yra dažnai naudojamas. [19]

1.9.3. eCryptfs

eCryptfs (angl. ***E**nterprise **Crypt**ographic **F**iles**y**stem)* yra disko šifravimo programos paketas skirtas Linux operacinėms sistemoms. Ji taip pat veikia failų sistemos šifravimo principu, nuo *EncFS* skiriasi tuo, kad veikia kaip branduolio modulis. *eCryptfs* visus kriptografinius metaduomenis (naudojamą algoritmą, raktą ir t. t.) saugo kiekvieno failo antraštėje, todėl failus galima lengvai perkelti į kitus įrenginius. [20]

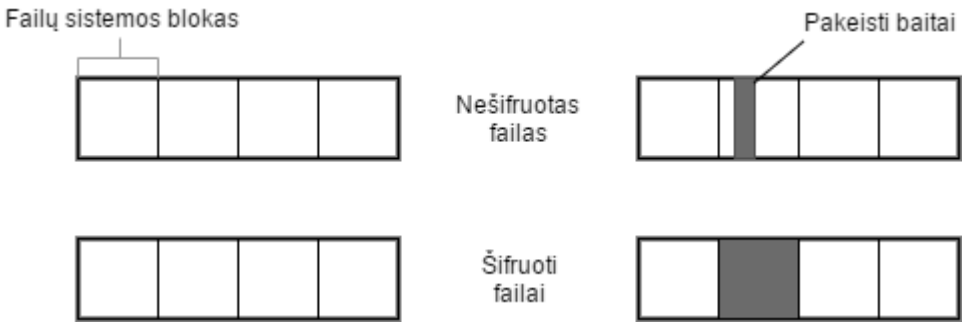
2014 metų sausio 22 dieną atliktas saugumo auditas atskleidė kelias šio įrankio saugumo spragas [21]:

a. *eCryptfs-rewrap-passphrase* komanda nesugeneruoja naujos druskinimo reikšmės (angl. *salt*) kai pakeičiamas slaptažodis. Šios spragos išnaudojimas ir poveikis saugumui mažas.
b. šakninis IV yra MD5 maišos funkcijos rezultatas, naudojant sesijos šifravimo raktą. Šios spragos išnaudojimas ir poveikis saugumui mažas.
c. failo vardų šifravimas nenaudoja *IV*. Šios spragos išnaudojimas ir poveikis saugumui mažas.

Lyginant su *EncFS*, *eCryptfs* turi kur kas mažiau saugumo spragų, tačiau yra aspektų, kuriuos galima pagerinti suteikiant didesnį saugumą. [21]

1.10. EncFs ir eCryptfs failų šifravimo eksperimentinis tyrimas

Analizuojant esamus sprendimus atliktas eksperimentinis tyrimas, kuriuo siekiama nustatyti, kaip pasikeičia šifruota failas, kai nešifruotame faile yra pakeičiami, pridedami ir ištrinami keli baitai. Eksperimentas atliktas naudojant 4 KB dydžio failą. Pirmiausia tyrimas atliekamas su *EncFs* programa.. Faile pakeitus kelis baitus ir palyginus pirminio ir pakeisto failų užšifruotas versijas pastebėta, kad skiriasi tik failų sistemos blokas, kuriame buvo atliktas pakeitimas. Prieš ir po jo einantys blokai liko nepakitę. Gauti rezultatai pavaizduoti 14 pav., pilka spalva pažymėti skirtumai.



14 pav. EncFS šifravimas pakeitus kelis baitus

To priežastis yra ta, kad failų sistemos blokai šifruojami atskirai, nepriklausomai vienas nuo kito. Tik kiekvieno jų turinys yra šifruojamas *CBC* režimu, dėl ko ir pasikeičia tik tas blokas, kuriame atliktas pakeitimas.

Faile įterpus kelis papildomus baitus, rezultatai kitokie. Šiuo atveju užšifruotuose failuose skirtumas atsiranda nuo to bloko, kuriame atsirado nauji baitai ir tęsiasi iki pat failo pabaigos (žr. 15 pav.). Skiriasi ne tik blokas su naujais baitais, bet ir visi einantys po jo.

15 pav. EncFS šifravimas pridėjus kelis baitus

Taip yra todėl, kad naujai pridėti baitai pastumia kitus baitus per tiek pozicijų, kiek buvo pridėta baitų taip pakeisdami ne tik to vieno bloko turinį, bet ir visų kitų, kurie eina po jo. Ištrynus

kelis baitus rezultatai panašūs – nuo to bloko, kuriame buvo atliktas pakeitimas failai skiriasi iki pat pabaigos (žr. 16 pav.).

16 pav. EncFS šifravimas ištrynus kelis baitus

Analogiški eksperimentai atlikti ir su *eCrytpFs* programa. Visais atvejais pakeitus nešifruotą failą, šifruotas failas pasikeitė visiškai. Nepriklausomai, ar buvo pakeisti, prididėti ar ištrinti baitai, failas pakito nuo pat pradžios iki pabaigos.

Iš eksperimento rezultatų matosi, kad šifruojant *EncFs* programa kyla problema kai faile yra pridedami arba ištrinami baitai. Priklausomai nuo vietos, kurioje atliktas pakeitimas, skiriasi, kiek pasikeičia failas. Dėl kelių baitų pridėjimo ar ištrynimo gali visiškai pasikeisti visas šifruotas failas, jei pakeitimas atliktas failo pradžioje. *eCryptfs* rezultatai blogesni – nepriklausomai nuo atlikto pakeitimo, pasikeičia visas šifruotas failas.

1.11. Analizės išvados

Egzistuoja keletas skirtingų failų šifravimo metodų, kurių kiekvienas turi savų stiprių ir silpnų vietų – vieni užtikrina didesnį saugumą, kiti orientuoti į veikimo spartą. Todėl jų pasirinkimas priklauso nuo atliekamos užduoties. Failų sinchronizavimui taip pat egzistuoja įvairių metodų, kurių pasirinkimas priklauso nuo norimos atlikti užduoties.

Atlikus skirtingų failų šifravimo metodų analizę, nuspręsta darbe naudoti atskirų failų šifravimą dėl šio metodo privalumo – pakeitus vieną failą reikia nusiųsti daug mažiau duomenų į nutolusį serverį. Disko lygio šifravimo atveju po failo pakeitimo reikia perduoti visą virtualų diską.

Atlikus rinkoje egzistuojančių failų šifravimo sprendimų analizę pastebėta, kad jie turi įvairių saugumo spragų, dėl kurių atsiranda duomenų konfidencialumo pažeidimo rizika. *EncFs* naudojamas 64 bitų *MAC* yra per trumpas, paskutinis failo blokas šifruojamas naudojant srautinį šifrą vietoj blokinio, taip pat failo skylės nėra šifruojamos. *eCryptfs* nenaudoja *IV* failų vardų šifravimui, šakninis *IV* yra *MD5* maišos funkcijos rezultatas naudojant sesijos šifravimo raktą. Šios spragos pagrinde atsirado dėl šių įrankių amžiaus. Nuo to laiko, kai jie buvo sukurti, kompiuterių skaičiavimo pajėgumai smarkiai išaugo ir tai, kas tuo metu užtikrino pakankamą saugumą, dabar gali būti lengvai nulaužiama.

Išanalizavus *EncFs* šifravimo metodo veikimą nustatyta, kad pridėjus arba ištrynus kelis baitus faile, pasikeičia šifruotas failas nuo tos vietos, kur atliktas pakeitimas. Jei pakeista failo pradžia – pakinta visas šifruotas failas. Išanalizavus *eCryptfs* metodo veikimą nustatyta, kad atlikus bet kokį pakeitimą keičiasi visas šifruotas failas. Dėl tokio abiejų metodų veikimo gali smarkiai išaugti perduodamas duomenų kiekis, kai šifruoti failai saugomi serveryje.

Išanalizavus tinklines failų sistemas *CIFS/SMB* ir *NFS* nustatyta, kad šios sistemos turi tik elementarias komandas skirtas darbui su failais ir dalies failo atnaujinimas nėra palaikomas. Dėl šios priežasties po bet kokio failo pakitimo į serverį yra perduodamas visas failas.

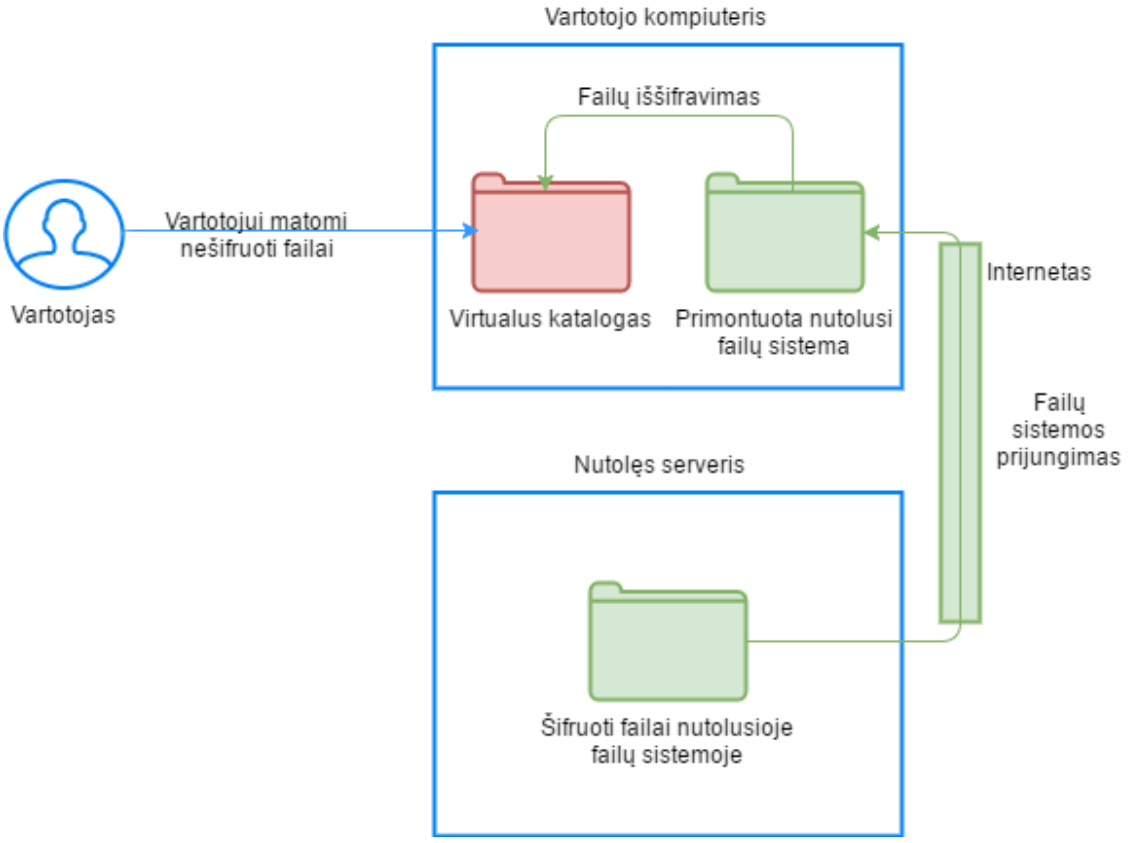
Apžvelgus analizės rezultatus nuspręsta pasiūlyti naują šifravimo metodą, kuris išspręstų pagrindines problemas rastas analizės metu: failų šifravimas po baitų pridėjimo ar ištrynimo, didelis duomenų srautas į serverį po failo redagavimo.

2. FAILŲ ŠIFRAVIMO METODAS

Šiame skyrelyje aprašytas pasiūlytas naujas failų šifravimo metodas sprendžiantis analizės metu nustatytas problemas. Naudojamas atskirų failų šifravimas, nes šiuo atveju pakeitus vieną failą yra perduodama kur kas mažiau duomenų į nutolusį serverį, lyginant su disko lygio šifravimu. Šifravimo metu failai išskaidomi į mažesnius failus, kurie yra užšifruojami ir išsaugomi serveryje. Kadangi analizės metu nustatyta, kad tinklinės failų sistemos nepalaiko duomenų įterpimo į failą, tokiu būdu sprendžiama ši problema. Į nutolusį serverį perduodami tik tie maži failai, kurie pasikeitė. Šių, mažesnių failų, dydis nėra fiksuotas, todėl nešifruotame faile ištrynus kelis baitus, pasikeičia tik tas šifruotas failas, kuriame yra pakitusi nešifruoto failo dalis. Įterpus į nešifruotą failą naujų baitų yra sukuriamas naujas šifruotas failas ir visi sekantys failai lieka nepakitę. Tokiu būdu yra sumažinamas perduodamų duomenų kiekis į serverį pakitus failui.

2.1. Metodo veikimo diagrama

17 pav. pateikta siūlomo metodo veikimo schema, vaizduojanti, kaip veikia sprendimas. Žalia spalva pažymėtos vietos, kur failai yra apsaugoti, raudonai – neapsaugoti.



17 pav. Metodo veikimo diagrama

Pirmiausia prie kompiuterio yra primontuojama nutolusio serverio failų sistemą. Tokiu būdu vartotojas serveryje esančią failų sistemą gali peržiūrėti savo kompiuteryje, tam tikrame aplanke, prie

kurio yra primontuojama failų sistema. Tai yra atliekama tam, kad šio metodo naudojimas neapsunkintų vartotojui darbo su jo failais. Iš vartotojo perspektyvos, ši primontuota failų sistema niekuo nesiskiria nuo paprasto aplankalo su jame esančiais failais. Serveryje bus laikomi tik šifruoti failai, su užšifruotu turiniu, failo pavadinimu, kai kuriais metaduomenimis. Taip pat visi užšifruoti failai yra laikomi šakniniame kataloge. Net jei ir nešifruotas failas yra kokiame nors pakatalogyje, šifruojant jie yra išsaugomi šakniniame kataloge. Informacija, apie aplankų medį ir juose esančius failus yra saugoma atskirame *XML* faile, kuris taip pat yra užšifruotas ir saugomas tame pačiame šakniniame kataloge. Šifruojant failai yra skaidomi į mažesnius failus su atsitiktinai sugeneruotu pavadinimu. Toks metodas pasirinktas dėl to, kad ir nematant failų pavadinimų ar jų turinio, galima daryti tam tikras išvadas iš failų ir aplankų struktūros. Matant failų sistemos medį ir kiek failų yra kiekviename aplanke, galima daryti tam tikras išvadas, kas yra saugoma toje failų sistemoje. Siekiant nuo to apsisaugoti, nuspręsta slėpti failų sistemos struktūrą.

Vartotojas visus failus, kurie yra nutolusioje failų sistemoje, mato specialiame virtualiame kataloge. Šiame kataloge vartotojui rodomos nešifruotos failų versijos, taip pat atkurta visa katalogų struktūra. Iš vartotojo perspektyvos šis katalogas nesiskiria niekuo nuo įprastinių katalogų. Visos operacijos yra atliekamos taip pat, kaip ir su įprastais failais bei katalogas. Vartotojas gali juos redaguoti, trinti, kurti naujus. Bet šiame kataloge nėra saugomi jokie failai. Visa aplanko struktūra yra atkuriama iš informacijos, esančios užšifruotame kataloge. Tik tada, kai vartotojas atidaro tam tikrą failą, jo visa informacija yra iššifruojama ir nešifruotas failas yra saugomas darbinėje kompiuterio atmintyje. Atlikus failo pakeitimus ir jį išsaugojus, failas yra užšifruojamas ir išsaugomas serveryje. Išjungus programą, aplankas, kuriame buvo atvaizduojami nešifruoti failai, lieka tuščias ir jį atidarius, vartotojas nebemato jokių prieš tai buvusių failų.

Dokumento atidarymo ir saugojimo metu failų turinys yra siunčiamas iš serverio į kompiuterį, ir atvirkščiai, naudojant tinklinę failų sistemą. Siūlomas metodas gali veikti ir nešifruotu kanalu, duomenų saugumas persiuntimo (angl. *in transit*) metu yra užtikrinamas juos užšifruojant prieš siuntimą. Tinklu perduodami duomenys yra užšifruoti, todėl jie gali būti siunčiami bet kokiais, net ir nesaugiais, kanalais. Pašaliniai asmenys siuntimo metu perėmę šiuos failus negalės jų peržiūrėti, nes neturės rakto, reikalingo failų iššifravimui. Kadangi failai yra užšifruojami vartotojo kompiuteryje prieš siuntimą ir iššifruojami kompiuteryje po failo atsisiuntimo, yra užtikrinamas failų saugumas saugojimo (angl. *at rest*) metu. Serveryje papildomai nereikia diegti jokios programinės failų šifravimo įrangos. Įsilaužėliai ar kiti pašaliniai asmenys serveryje mato tik šifruotus failus.

2.2. Raktų generavimas

Raktų generavimui naudojama *PBKDF2* (*Password-Based Key Derivation Function 2*) funkcija. [22] [23] Daug kartų panaudojant pseudoatsitiktines funkcijas ant įvesties slaptažodžio ir atsitiktinės reikšmės (angl. *salt*) yra sugeneruojamas slaptažodis, kuris naudojamas tolesniems metodo veiksmams atlikti. Funkcija turi penkis įvesties parametrus:

$$DK = PBKDF2(PRF, Password, Salt, c, dkLen)$$

DK – *PBKDF2* funkcijos sugeneruotas raktas.

PRF – pseudoatsitiktinė funkcija

Password – pirminis slaptažodis, iš kurio yra gaunamas *DK*

Salt – atsitiktinių bitų seka

C – iteracijų skaičius

dkLen – norimas gauto rakto ilgis

$$DK = PBKDF2(HMAC{-}SHA1, passphrase, ssid, 4096, 256)$$

Pagrindinis raktas nėra saugomas faile. Kiekvieną kartą prijungiant nutolusią failų sistemą prie kompiuterio raktas yra skaičiuojamas iš naujo. Sistema su 1.6 Ghz 64 bitų procesoriumi per pusę sekundės atlieka apie 64 tūkstančius iteracijų, tad šis procesas neįtakos metodo greitaveikos.

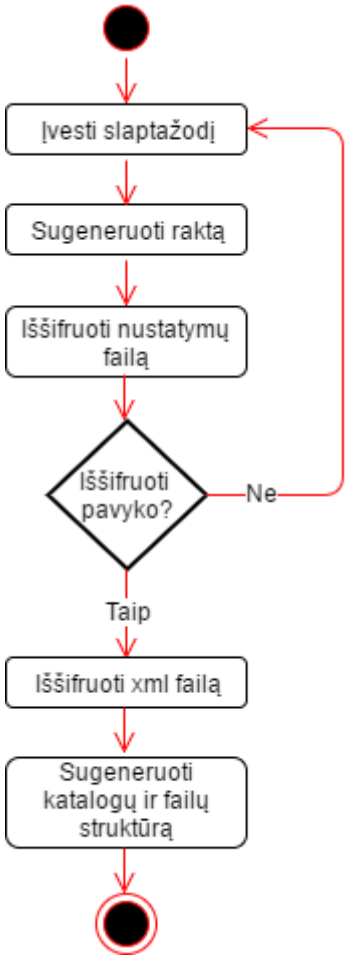
2.3. Rakto saugojimas

Vartotojo turimas slaptažodis naudojamas užšifruoti tik nustatymų failui. Šiame faile saugoma informacija apie šifravimui naudojamą algoritmą, režimą, rakto ilgį, užšifruoto failo dydį. Taip pat saugomas tikrasis raktas, naudojamas failų šifravimui. Primontavus failų sistemą, pirmiausia yra iššifruojamas šis failas ir toliau visi kiti failai iššifruojami naudojant šiuos nustatymus ir raktą. Toks metodas pasirinktas tam, kad vartotojui pasikeitus slaptažodį, nereikėtų iš naujo peršifruoti visų failų, taip sugeneruojant didelį duomenų srautą į serverį. Pasikeitus slaptažodžiui yra peršifruojamas tik nustatymų failas.

Nustatymų failas taip pat naudojamas patikrinti, ar vartotojas įvedė teisingą slaptažodį. Prieš failo užšifravimą yra apskaičiuojama jo santrauka ir po užšifravimo įrašoma į failo antraštę. Iššifravus failą su vartotojo nurodytu slaptažodžiu vėl yra apskaičiuojama santrauka ir palyginama su ta, kuri išsaugota antraštėje. Jų sutapimas reiškia, kad nurodytas tas slaptažodis yra teisingas. Santraukos skaičiavimui naudojama *SHA-256* funkcija. Ji pasirinkta, nes yra viena iš Nacionalinio Standartų ir Technologijos instituto (angl. *NIST*) rekomenduojamų funkcijų. [24] [25]

2.4. Failų sistemos primontavimo algoritmas

Primontavus failų sistemą pirmas žingsnis yra suformuoti nešifruotų failų ir katalogų struktūrą. Šis algoritmas pateikiamas 18 pav. Pirmiausia vartotojo yra paprašoma įvesti slaptažodį, iš kurio pasinaudojus *PBKDF2* funkcija sugeneruojamas šifravimo raktas. Kadangi vartotojo pasirinkti šifravimo nustatymai yra saugomi užšifruotame nustatymų faile, šis failas yra užšifruojamas naudojant gautą raktą iš vartotojo slaptažodžio ir numatytuosius parametrus, kurių vartotojas pasirinkti negali. Sugeneravus raktą, bandoma iššifruoti nustatymų failą. Jei buvo įvestas klaidingas slaptažodis, failo iššifruoti nepavyks ir vartotojo paprašoma iš naujo nurodyti slaptažodį. Jei nurodytas sėkmingas slaptažodis, nustatymų failas yra iššifruojamas ir su jame esančiu šifravimo raktu atliekami tolimesni failų iššifravimo ir užšifravimo veiksmai.



18 pav. Failų sistemos primontavimo algoritmas

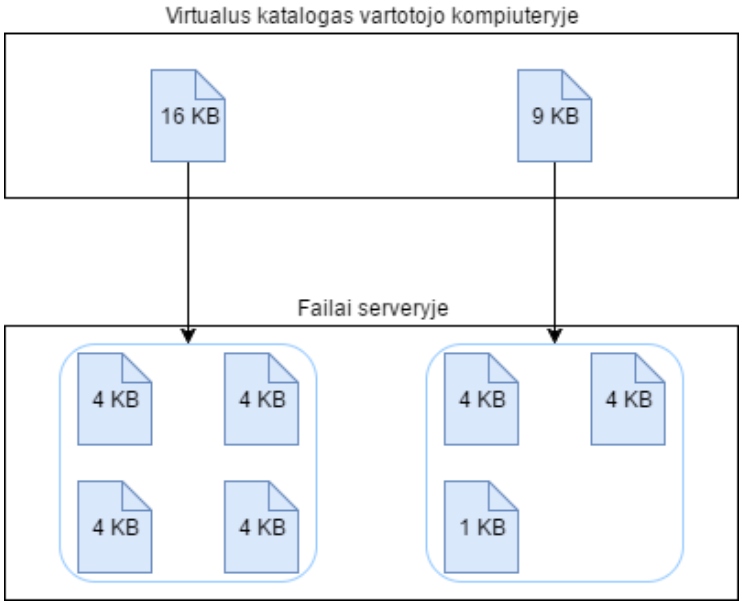
Visa failų ir katalogų struktūra yra saugoma specialiame *XML* faile, kuriame aprašomas aplankų medis, aplanke esančių failų pavadinimai, failų metaduomenys ir užšifruotų blokų failai, kurie sudaro tą failą (21 pav.). Sėkmingai iššifravus nustatymų failą, programa iššifruoja failų struktūrą aprašantį *XML* failą ir tada suformuoja struktūrą vartotojui.

2.5. Failų saugojimas

Failų saugojimui naudojami du skirtingi algoritmai, kurių pasirinkimas priklauso nuo to, ar šifruojamas naujas failas, ar tai pakeistas jau egzistuojantis failas. Pirmuoju atveju failas yra paprasčiausiai užšifruojamas ir nusiunčiamas į serverį, antru atveju atliekamas failo palyginimas, siekiant nustatyti kurios failo dalys keitėsi ir atnaujinti tik jas.

2.5.1. Naujo failo saugojimas

Naujo failo šifravimas, kai jis neturi atitikmens serveryje, atliekamas naudojant *S* dydžio blokais, kurie vadinami failų sistemos blokais. *S* turi būti 16 kartotinis, kadangi naudojamas *AES* šifravimo metodas naudoja 16 baitų dydžio blokus. Pirmiausia failas suskaidomas į blokus, tada šie blokai yra užšifruojami naudojant *AES* šifravimo metodą veikiantį *CBC* režimu ir gautos šifrogramos yra rašomos į atskirus failus. Kiekvienas užšifruotas *S* dydžio blokas yra išsaugomas kaip naujas failas (19 pav.). Failas yra suskaidomas dėl kelių priežasčių: pirmiausia, taip bus paslėptas konkretaus failo dydis, be to, tai padės sutaupyti duomenų srautą, kai failas bus redaguojamas. Atlikus smulkius failo pakeitimus, į serverį reikės nusiųsti tik tuos blokus, kurie keitėsi. Bloko dydis *S* nustatomas pagal serveryje esančios failų sistemos bloko dydžio, kuris dažniausiai lygus 4 KB. Kadangi tai yra mažiausias duomenų kiekis, kurį failas gali užimti diske, pasirinkus būtent tokio dydžio blokus yra optimaliai išnaudojama disko vieta. Parinkus mažesnę *S* reikšmę, pavyzdžiui, 1 KB, failas vis tiek užimtų 4 KB disko vietos. Atsižvelgiant į šifruojamo failo dydį, *S* reikšmė gali kisti – kuo failas didesnis, tuo didesnė reikšmė. Tačiau ji turi būti failų sistemos bloko kartotiniu.

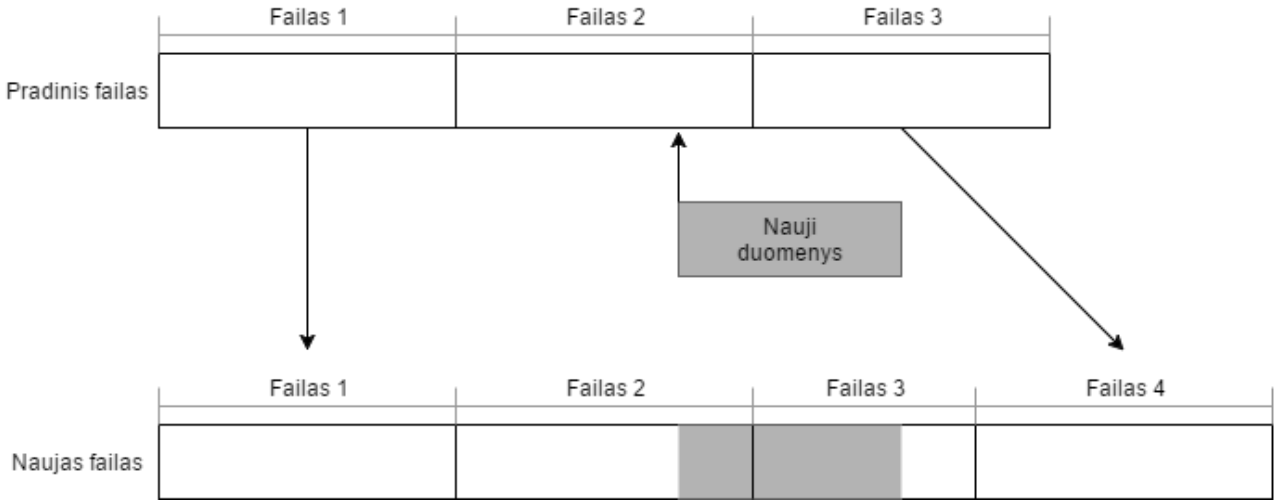


19 pav. Failų išskaidymas

Failo skaidymo metu visa informacija, nurodanti, kurie blokai sudaro tą failą ir jų eilės tvarką, yra surašoma į tą patį *XML* failą, kuris aprašo failų ir aplankų struktūrą (pavyzdys pateiktas 21 pav.).

2.5.2. Redaguoto failo saugojimas

Pareikalavus kokio nors failo ar jo dalies iš virtualaus aplanko yra nuskaitoma *XML* failo informacija iš kokių šifruotų failų jis susideda, bei kokia eilės tvarka. Toliau nustatoma, kurie šifruoti failai sudaro tą dalį, kurią norima peržiūrėti, ir nustačius – parsiunčiami iš serverio, iššifruojami bei perduodami programai, kuri jų pareikalavo. Baigus darbą ir išsaugojus redaguojamą failą, jis yra vėl šifruojamas, tačiau, skirtingai nei naujo failo saugojimo metu, yra atliekamas naujo ir seno bloko palyginimas. Nustačius, kad blokas nepakito, jis nėra iš naujo saugomas, taip sutaupant duomenų srauto. Jei bloko viduje pakito keletas baitų, tačiau nebuvo ištrintų, ar pridėtų baitų, jis yra užšifruojamas ir nusiunčiamas į serverį. Atsiradus naujiems baitams, yra surandama, kiek tiksliai atsirado naujos informacijos ir iš jų sudaromas vienas ar keli blokai (20 pav.). Paskutinio iš šių naujų blokų dydis turi būti 16 kartotinis, tačiau nebūtinai turi būti lygus *S*.



20 pav. Redaguoto failo šifravimas

Atsiradus naujiems blokams pasikeičia ir jų eilės tvarka, nes visi sekantys blokai yra pastumiami per tiek pozicijų, kiek naujų blokų buvo sukurta, todėl nauja eilės tvarka išsaugoma *XML* faile. Taip pat kuriant naują bloką, tikrinama prieš ir po jo esančių blokų dydis, siekiant nustatyti, ar jų ilgis lygus *S*, jei ne – nauja informacija sujungiama su jais.

2.6. Failų ir aplankų struktūros XML failas

Serveryje laikomas užšifruotas *XML* failas, kuriame yra aprašyta visa informacija reikalinga atkurti failų ir aplankų struktūrą. Jame saugoma informacija, apie kiekvieno aplanko turinį – kokie aplankai ir failai jame yra, taip pat surašyti failų metaduomenys. Kadangi šifravimo metu vienas failas yra išskaidomas į daug mažų failų, šiame *XML* dokumente taip pat yra saugoma informacija, kurie failai sudaro pradinį, ir kokia eilės tvarka jie išdėstyti. Pavyzdys pateiktas 21 pav.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<folder>
    <name>Aplankas_1</name>
    <content>
        <folder>
            <name>Aplankas_2</name>
            <content>
                <file>
                    <name>Failas_1</name>
                    <metadata>
                        <created>2016-10-10</created>
                        <modified>2016-10-11</modified>
                    </metadata>
                    <blocks>
                        <block id="1">ADJHGWI</block>
                        <block id="2">IAHDSKA</block>
                        <block id="3">BJSDKIO</block>
                        <block id="4">TNDKKAS</block>
                    </blocks>
                </file>
            </content>
        </folder>
        <file>
            <name>Failas_2</name>
            <metadata>
                <created>2016-11-01</created>
                <modified>2016-11-02</modified>
            </metadata>
            <blocks>
                <block id="1">KJSDJAS</block>
                <block id="2">YWIEWDJ</block>
                <block id="3">YWOEWDS</block>
                <block id="4">NBBSJIW</block>
                <block id="5">IUQWOEU</block>
                <block id="6">BHDSHDS</block>
            </blocks>
        </file>
    </content>
</folder>
```

21 pav. XML failo pavyzdys

Kadangi metaduomenys ir pavadinimai yra saugomi viename faile, o ne kiekvieno failo antraštėje, formuojant virtualų aplanką vartotojui, nereikia nuskaityti kiekvieno failo antraštės atskirai norint surinkti šią informaciją. Tam pakanka atšifruoti *XML* failą ir iš jame saugomos informacijos yra sudaroma failų ir katalogų struktūra.

3. FAILŲ ŠIFRAVIMO ĮRANKIO PROJEKTAS IR TYRIMAS

3.1. Įvadas

Remiantis šiame darbe sudarytu nauju failų šifravimo metodu realizuotas eksperimentinis šio metodo prototipas. Realizacijai pasirinkta *Java* programavimo kalba dėl to, kad ji veikia skirtingose operacinėse sistemose. Taip pat naudojama atviro kodo kriptografijos biblioteka *Bouncy Castle Crypto*. Kadangi esami sprendimai, kurie naudojami šiuose tyrimuose, veikia *Linux* pagrindu sukurtose operacinėse sistemose, visi tyrimai atliekami *Ubuntu 16.04.2* OS. Taip bus užtikrinti kuo tikslesni rezultatai. Naudojantis sukurtu prototipu šifravimo metodas vertinamas pokyčio aspektu, t. y. kiek pasikeičia užšifruotas failas po jo pakeitimo lyginant su pirmine versija. Taip pat atliekamas tyrimas su *Linux* branduolio išeities kodo failais, siekiant nustatyti koks kiekis duomenų perduodamas į serverį šifruojant pakeitimus tarp dviejų versijų.

Eksperimentas atliekamas naudojant šią aparatūrinę įranga:

a. procesorius: Intel(R) Core(TM) i5-4210U CPU @ 1.70 GHz;
b. darbinė atmintis: 16 GB;
c. operacinė sistema: Windows 10 Enterprise.

Prototipas šifravimui naudoja *AES* šifravimo algoritmą su 256 bitų ilgio raktais, naudojamas šifravimo režimas – *CBC*. Failai skaidomi į 4 KB dydžio failus. Failų metaduomenys, informacija, iš kokių failų susideda pirminis failas saugoma atskirame faile.

3.2. Realių failų analizė

Prieš atliekant eksperimentus, atlikta realių failų analizė siekiant išsiaiškinti, kokio dydžio failai dažniausiai būna saugomi failų sistemose ir kokie pakeitimai juose atliekami. Šiai analizei pasirinkti *Linux* branduolio 4.9 ir 4.10 versijų išeities kodai.

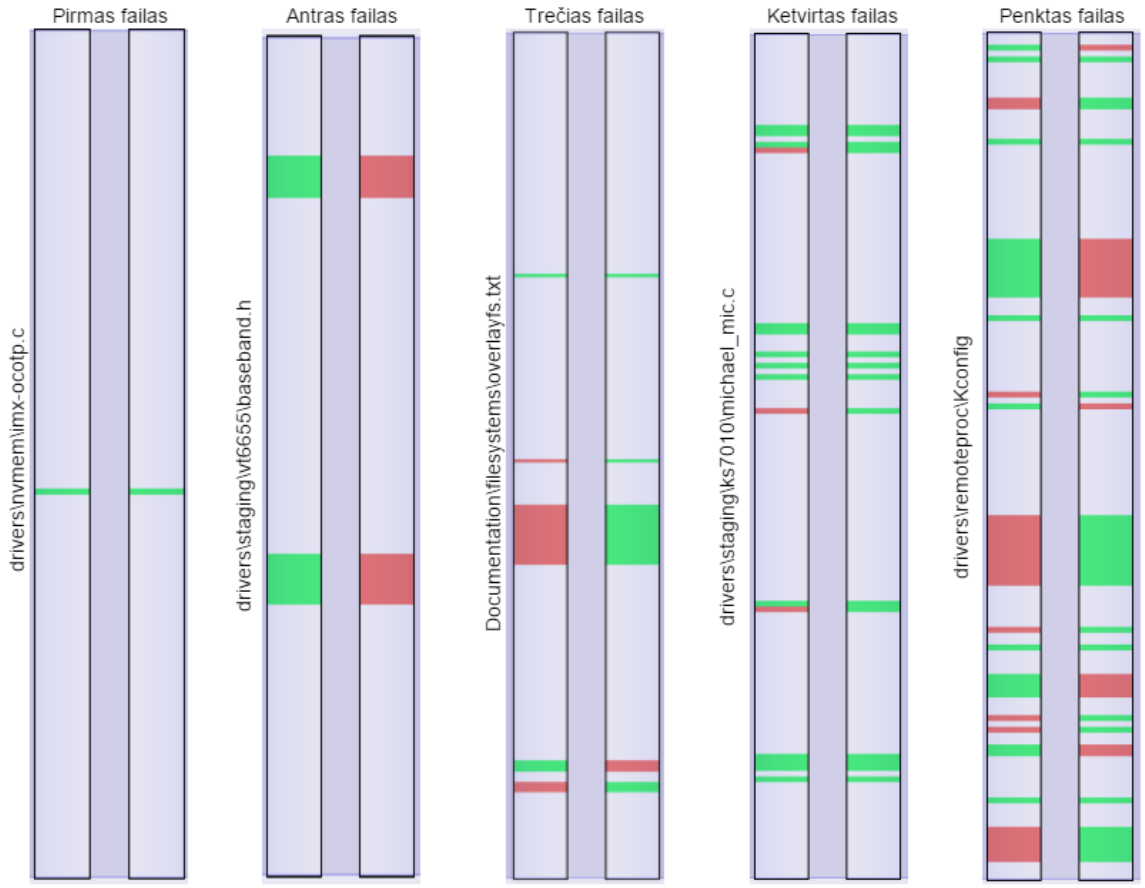
Pirmiausia atlikta Linux branduolio 4.9 versijos išeities kodo failų analizė siekiant nustatyti, kiek ir kokio dydžio yra failai. Lentelėse pavaizduoti tik tie, kurių turinys skiriasi lyginant su 4.10 branduolio versija. Iš viso tokių failų yra 10096, atrinkimas atliktas naudojant *WinMerge* programinę įrangą. Failai suskirstyti pagal dydį į intervalus ir rezultatai pateikti 2 lentelėje.

2 lentelė. Realių failų dydžių tyrimo rezultatai

Dydžio intervalas (KB)	Failų kiekis		Dydžio intervalas (KB)	Failų kiekis
0 – 1	513		40 – 50	425
1 – 2	615		50 – 60	289
2 – 3	578		60 – 70	208
3 – 4	528		70 – 80	142
4 – 5	477		80 – 90	104
5 – 6	452		90 – 100	92
6 – 7	381		100 – 200	286
7 – 8	368		200 – 300	47
8 – 9	343		300 – 400	12
9 – 10	335		400 – 500	6
10 – 20	1823		500 – 600	2
20 – 30	1049		600 – 1000	0
30 – 40	586		1000 – 2000	1

Iš lentelėje pateiktų rezultatų matosi, kad daugiausiai failų yra intervale nuo 10 iki 20 kilobaitų, šiek tiek mažiau – nuo 20 iki 30 kilobaitų. Kituose intervaluose iki 50 kilobaitų failų kiekiai panašūs.

Šiuose failuose rasta įvairios apimties pakeitimų. Vieniuose keitėsi tik viena, ar dalis eilutės, kituose pakeista grupė iš eilės einančių eilučių, trečiuose pakeitimai išsibarstę po visą failą. 22 pav. pateiktas pavyzdys, kaip pakito keli failai. Kairėje esantis stulpelis simbolizuoją pradinį failą, dešinėje – pakeistą. Žalia spalva pažymėtos vietos, kur pasikeitė viena arba kelios eilutės, raudona spalva rodo, kad tos dalies nėra vienam ar kitam faile.



22 pav. Pakitusių failų palyginimas

Pirmame faile buvo pakeista tik viena eilutė, antrame faile ištrintos kelios eilutės, trečiame, ketvirtame ir penktame failuose vienur buvo pakeistas turinys, kitur pridėtas arba ištrintas. Taip pat pakeitimai juose išsibarstę po visą failą.

Taip pat išanalizuota, kaip pakito šių failų dydžiai. Rasta padidėjusių, sumažėjusių ir nepakitusių. Šie rezultatai pateikti 3 lentelėje. Pateikiamas failų kiekis ir kokią dalį jie sudaro nuo visų pakitusių failų.

3 lentelė. Failų dydžio pakitimai

	Kiekis	**Dalis nuo visų pakitusių failų**
Padidėjo	3326	32,94%
Nepakito	172	1,70%
Sumažėjo	6598	65,35%

Iš lentelės duomenų matosi, kad didžiosios dalies failų dydis sumažėjo, kur kas mažiau failų, kurių dydis padidėjo ir labai maža dalis tokių, kurių turinys pakito, bet dydis išliko toks pat.

Išanalizuoti failai, kurių dydis padidėjo arba sumažėjo, siekiant nustatyti, kokio dydžio pakeitimai dažniausiai atliekami. Rezultatai suskirstyti į intervalus ir pateikti 4 lentelėje.

4 lentelė. Failų pakitimas baitais

Pakitimo dydžio intervalas (B)	**Padidėjo**	**Sumažėjo**	**Viso failų**
1 – 10	585	1328	1913
11 – 20	272	383	655
21 - 30	254	272	526
31 - 40	305	233	538
41 - 50	164	206	370
51 - 60	121	157	278
61 - 70	104	120	224
71 - 80	79	112	191
81 - 90	67	95	162
91 - 100	76	93	169
101+	1299	3188	4487

Rezultatuose matosi, kad yra didelis kiekis failų, kuriuose atlikti nuo vieno iki dešimties baitų pakeitimai. Taip pat pastebima tendencija, kad didėjant pakitimo dydžiui, mažėja failų kiekis. Iš to galima daryti išvadą, kad didžiosios dalies failų dydis pakito tik per kelis baitus.

Remiantis šios analizės rezultatais, nuspręsta eksperimentuose naudoti 1, 2, 5, 10, 20, 50 ir 100 kilobaitų dydžio failus. Juose atlikti pakeitimus pradžioje, viduryje ir pabaigoje, pakeičiant, pridedant ir ištrinant po kelis baitus. Kadangi pastebėta, kad yra daug failų, kuriuose keitėsi vos keli baitai, eksperimentuose bus keičiama, pridedama ir ištrinama po 4 baitus.

3.3. Tyrimas

Atliekant tyrimus siekiama palyginti suprojektuotą šifravimo metodą su kitais, rinkoje egzistuojančiais produktais: *eCryptfs* ir *EncFS*. Tiriama kaip pasikeičia failų dydis po užšifravimo ir atliekant failų pakeitimus:

a. pakeičiant kelis baitus kitais failų pradžioje, viduryje ir pabaigoje;
b. pridedant papildomų baitų failų pradžioje, viduryje ir pabaigoje;
c. ištrinant kelis baitus failų pradžioje, viduryje ir pabaigoje.

Taip pat tiriama, kiek duomenų yra perduodama į serverį po failo redagavimo. Tyrime naudojama NFS tinklinė failų sistema ir *Wireshark* tinklo paketų analizės programinė įranga siekiant nustatyti, kiek duomenų perduodama į serverį.

EncFs veikia su šiais nustatymais:

a. šifravimo algoritmas: AES su 16 baitų bloko šifru;
b. rakto dydis: 256 bitai;
c. failų sistemos bloko dydis: 1024 baitai;
d. failų vardai nešifruojami.

eCryptfs nustatymai:

a. šifravimo algoritmas: AES su 16 baitų bloko šifru;
b. rakto dydis: 128 bitai;
c. failų sistemos bloko dydis: 1024 baitai;
d. failų vardai nešifruojami.

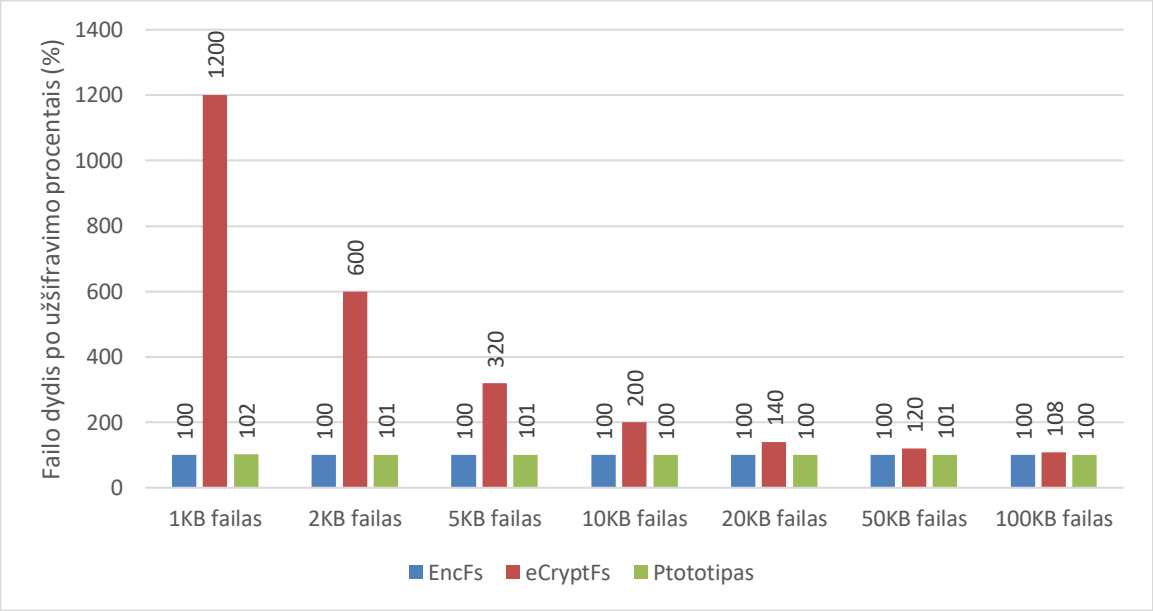
3.3.1. Užšifruoto failo dydžio tyrimas

Šiuo tyrimu siekiama nustatyti, kaip kinta skirtingos apimties failų dydžiai juos užšifravus. Tyrimo rezultatai pateikti 5 lentelėje. *EncFs* atveju dydis išliko toks pat, failus užšifravus prototipu, jų dydis šiek tiek išaugo. To priežastis – visi failo blokai yra šifruojami naudojant blokinį šifrą, todėl šiuo atveju jų ilgis privalo būti 16 kartotinis. Tam pasiekti prie bloko pabaigos yra pridedama papildomų baitų, kol jo ilgis pasidaro tinkamu. Tačiau, jei bloko ilgis iš karto yra tinkamas, prie jo yra pridedami papildomi 16 baitų. Šiuo atveju tai ir buvo padaryta. Taip pat skaičiuojant dydį buvo įskaičiuotas ir *XML* failo dydis, kuriame saugoma informacija apie failų struktūrą.

5 lentelė. Failo šifravimo tyrimo rezultatai

	Dydis (B)		
	EncFS	eCryptfs	Prototipas
1 KB failas	1 024	12 288	1 040
2 KB failas	2 048	12 288	2 064
5 KB failas	5 120	16 384	5 152
10 KB failas	10 240	20 480	10 288
20 KB failas	20 480	28 672	20 560
50 KB failas	51 200	61 440	51 456
100 KB failas	102 400	110 592	102 800

eCryptfs rezultatai šiame tyrime blogiausi – šifruojant mažos apimties failus, jų dydis išauga kelis kartus. Tyrime naudojamo mažiausio failo atveju ši informacija kelis kartus didesnė už patį failą. 23 pav. pateikta diagrama, grafiškai vaizduojanti, kiek pasikeičia užšifruoto failo dydis, lyginant su nešifruotu. Rezultatai išreikšti procentais.



23 pav. Užšifruoto failo dydžio pokyčio diagrama

Grafike matosi, kad didėjant failo dydžiui, *EncFs* ir prototipo failų procentinis padidėjimas vis mažėja. *eCryptfs* nepriklausomai nuo failo dydžio, nekeičia jo apimties. Sprendžiant iš šio tyrimo rezultatų, *eCryptfs* nerekomenduojama naudoti, jei norima šifruoti mažos apimties failus. *EncFs* šiame tyrime pasirodė geriausiai, bet ne daug nuo jo atsiliko prototipas.

3.3.2. Failų šifravimo pakeitus baitus tyrimas

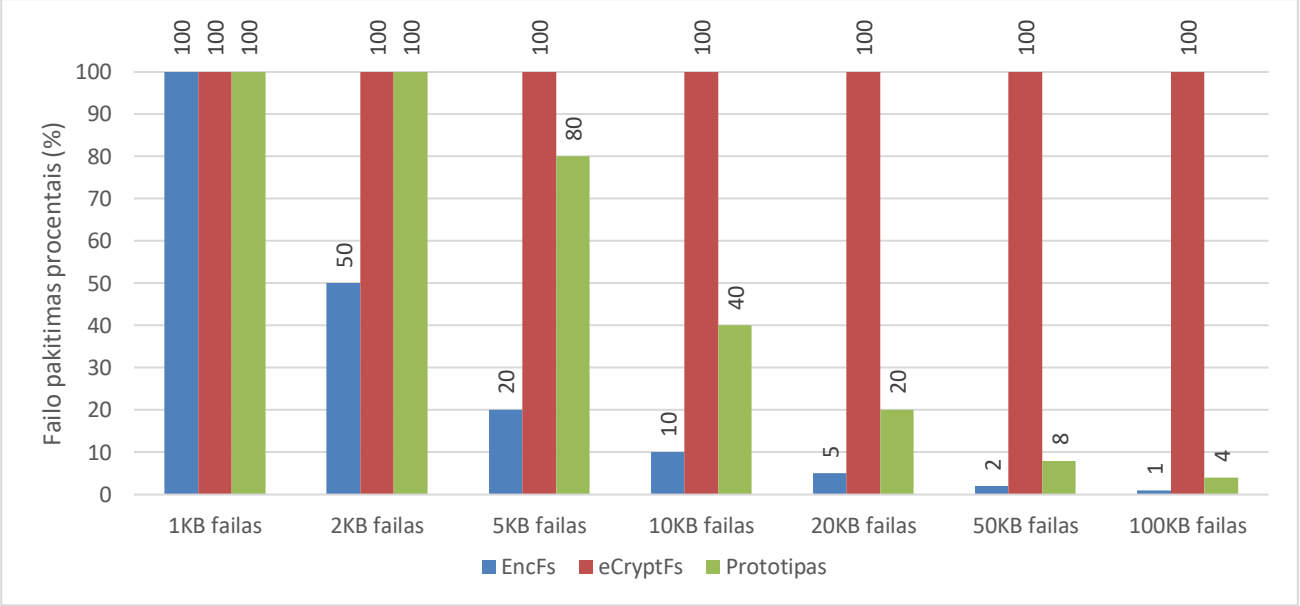
Šiuo tyrimu siekiama nustatyti kaip elgiasi skirtingos programos, kai tam tikroje failo dalyje yra pakeičiami keli baitai. Pirmiausia užšifruojamas bandymo failas ir išsaugomas gautas šifruotas failas. Tada pakeičiami 4 baitai bandymo faile ir išsaugomas naujas šifruotas failas, bei palyginamas su prieš tai išsaugotu. Tokiu būdu nustatoma, kokia dalis failo pakito. Taip pat naudojantis *Wireshark* tinklo paketų analizės programine įranga fiksuojama, kiek duomenų perduota į serverį. Fiksuojama tik ta informacija, kuri perduota po atlikto failo pakeitimo.

Pirmiausia pakeitimas atliekamas pačioje failo pradžioje, tarp 1 ir 1024 baito, kadangi *EncFs* šifruoja 1 kilobaito blokais. Rezultatai pateikti 6 lentelėje.

6 lentelė. Baitų pakeitimo failo pradžioje tyrimo rezultatai

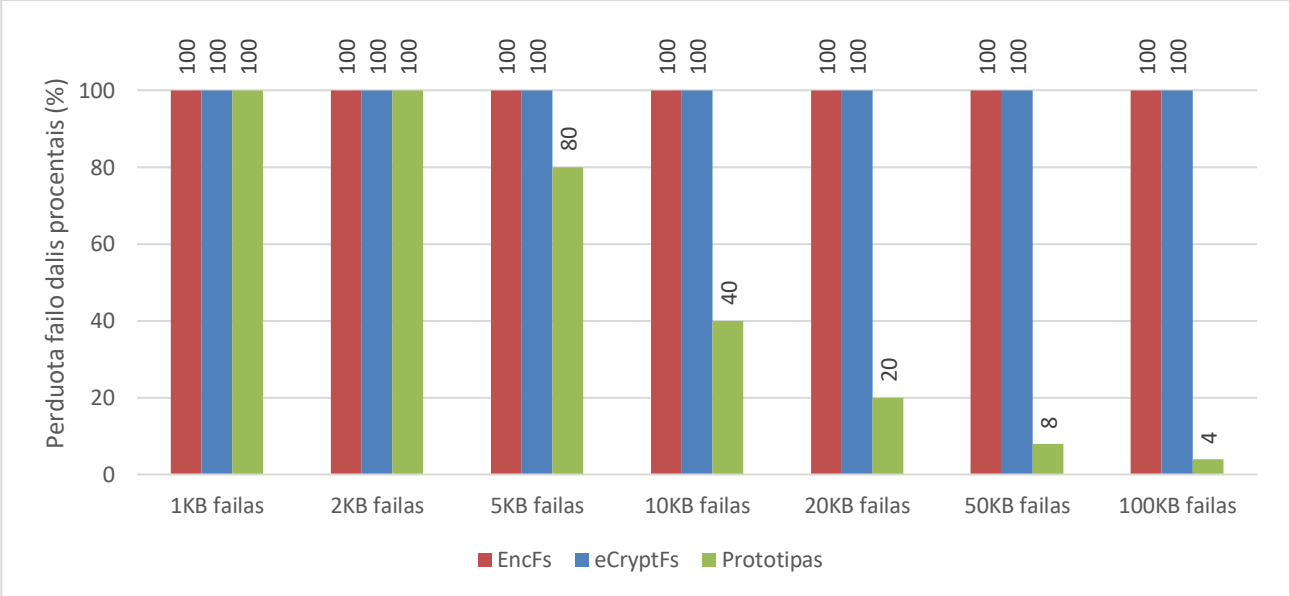
	EncFS		eCryptfs		Prototipas	
	Pakito (B)	Nusiųsta (B)	Pakito (B)	Nusiųsta (B)	Pakito (B)	Nusiųsta (B)
1 KB failas	1 024	1 024	12 288	12 288	1 040	1 040
2 KB failas	1 024	2 048	12 288	12 288	2 064	2 064
5 KB failas	1 024	5 120	16 384	16 384	4 096	4 096
10 KB failas	1 024	10 240	20 480	20 480	4 096	4 096
20 KB failas	1 024	20 480	28 672	28 672	4 096	4 096
50 KB failas	1 024	51 200	61 440	61 440	4 096	4 096
100 KB failas	1 024	102 400	110 592	110 592	4 096	4 096

Šifruojant *EncFs*, visais atvejais failai pakito tik per 1 KB, tačiau į nutolusį serverį nusiųstas visas failas. *eCryptfs* visada keitėsi ir buvo nusiųstas visas failo turinys. Šifruojant prototipu, pasikeitė visas failas kai jo dydis mažesnis už 4 KB, o didesniems failams keitėsi tik 4 KB, ir tik jie buvo nusiųsti į serverį. 24 pav. grafiškai pavaizduota kokia dalis viso šifruoto failo pakito po atlikto pakeitimo. Rezultatai pateikti procentais.



24 pav. Pakitusi šifruoto failo dalis pakeitus failo pradžią

Grafike matosi, kad beveik visais atvejais šifruojant *EncFs* įrankiu, failai pakito mažiau lyginant su prototipu. Tik 1 KB failas abiem atvejais pakito vienodai. Taip yra todėl, kad pagal nutylėjimą *EncFs* failus šifruoja skaidant į 1 KB duomenų blokus, o prototipas šifruoja 4 KB blokais. Žemiau pateiktas grafikas, vaizduojantis kiek procentų duomenų buvo nusiųsta į serverį atlikus failo pakeitimą. Procentai nurodo, kokia failo dalis buvo perduota per tinklą į serverį (žr. 25 pav.).



25 pav. Nusiųsti duomenys į serverį pakeitus failo pradžią

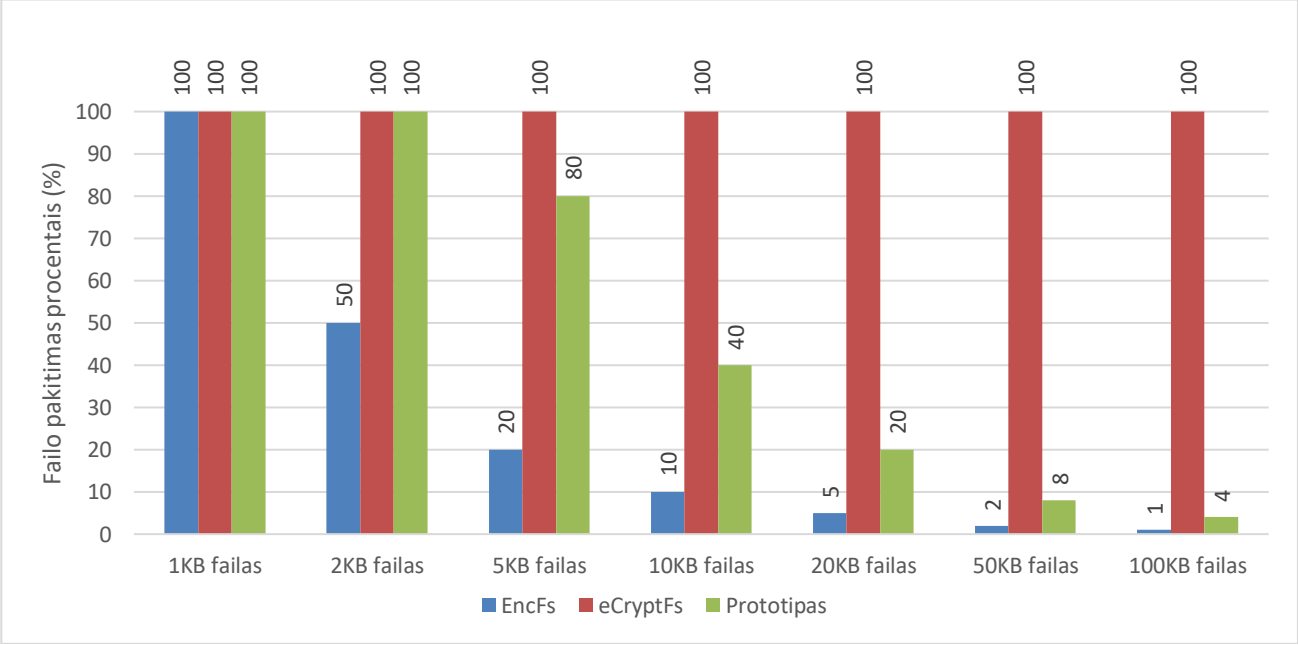
Šifruojant *EncFs* ir *eCryptfs* visais atvejais į serverį perduodamas visas failas, tuo tarpu šifruojant prototipu – perduodama tik ta dalis, kuri pasikeitė.

Toliau pakartotas prieš tai atliktas tyrimas, tačiau šį kartą keičiami baitai failo viduryje. Šie rezultatai pateikti 7 lentelėje.

7 lentelė. Baitų pakeitimo failo viduryje tyrimo rezultatai

	EncFS		eCryptfs		Prototipas	
	Pakito (B)	Nusiųsta (B)	Pakito (B)	Nusiųsta (B)	Pakito (B)	Nusiųsta (B)
1 KB failas	1 024	1 024	12 288	12 288	1 040	1 040
2 KB failas	1 024	2 048	12 288	12 288	2 064	2 064
5 KB failas	1 024	5 120	16 384	16 384	4 096	4 096
10 KB failas	1 024	10 240	20 480	20 480	4 096	4 096
20 KB failas	1 024	20 480	28 672	28 672	4 096	4 096
50 KB failas	1 024	51 200	61 440	61 440	4 096	4 096
100 KB failas	1 024	102 400	110 592	110 592	4 096	4 096

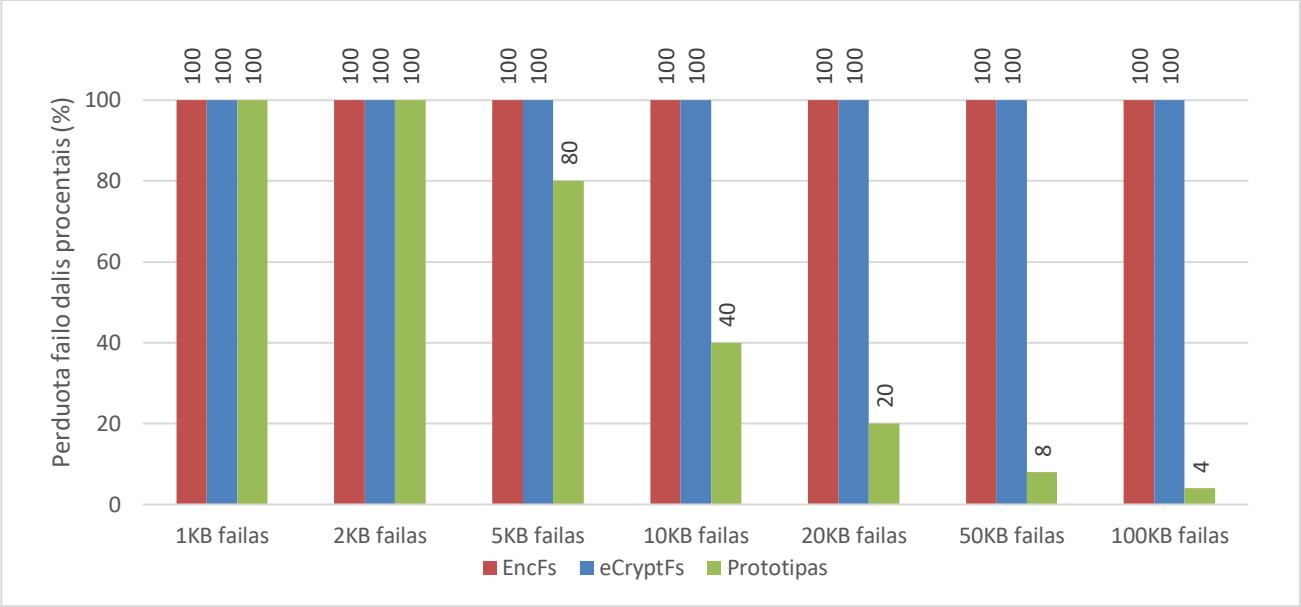
Kaip ir prieš tai atliktame tyrime, šifruojant *EncFs*, visais atvejais failai pakito tik per 1 KB, tačiau į nutolusį serverį nusiųstas visas failas. *eCryptfs* visada keitėsi ir buvo nusiųstas visas failo turinys. Šifruojant prototipu, pasikeitė visas failas kai jo dydis mažesnis už 4 KB, o didesniems failams keitėsi tik 4 KB, ir tik jie buvo nusiųsti į serverį. 26 pav. grafiškai pavaizduota kokia dalis viso šifruoto failo pakito po atlikto pakeitimo. Rezultatai pateikti procentais.



26 pav. Pakitusi šifruoto failo dalis pakeitus failo vidurį

Kaip matosi iš grafiko, rezultatai sutampa, su prieš tai gautais rezultatais, kai buvo keičiama failų pradžia. Beveik visais atvejais šifruojant *EncFs* įrankiu, failai pakito mažiau lyginant su prototipu.

Tik 1 KB failas abiem atvejais pakito vienodai. 27 pav. pateiktas grafikas, vaizduojantis kiek procentų duomenų buvo nusiųsta į serverį atlikus failo redagavimą. Procentai nurodo, kokia failo dalis buvo perduota per tinklą į serverį.



27 pav. Nusiųsti duomenys į serverį pakeitus failo vidurį

Šifruojant *EncFs* ir *eCryptfs* visais atvejais į serverį perduodamas visas failas, tuo tarpu šifruojant prototipu – perduodama tik ta dalis, kuri pasikeitė.

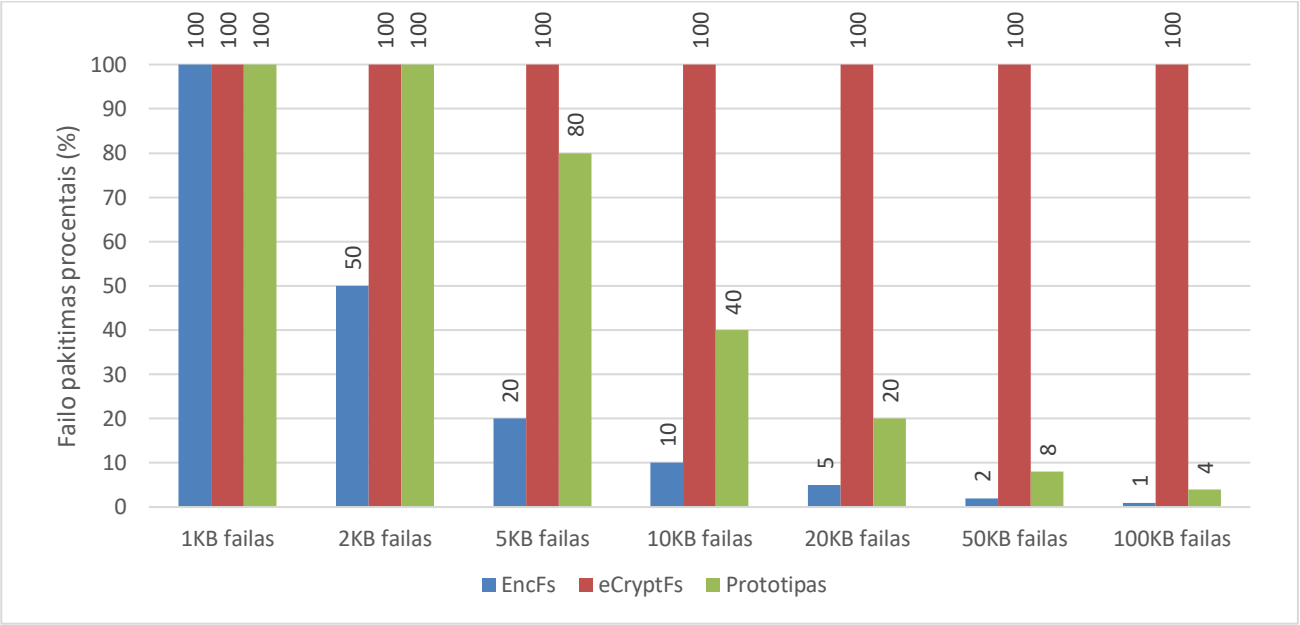
Veiksmai pakartoti pakeitimą atliekant failo pabaigoje. Rezultatai pateikti 8 lentelėje.

8 lentelė. Baitų pakeitimo failo pabaigoje tyrimo rezultatai

	EncFS		eCryptfs		Prototipas	
	Pakito (B)	Nusiųsta (B)	Pakito (B)	Nusiųsta (B)	Pakito (B)	Nusiųsta (B)
1 KB failas	1 024	1 024	12 288	12 288	1 040	1 040
2 KB failas	1 024	2 048	12 288	12 288	2 064	2 064
5 KB failas	1 024	5 120	16 384	16 384	4 096	4 096
10 KB failas	1 024	10 240	20 480	20 480	4 096	4 096
20 KB failas	1 024	20 480	28 672	28 672	4 096	4 096
50 KB failas	1 024	51 200	61 440	61 440	4 096	4 096
100 KB failas	1 024	102 400	110 592	110 592	4 096	4 096

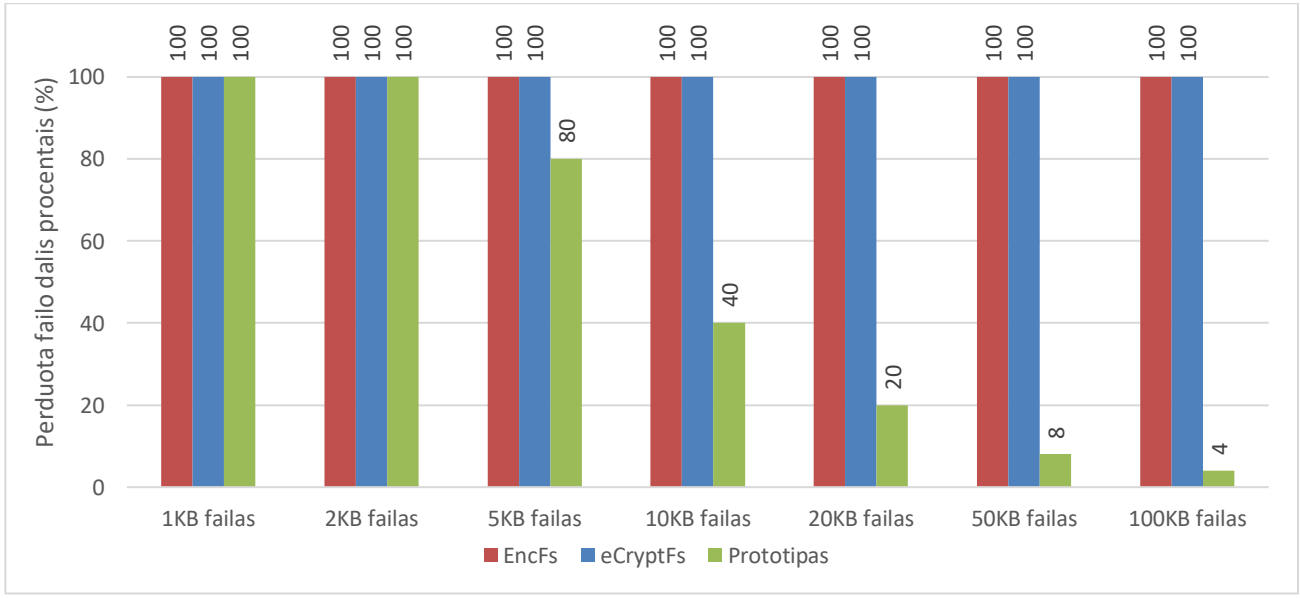
Rezultatai sutampa su prieš tai gautais. Šifruojant *EncFs*, visais atvejais failai pakito tik per 1 KB, tačiau į nutolusį serverį nusiųstas visas failas. *eCryptfs* visada keitėsi ir buvo nusiųstas visas failo turinys. Šifruojant prototipu, pasikeitė visas failas kai jo dydis mažesnis už 4 KB, o didesniems failams

keitėsi tik 4 KB, ir tik jie buvo nusiųsti į serverį. 28 pav. pavaizduota, kokia šifruoto failo dalis pasikeitė po atlikto failo pakeitimo.



28 pav. Pakitusi šifruoto failo dalis pakeitus failo pabaigą

Rezultatai sutampa, su prieš tai gautais. Beveik visais atvejais šifruojant *EncFs* įrankiu, failai pakito mažiau lyginant su prototipu. Tik 1 KB failas abiem atvejais pakito vienodai. Žemiau pateiktas grafikas, vaizduojantis kiek procentų duomenų buvo nusiųsta į serverį atlikus failo redagavimą. Procentai nurodo, kokia failo dalis buvo perduota per tinklą į serverį (žr. 29 pav.).



29 pav. Nusiųsti duomenys į serverį pakeitus failo pabaigą

Kaip ir prieš tai, rezultatai rodo, kad šifruojant *EncFs* ir *eCryptfs* visais atvejais į serverį perduodamas visas failas, tuo tarpu šifruojant prototipu – perduodama tik ta dalis, kuri pasikeitė.

Atlikus tyrimą nustatyta, kad šifruojant *eCryptfs* įrankiu, faile atlikus kelių baitų pakeitimą, visiškai pasikeičia visas failas, nepriklausomai nuo jo dydžio. Lyginant *EncFs* ir prototipu 6ifruotus failus, *EncFs* atveju kinta mažesnė failo dalis. Taip yra todėl, kad prototipas failus šifruoja didesniais duomenų blokais. Tačiau tiek *EncFs,* tiek *eCryptfs* po pakeitimo, į serverį yra perduodamas visas failas, tuo tarpu šifruojant prototipu – siunčiama tik ta dalis, kuri pakito.

3.3.3. Failų šifravimo pridėjus naujus baitus tyrimas

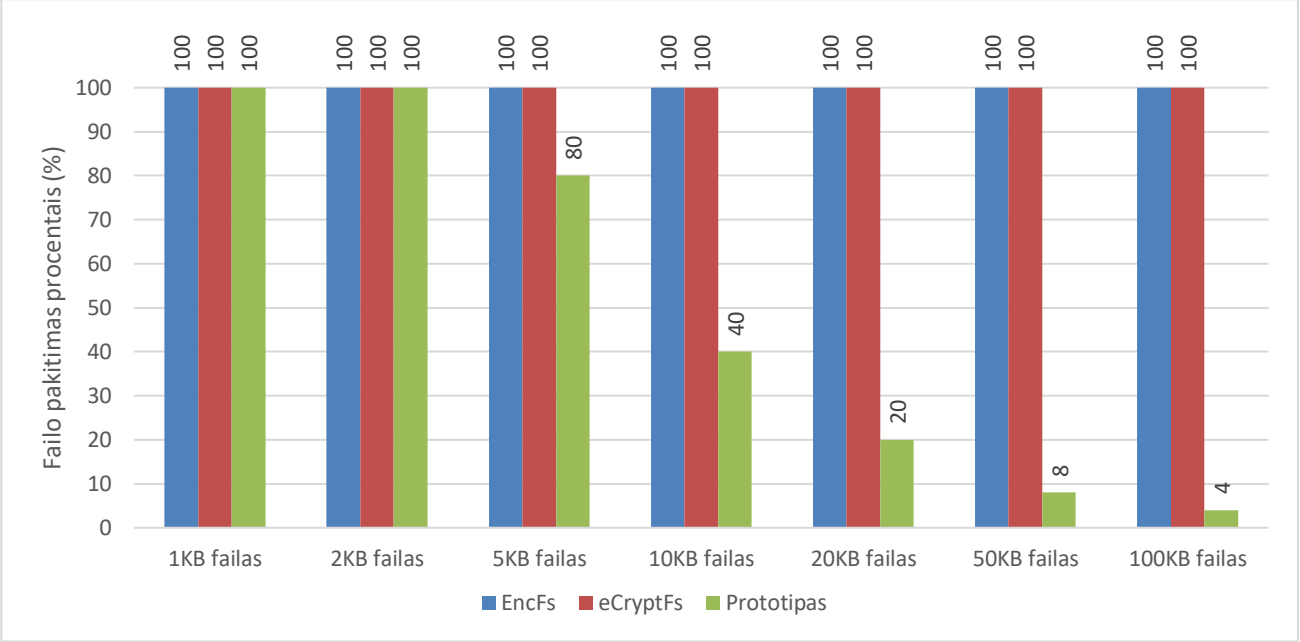
Šiuo tyrimu siekiama nustatyti kaip elgiasi skirtingos programos, kai tam tikroje failo dalyje yra pridedami keli baitai. Pirmiausia užšifruojamas bandymo failas, išsaugomas gautas šifruotas failas. Tada faile pridedami nauji baitai ir išsaugomas gautas šifruotas failas, bei palyginamas su prieš tai išsaugotu. Taip nustatoma, kokia dalis failo pakito. Taip pat naudojantis *Wireshark* tinklo paketų analizės programine įranga fiksuojama, kiek duomenų perduota į serverį. Fiksuojama tik ta informacija, kuri perduota po atlikto failo koregavimo.

Pirmiausia baitai pridedami pačioje failo pradžioje, tarp 1 ir 1024 baito, kadangi *EncFs* šifruoja 1 kilobaito blokais. Rezultatai pateikti 9 lentelėje.

9 lentelė. Baitų pridėjimo failo pradžioje tyrimo rezultatai

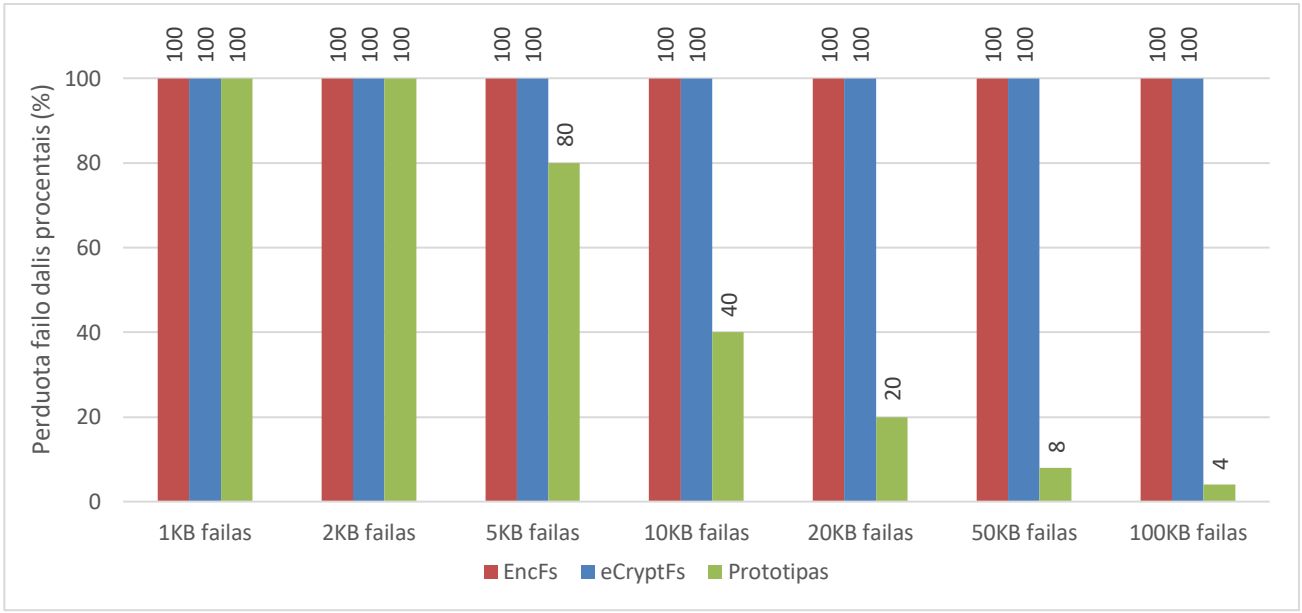
	EncFS		eCryptfs		Prototipas	
	Pakito (B)	Nusiųsta (B)	Pakito (B)	Nusiųsta (B)	Pakito (B)	Nusiųsta (B)
1 KB failas	1 024	1028	12 288	12 288	1 040	1 040
2 KB failas	2 048	2 052	12 288	12 288	2 064	2 064
5 KB failas	5 120	5 124	16 384	16 384	4096	4096
10 KB failas	10 240	10 244	20 480	20 480	4096	4096
20 KB failas	20 480	20 484	28 672	28 672	4096	4096
50 KB failas	51 200	51 204	61 440	61 440	4096	4096
100 KB failas	102 400	102 404	110 592	110 592	4096	4096

Šiuo atveju, tiek *EncFs,* tiek *eCryptfs* po pakeitimo failai visiškai pasikeitė. Prototipu šifruotuose failuose keitėsi visas failas tais atvejais, kai jo dydis mažesnis nei 4 KB, ir pasikeitė tik 4 KB kai failas didesnis. Gauti rezultatai grafiškai atvaizduoti 30 pav., pateikti procentais, kokia dalis pasikeitė.



30 pav. Pakitusi šifruoto failo dalis pridėjus papildomus baitus failo pradžioje

31 pav. pateiktas grafikas, vaizduojantis kokia dalis failo buvo nusiųsta į serverį.



31 pav. Nusiųsti duomenys į serverį pridėjus papildomus baitus failo pradžioje

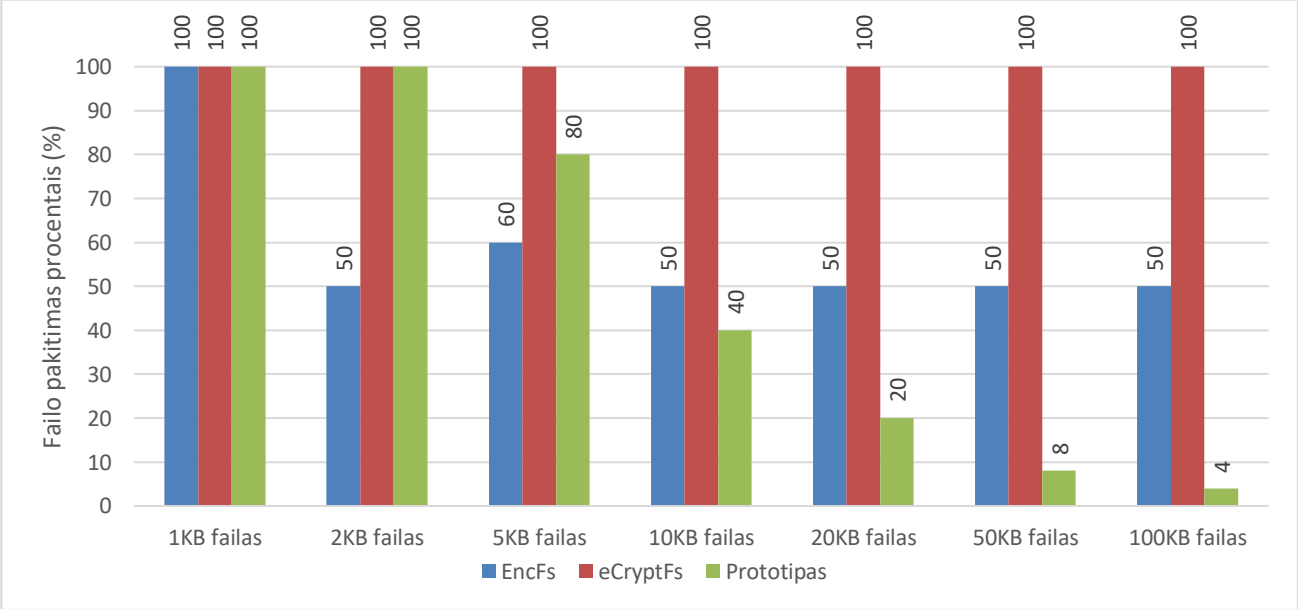
EncFs ir *eCryptfs* atvejais į serverį perduotas visas failas, šifruojant prototipu – tik ta dalis, kuri pasikeitė.

Veiksmai pakartoti su failais, tik šį kartą papildomi baitai pridėti failo viduryje. Rezultatai pateikti 10 lentelėje.

10 lentelė. Baitų pridėjimo failo viduryje tyrimo rezultatai

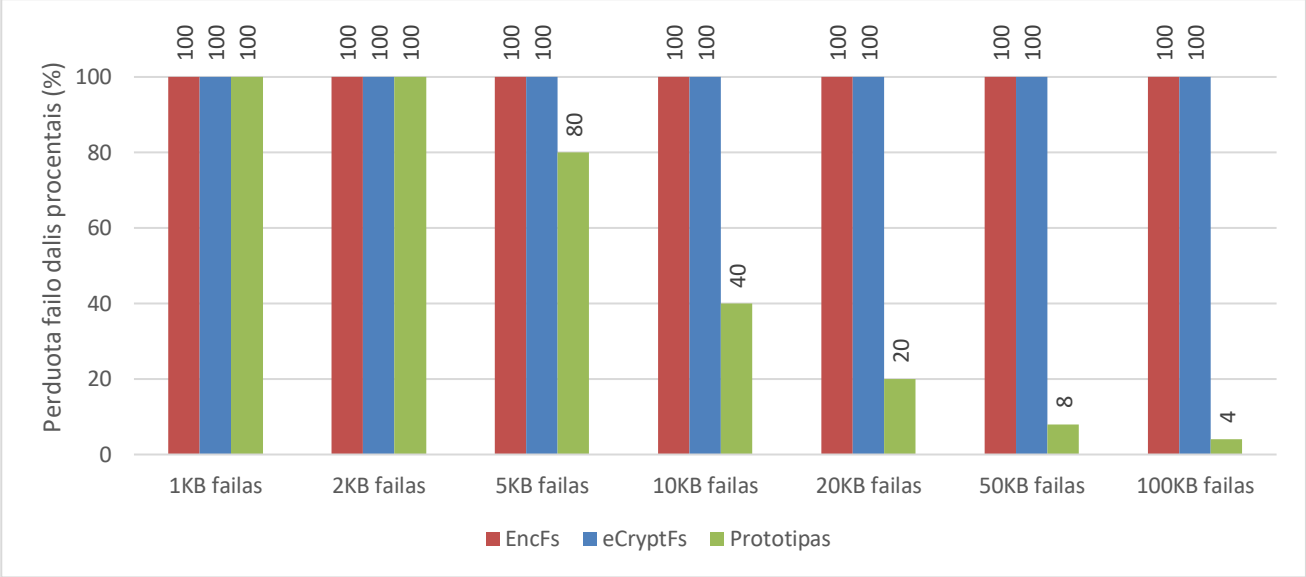
	EncFS		eCryptfs		Prototipas	
	Pakito (B)	Nusiųsta (B)	Pakito (B)	Nusiųsta (B)	Pakito (B)	Nusiųsta (B)
1KB failas	1 024	1 028	12 288	12 288	1 040	1 040
2KB failas	1 024	2 052	12 288	12 288	2 064	2 064
5KB failas	3 072	5 124	16 384	16 384	4 096	4 096
10KB failas	5 120	10 244	20 480	20 480	4 096	4 096
20KB failas	10 240	20 484	28 672	28 672	4 096	4 096
50KB failas	25 600	51 204	61 440	61 440	4 096	4 096
100KB failas	51 200	102 404	110 592	110 592	4 096	4 096507

Kaip ir anksčiau, po baitų pridėjimo *eCryptfs* šifruoti failai visiškai pasikeitė. *EncFs* šiuo atveju keitėsi tik nuo tos vietos, kur buvo pridėti nauji baitai, t. y. nuo failo vidurio. Prototipo rezultatai sutampa su gautais pridedant baitus failo pradžioje. Rezultatai grafiškai atvaizduoti 32 pav., pateikti procentais, kokia dalis viso failo pasikeitė.



32 pav. Pakitusi šifruoto failo dalis pridėjus papildomus baitus failo viduryje

Failai, kurių dydis mažesnis negu 10 KB mažiau pakito šifruojant *EncFs*, tačiau didesniuose failuose mažiau pakitimų įvyko šifruojant prototipu. 33 pav. pateiktas grafikas, vaizduojantis kokia dalis failo buvo nusiųsta į serverį.



33 pav. Nusiųsti duomenys į serverį pridėjus papildomus baitus failo viduryje

EncFs ir *eCryptfs* atvejais į serverį perduotas visas failas, šifruojant prototipu – tik ta dalis, kuri pasikeitė.

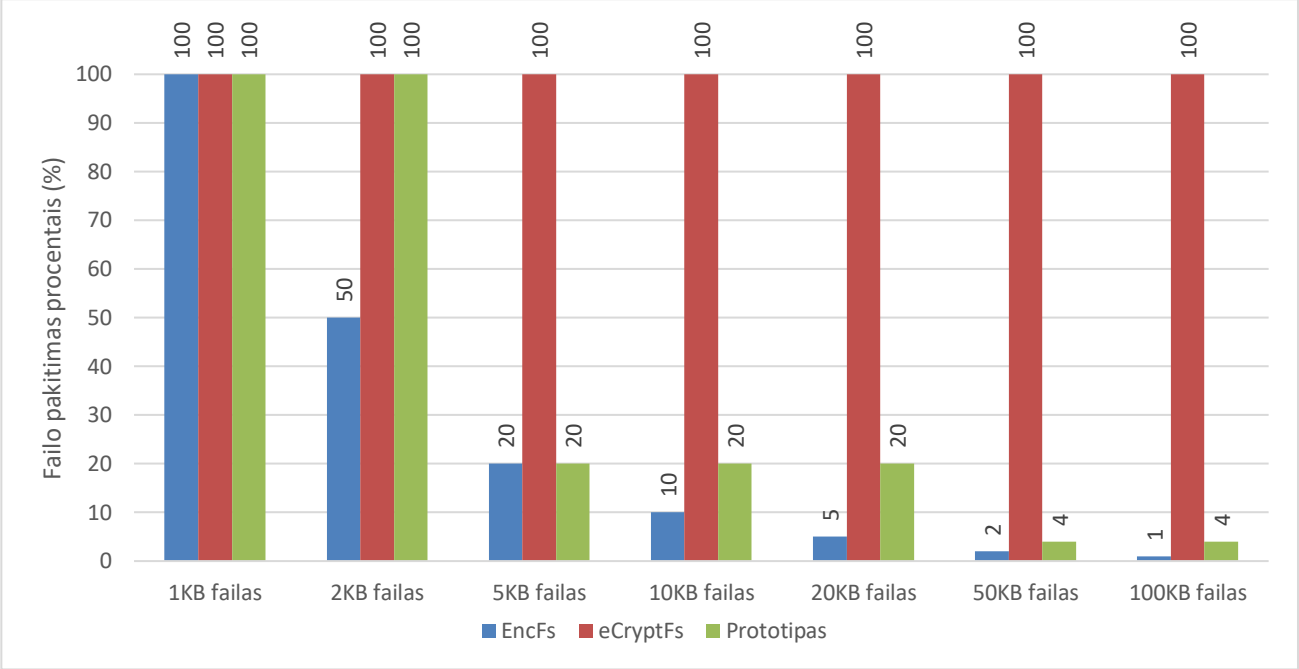
Veiksmai pakartoti baitus pridedant failo pabaigoje. Rezultatai pateikti 11 lentelėje.

11 lentelė. Baitų pridėjimo failo pabaigoje tyrimo rezultatai

	EncFS		eCryptfs		Prototipas	
	Pakito (B)	Nusiųsta (B)	Pakito (B)	Nusiųsta (B)	Pakito (B)	Nusiųsta (B)
1 KB failas	1 024	1 028	12 288	12 288	1 040	1 040
2 KB failas	1 024	2 052	12 288	12 288	2 064	2 064
5 KB failas	1 024	5 124	16 384	16 384	1 056	1 056
10 KB failas	1 024	10 244	20 480	20 480	2 096	2 096
20 KB failas	1 024	20 484	28 672	28 672	4 176	4 176
50 KB failas	1 024	51 204	61 440	61 440	2 256	2 256
100 KB failas	1 024	102 404	110 592	110 592	4 496	4 496

Kadangi baitai pridėti failo pabaigoje, *EncFs* atveju visada keitėsi tik 1 KB. *eCrytpFs* kaip ir visada – keitėsi visas failas. Prototipo atveju failo pakitimas priklausė nuo paskutinio bloko dydžio. Pavyzdžiui, išskaidžius 5 KB failą paskutinis blokas lygus 1 KB, todėl tik tiek ir pasikeitė.

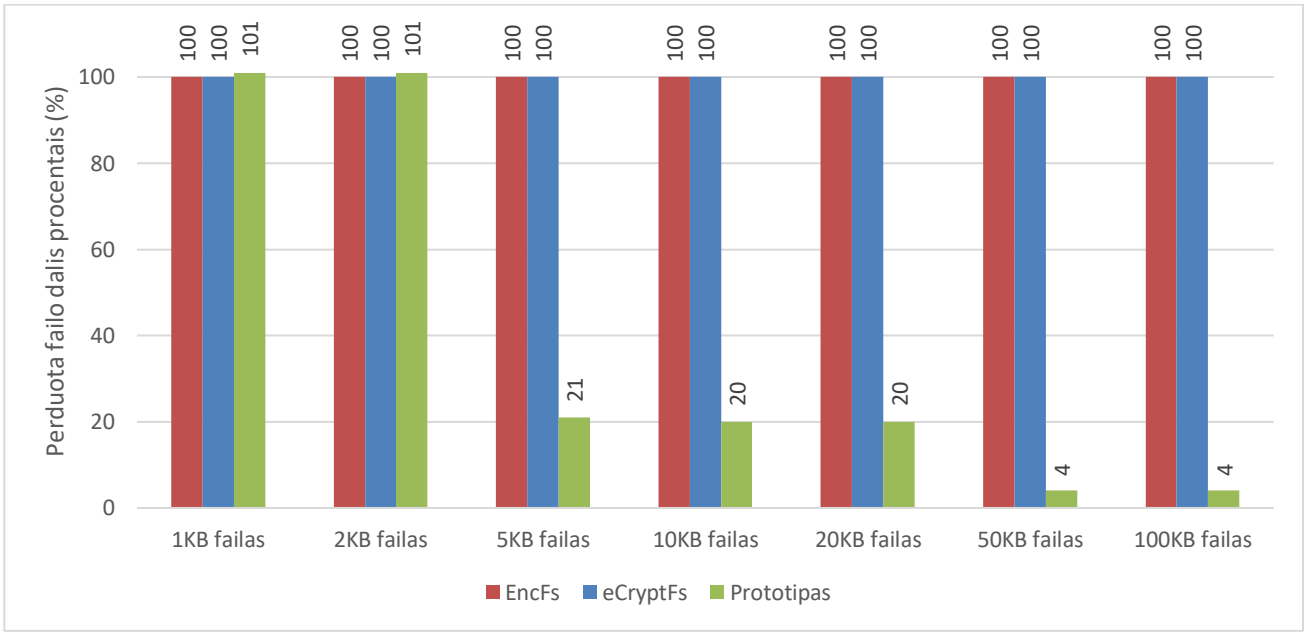
Rezultatai grafiškai pavaizduoti 34 pav., pateikti procentais, kokia dalis viso failo pasikeitė.



34 pav. Pakitusi šifruoto failo dalis pridėjus papildomus baitus failo pabaigoje

Prototipo failas pakito tiek pat arba daugiau, lyginant su *EncFs*. Taip yra dėl to, kad EncFs failą skaido į mažesnius blokus. *eCryptfs* kaip visada, keitėsi visas failas.

35 pav. pateiktas grafikas, vaizduojantis kokia failo dalis buvo nusiųsta į serverį.



35 pav. Nusiųsti duomenys į serverį pridėjus papildomus baitus failo pabaigoje

EncFs ir *eCryptfs* atvejais į serverį perduotas visas failas, šifruojant prototipu – tik ta dalis, kuri pasikeitė.

3.3.4. Failų šifravimo ištrynus baitus tyrimas

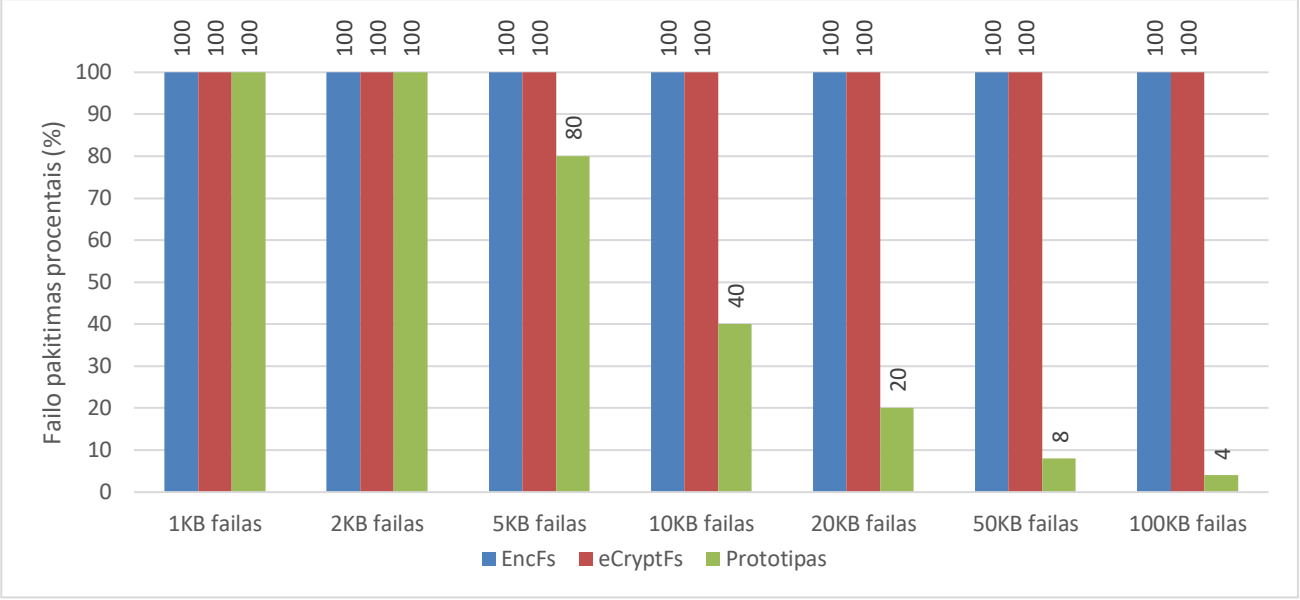
Šiuo tyrimu siekiama nustatyti kaip elgiasi skirtingos programos, kai tam tikroje failo dalyje yra ištrinami keli baitai. Pirmiausia užšifruojamas bandymo failas, išsaugomas gautas šifruotas failas. Tada faile ištrinami keli baitai ir išsaugomas gautas šifruotas failas, bei palyginamas su prieš tai išsaugotu. Taip nustatoma, kokia dalis failo pakito. Taip pat naudojantis *Wireshark* tinklo paketų analizės programine įranga fiksuojama, kiek duomenų perduota į serverį. Fiksuojama tik ta informacija, kuri perduota po atlikto failo koregavimo.

Pirmiausia baitai ištrinami pačioje failo pradžioje, tarp 1 ir 1024 baito, kadangi *EncFs* šifruoja 1 kilobaito blokais. Rezultatai pateikti 12 lentelėje.

12 lentelė. Baitų ištrynimo failo pradžioje tyrimo rezultatai

	EncFS		eCryptfs		Prototipas	
	Pakito (B)	Nusiųsta (B)	Pakito (B)	Nusiųsta (B)	Pakito (B)	Nusiųsta (B)
1 KB failas	1 024	1 020	12 288	12 288	1 040	1 024
2 KB failas	2 048	2 044	12 288	12 288	2 064	2 048
5 KB failas	5 120	5 116	16 384	16 384	4 096	4 080
10 KB failas	10 240	10 236	20 480	20 480	4 096	4 080
20 KB failas	20 480	20 476	28 672	28 672	4 096	4 080
50 KB failas	51 200	51 196	61 440	61 440	4 096	4 080
100 KB failas	102 400	102 396	110 592	110 592	4 096	4 080

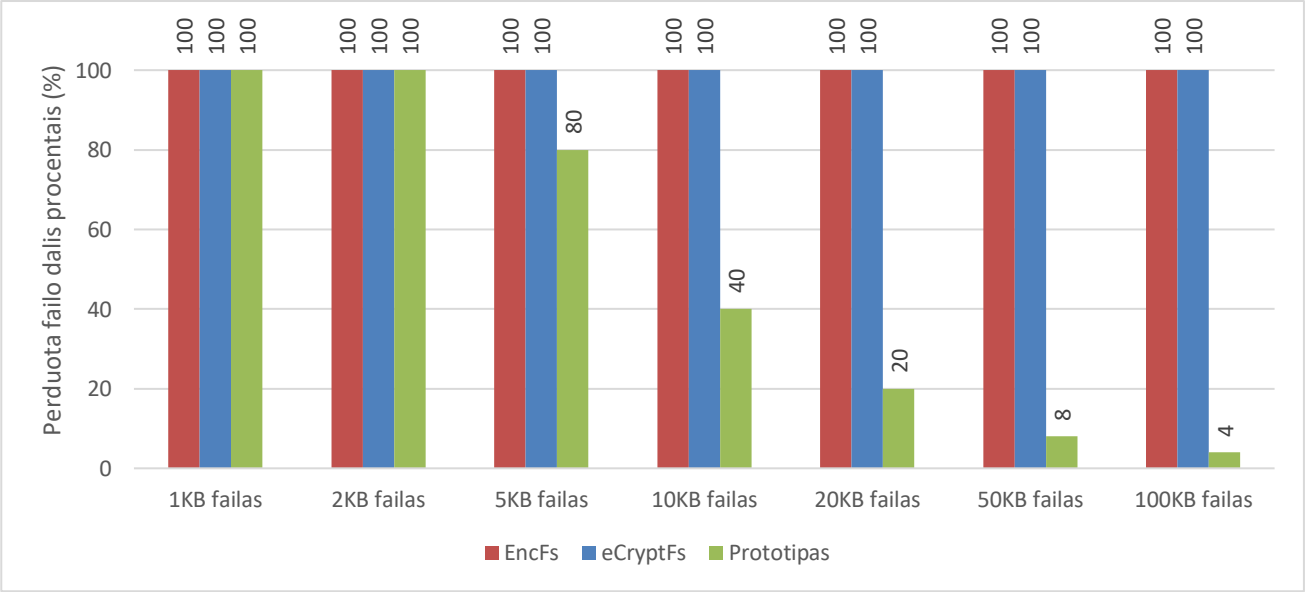
EncFs ir prototipo rezultatuose matosi, kad failuose pakito daugiau baitų, negu buvo nusiųsta. Taip yra todėl, nes pateikiama kiek baitų pakito pradinio failo atžvilgiu. Jei failo dydis buvo 1024 baitai ir ištrinti 4 baitai, traktuojama, kad pasikeitė visi 1024 pradiniai baitai. Kaip ir ankstesniuose eksperimentuose, failai užšifruoti *eCryptfs* pasikeitė visiškai. Šiuo atveju taip pat keitėsi visas failas ir šifruojant *EncFs*. Prototipo šifruotuose failuose pasikeitė tik 4 KB ar net mažiau. Rezultatai grafiškai pavaizduoti 36 pav., pateikti procentais, kokia dalis viso failo pasikeitė.



36 pav. Pakitusi šifruoto failo dalis ištrynus baitus failo pradžioje

Kadangi pakeitimas atliktas pačioje failo pradžioje, šifruojant *EncFs* programa keitėsi visas šifruotų failų turinys. Taip pat visiškai pasikeitė ir eCryptfs programa užšifruoti failai. Prototipo atveju kuo didesnis failas, tuo mažesnė dalis pasikeitė.

37 pav. pateiktas grafikas, vaizduojantis kokia dalis failo nusiųsta į serverį po jo pakeitimo. Rezultatai pateikti procentais.



37 pav. Nusiųsti duomenys į serverį ištrynus baitus failo pradžioje

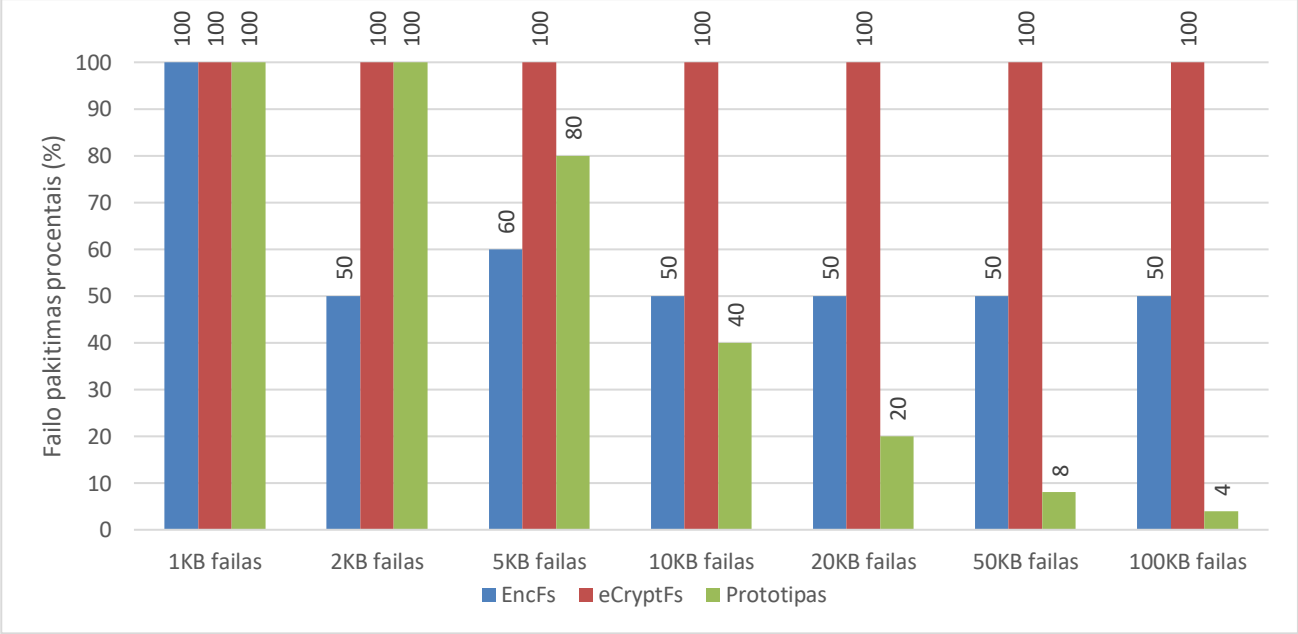
Visas failas perduotas kai šifravimui naudoti *EncFs* ir *eCryptfs*, šifruojant prototipu – perduota tik ta dalis, kur pasikeitė.

Veiksmai pakartoti baitus ištrinant failo viduryje. Rezultatai pateikti 13 lentelėje.

13 lentelė. Baitų ištrynimo failo viduryje tyrimo rezultatai

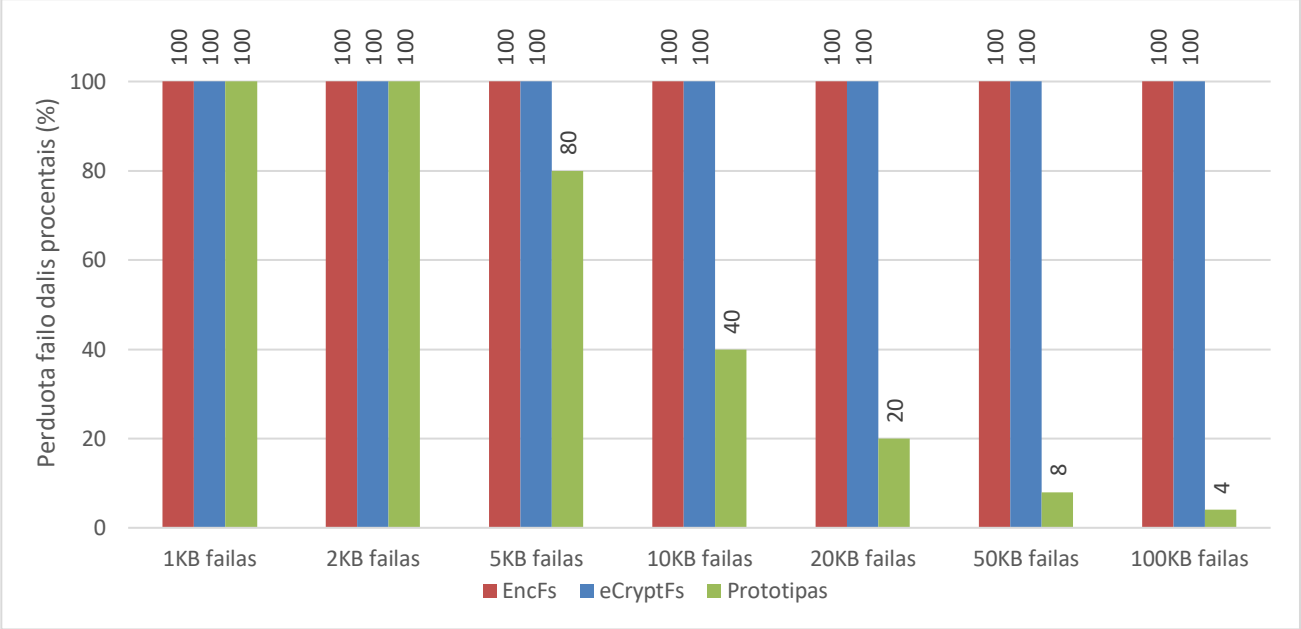
	EncFS		eCryptfs		Prototipas	
	Pakito (B)	Nusiųsta (B)	Pakito (B)	Nusiųsta (B)	Pakito (B)	Nusiųsta (B)
1 KB failas	1 024	1 020	12 288	12 288	1 040	1 024
2 KB failas	1 024	2 044	12 288	12 288	2 064	2 048
5 KB failas	3 072	5 116	16 384	16 384	4 096	4 080
10 KB failas	5 120	10 236	20 480	20 480	4 096	4 080
20 KB failas	10 240	20 476	28 672	28 672	4 096	4 080
50 KB failas	25 600	51 196	61 440	61 440	4 096	4 080
100 KB failas	51 200	102 396	110 592	110 592	4 096	4 080

Šifruojant *eCryptfs* programa pasikeitė visas failų turinys, prototipu – didžiausias pakitimas lygus 4 KB. Šifruojant *EncFs* programa pasikeitė pusė šifruoto failo turinio, kai failas didesnis už 1 KB, išskyrus 5 KB dydžio faile. Šio failo pakitimas lygus 60 procentų, nes šifruojama 1 KB dydžio blokais ir pakitimas atliktas 3 bloke. Kadangi ištrinti keli baitai, todėl keitėsi ir visi sekantys blokai ir iš viso pasikeitė 3 blokai iš 5. Rezultatai grafiškai pavaizduoti 38 pav., pateikti procentais, kokia dalis viso failo pasikeitė.



38 pav. Pakitusi šifruoto failo dalis ištrynus baitus failo viduryje

39 pav. pateiktas grafikas vaizduojantis, kokia dalis viso failo buvo nusiųsta į serverį. Rezultatai pateikti procentais.



39 pav. Nusiųsti duomenys į serverį ištrynus baitus failo viduryje

Kaip ir ankstesniuose tyrimuose, šifruojant *EncFs* ir *eCryptfs* visas failų turinys buvo nusiųstas į serverį. Šifruojant prototipu – nusiųsta tik ta dalis, kuri pasikeitė.

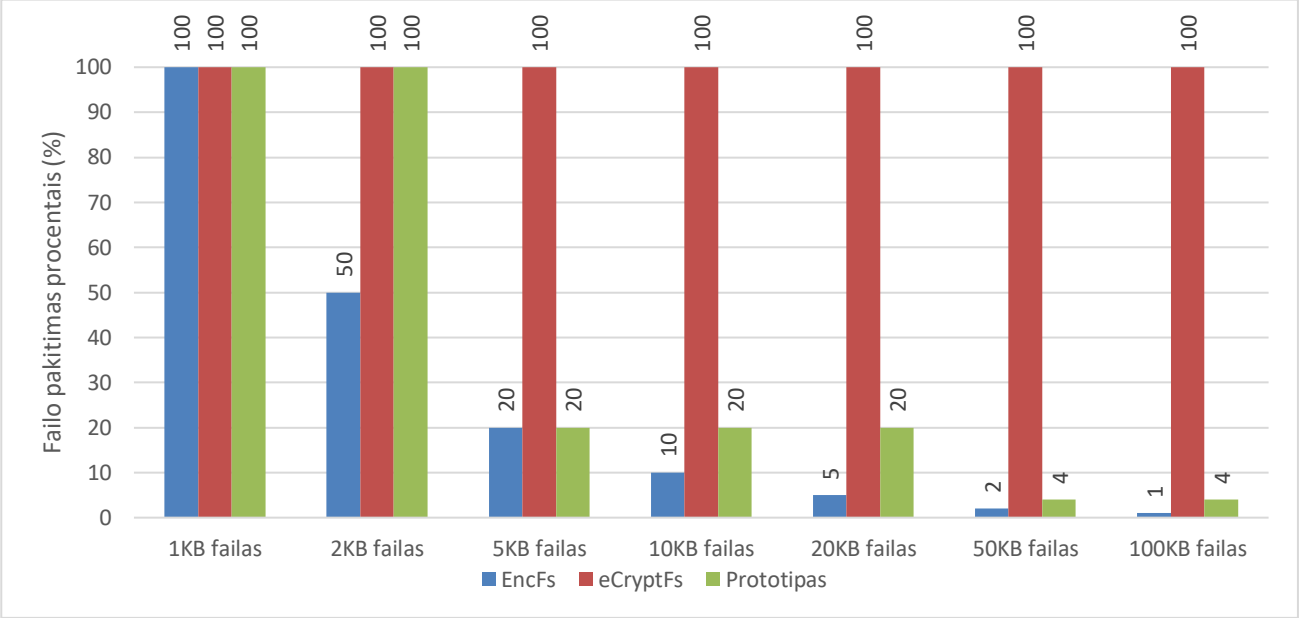
Veiksmai pakartoti baitus ištrinant failo pabaigoje. Rezultatai pateikti 14 lentelėje.

14 lentelė. Baitų ištrynimo failo pabaigoje tyrimo rezultatai

	EncFS		eCryptfs		Prototipas	
	Pakito (B)	Nusiųsta (B)	Pakito (B)	Nusiųsta (B)	Pakito (B)	Nusiųsta (B)
1 KB failas	1 024	1 020	12 288	12 288	1 040	1 024
2 KB failas	1 024	2 044	12 288	12 288	2 064	2 048
5 KB failas	1 024	5 116	16 384	16 384	1 056	1 040
10 KB failas	1 024	10 236	20 480	20 480	2 096	2 080
20 KB failas	1 024	20 476	28 672	28 672	4 176	4 160
50 KB failas	1 024	51 196	61 440	61 440	2 256	2 240
100 KB failas	1 024	102 396	110 592	110 592	4 496	4 480

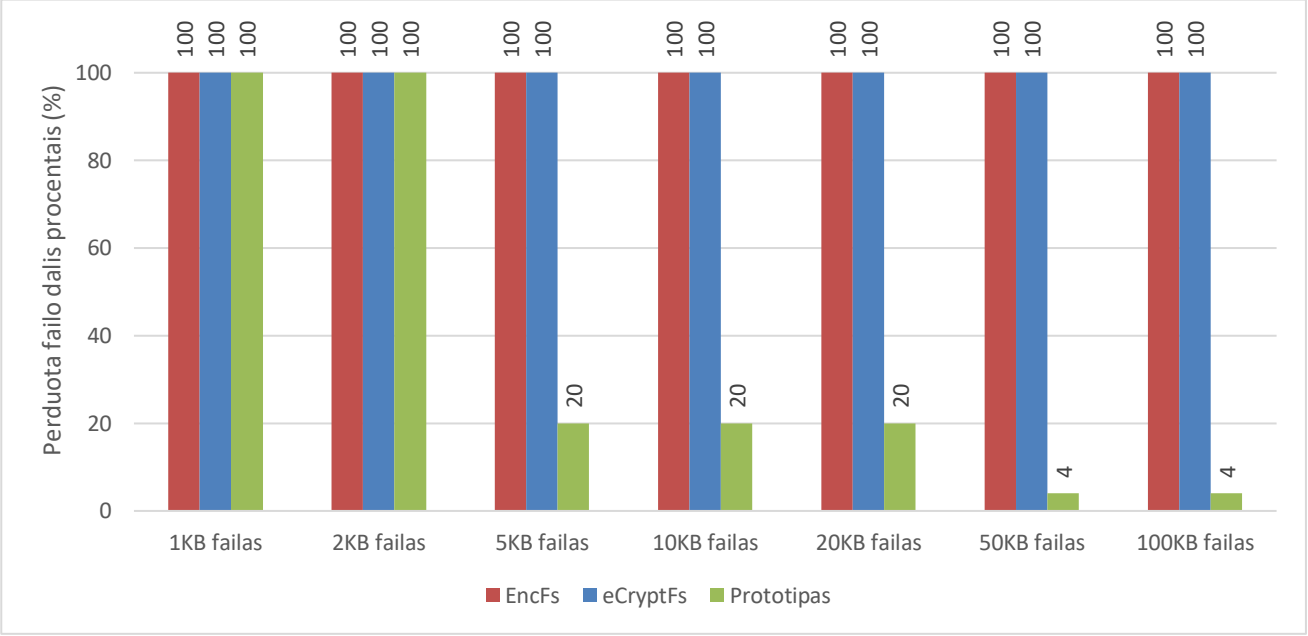
EncFs atveju visada keitėsi tik 1 KB, nes baitai ištrinti failo pabaigoje. *eCrytpFs* kaip ir visada – keitėsi visas failas. Prototipo atveju failo pakitimas priklausė nuo paskutinio bloko dydžio. Pavyzdžiui, išskaidžius 5 KB failą paskutinis blokas lygus 1 KB, todėl tok toks kiekis duomenų ir pasikeitė.

Rezultatai grafiškai pavaizduoti 40 pav., pateikti procentais, kokia dalis viso failo pasikeitė.



40 pav. Pakitusi šifruoto failo dalis ištrynus baitus failo pabaigoje

41 pav. grafiškai pateikti rezultatai, kokia dalis viso failo buvo nusiųsta į serverį.

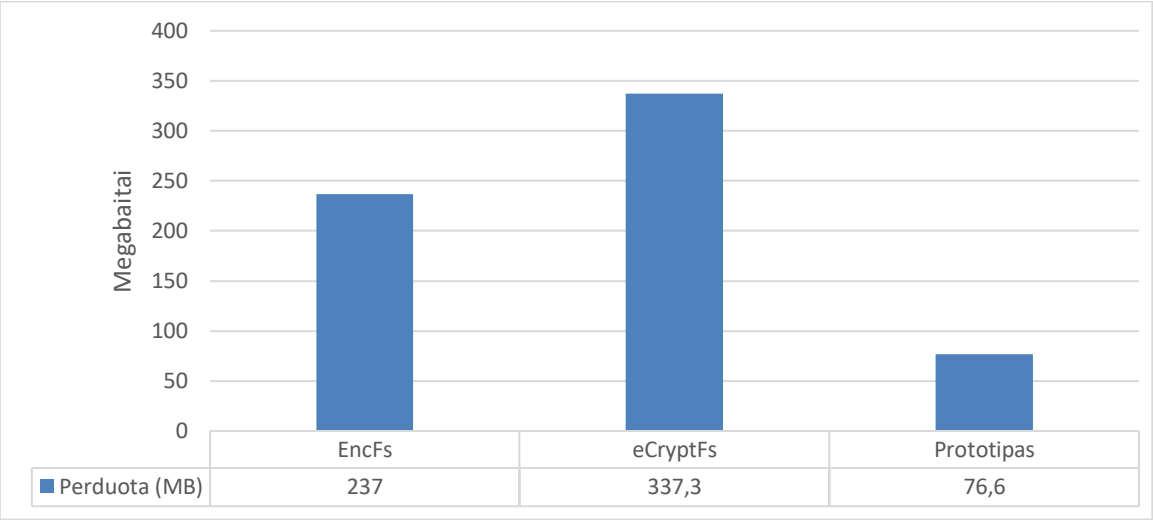


41 pav. Nusiųsti duomenys į serverį ištrynus baitus failo pabaigoje

Kaip ir ankstesniuose tyrimuose, šifruojant *EncFs* ir *eCryptfs* visas failų turinys buvo nusiųstas į serverį. Šifruojant prototipu – nusiųsta tik ta dalis, kuri pasikeitė.

3.3.5. Linux branduolio išeities kodo šifravimo tyrimas

Šiame tyrime naudojami Linux operacinės sistemos branduolio išeities kodai. Siekiama patikrinti prototipo veikimą su realiais failais. Naudojami 4.9 ir 4.10 versijų failai, šifruojami tik tie failai, kuriuose yra pakeitimų. Tarp šių dviejų versijų iš viso skiriasi 10096 failai, kurių bendras užimamas dydis pakito nuo 234 MB iki 237 MB. Tyrimo metu pirmiausia užšifruoti 4.9 versijos failai. Tada užrašomi 4.10 versijos failai ir tikrinama, kiek po užšifravimo yra perduodama duomenų į serverį. Gauti rezultatai pateikti 42 pav.



42 pav. Perduotas duomenų kiekis į serverį šifruojant Linux branduolio failus

Kaip matosi diagramoje, mažiausiai duomenų perduota failus šifruojant prototipu – tik trečdalis visų duomenų. Šifruojant *EncFs* perduotų duomenų kiekis lygus suminiam failų dydžiui, o *eCryptfs* – perduota 42% daugiau duomenų, negu suminis failų kiekis.

3.4. Tyrimo išvados

Skirtingų dydžių failų šifravimo tyrimas parodė, kad *eCryptfs* į failo antraštę įrašo nemažai informacijos, kurios dydis gali būti kelis kartus didesnis už pačio failo. Tyrimo metu nustatyta, kad failo dydis padidėja nuo 8 iki 11 KB. Dėl šios papildomos informacijos 1 KB failo dydis išauga 1200 procentų. *EncFs* rezultatai geriausi – failų dydis nė kiek nepakito. Prototipo rezultatai panašūs – failai padidėjo per 1 arba 2 procentus, priklausomai nuo failo dydžio.

Baitų pakeitimo skirtingose failo vietose tyrimas parodė, kad atlikus net ir menkiausią pakitimą *eCryptfs* programa šifruoti failai visiškai pasikeičia. *EncFs* ir prototipo atveju pasikeičia tik tas šifruotas failų sistemos blokas, kuriame buvo atliktas pakeitimas, šiame tyrime atitinkamai 1 KB ir 4KB dydžio blokai.

Baitų pridėjimo skirtingose failo vietose tyrime *eCryptfs* rezultatai panašūs, kaip ir baitų pakeitimo tyrime. Bet koks pakeitimas faile visiškai pakeičia šifruotą failą. *EncFs* failas kinta tik nuo to bloko, kuriame atsiranda nauji baitai, prieš tai esantys blokai išlieka nepakitę. Jei baitai pridėti failo pradžioje, tai pasikeičia 100 procentų failo, jei failo viduryje – 50 procentų. Prototipo šifruojamuose failuose nesvarbu, kur atliktas pakeitimas, keičiasi tik tas blokas, kuriame pridėti nauji baitai. Visi prieš tai ir po jo einantys blokai išlieka nepakitę. Kai failo dydis didesnis už 4 KB, keičiasi tik 4 KB, jei mažesnis – keičiasi visas failas.

Baitų ištrynimo skirtingose failo vietose tyrime visų tiriamų programų rezultatai tokie pat, kaip ir baitų pridėjimo tyrime.

Atliktas tyrimas su realiais *Linux* branduolio 4.9 ir 4.10 versijos failais parodė, kad šifruojant prototipu po failų pakeitimo į serverį perduoda tik 32 procentai duomenų nuo bendro failų dydžio. Šifruojant *EncFs* perduota 100 procentų duomenų, o *eCryptfs* atveju – 142 procentai duomenų.

Nors atliekant įvairius pakeitimus skirtingose failo vietose pakitimai šifruotose failuose skiriasi, tačiau dėl to, kad labiausiai paplitusių tinklinių failų sistemų protokolai neturi galimybės atnaujinti failą tik siunčiant tam tikras failo dalis, kurios pakito, *eCrpytFs* ir *EncFs* atvejais nepriklausomai nuo pakitimų kiekio, į serverį yra siunčiamas visas failas. Šioje vietoje išryškėja pagrindinis pasiūlyto naujo metodo pranašumas, nes dėl to, kad failas yra skaidomas į mažesnius, yra apeinama spraga protokoluose ir į serverį siunčiami tik tos failo dalys, kurios pakito.

4. IŠVADOS

1) Atlikus failų šifravimo metodų analizę nustatytos dvi pagrindinės problemos: smulkus failo pakeitimas gali smarkiai pakeisti šifruotą failą; labiausiai paplitusios tinklinės failų sistemos nepalaiko dalinio failo atnaujinimo ir įvykus pakeitimui, yra persiunčiamas visas failas. Abu šie trūkumai padidina tinklu perduodamų duomenų kiekį.
2) Atsižvelgiant į analizės išvadas, sudarytas naujas failų šifravimo metodas, sprendžiantis nustatytas problemas. Pasirinktas failų sistemos lygio šifravimas, nes, lyginant su disko lygio šifravimu, po failo pakeitimo į serverį perduodama daug mažiau informacijos. Siūlomas metodas paremtas failų skaidymu į mažesnius failus, taip sumažinant perduodamų duomenų kiekį pakitus failui. Po failo pakitimo, į serverį perduodami tik tie mažesni failai, kurie pakito, užuot perduodant visą failą.
3) Eksperimentiniam pasiūlyto naujo failų šifravimo metodo įvertinimui realizuotas metodo prototipas. Šifravimui naudojamas *AES* algoritmas veikiantis *CBC* režimu. Failai skaidomi į 4 KB dydžio failus.
4) Sukurtas prototipas eksperimentiškai palygintas su kitais, egzistuojančiais sprendimais. Gauti rezultatai parodė, kad priklausomai nuo failo dydžio ir atlikto pakeitimo, vienais atvejais šifruotas failas mažiau pakito šifruojant *EncFs,* kitais atvejais mažiau pakito šifruojant prototipu. Šifruojant *eCrytpFs* visais atvejais pakito visas failas ir dydis padidėjo per 8 – 10 KB. Tačiau užšifravus failus prototipu į serverį perduota mažiau duomenų lyginant su kitais, kai failo dydis didesnis negu 4 KB, ir perduota tiek pat, kai failo dydis mažesnis už 4 KB.
5) Atliktas eksperimentas su realiais Linux branduolio 4.9 ir 4.10 failais, kurio metu buvo tiriama kiek duomenų perduota į serverį užšifravus atliktus pakeitimus tarp 4.9 ir 4.10 versijų, parodė, kad failus šifruojant šiame darbe pasiūlytu nauju failų šifravimo metodu perduotų duomenų kiekis tris kartus mažesnis lyginant su *EncFs* ir 4.5 karto mažesnis lyginant su *eCryptfs*. Pakitusių failų bendras dydis lygus 237 MB, šifruojant prototipu perduota 76,6 MB, *EncFs* – 237 MB ir *eCryptfs* – 337,3 MB.

5. LITERATŪRA

[1] A. Desoky, „Cryptography: Algorithms and Standards," *Signal Processing and Information Technology,* pp. 924-929, 2015.

[2] „OWASP Guide to Cryptography," OWASP, 12 09 2015. [Tinkle]. Pasiekiama: https://www.owasp.org/index.php/Guide_to_Cryptography#Symmetric_Cryptography. [Kreiptasi 19 04 2016].

[3] M. Dworkin, „Recommendation for Block Cipher Modes of Operation," *NIST Special Publication 800-38A 2001 Edition ,* 2001.

[4] *ISO/IEC 10116:2006 Modes of operation for an n-bit block cipher,* International Organization for Standardization, 2006, p. 41.

[5] Roohi Banu, Tanya Vladimirova, „Fault-Tolerant Encryption for Space Applications," *IEEE Transactions on Aerospace and Electronic Systems,* nr. 45, pp. 266 - 279, 27 03 2009.

[6] A. M. Muteb, „A Comprehensive Literature Review of Asymmetric Key Cryptography Algorithms for Establishment of the Existing Gap," *Software Engineering Conference (MySEC),* 2015.

[7] Eligijus Sakalauskas, Tomas Blažauskas, Kęstutis Lukšys, Elektroninių dokumentų ir duomenų sauga, Kaunas: Kauno technologijos universitetas, 2008, p. 168.

[8] „What is Volume Encryption," [Tinkle]. Pasiekiama: https://www.jetico.com/web_help/bcve3/html/01_introduction/02_what_is_ve.htm. [Kreiptasi 13 01 2016].

[9] Tilo Muller, Felix C. Freiling, „A Systematic Assessment of the Security of Full Disk Encryption," *IEEE Transactions on Dependable and Secure Computing,* pp. 491 - 503, 2015.

[10] „Efficient Batched Synchronization in Dropbox-like," [Tinkle]. Pasiekiama: http://www.ccs.neu.edu/home/cbw/pdf/li_middleware13.pdf. [Kreiptasi 13 01 2016].

[11] Y. Hao, I. Utku ir S. Torsten, „Algorithms for Low-Latency Remote File Synchronization," 2008. [Tinkle]. Pasiekiama: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.304.6816&rep=rep1&type=pdf. [Kreiptasi 13 01 2016].

[12] K. J. L. Kumar, „Implementing Network File System Protocol for Highly Available Clustered Applications on Network Attached Storage," *Computational Intelligence and Communication Networks (CICN),* pp. 496-499, 2013.

[13] Ming Chen, Dean Hildebrand, Geoff Kuenning, Soujanya Shankaranarayana, Bharat Singh, Erez Zadok, „Newer Is Sometimes Better: An Evaluation of NFSv4.1," *Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems,* pp. 165-176, 2015.

[14] Russel Sandberg, David Goldberg, Steve Kleiman, Dan Walsh, Bob Lyon, „Design and implementation of the Sun Network Filesystem," 1985. [Tinkle]. Pasiekiama: http://www-inst.eecs.berkeley.edu/~cs262a/sp02/Papers/nfs.pdf. [Kreiptasi 11 04 2016].

[15] Yang Li, Li Yeli, Zheng Liangbin, „Transparent Encryption Based on Network File System Filtering Driver," *Electric Information and Control Engineering (ICEICE),* pp. 6339-6342, 2011.

[16] Alex Balducci, Sean Devlin, Tom Ritter, „Open Crypto Audit Project," 13 03 2015. [Tinkle]. Pasiekiama: https://opencryptoaudit.org/reports/TrueCrypt_Phase_II_NCC_OCAP_final.pdf. [Kreiptasi 24 02 2016].

[17] „IDRIX Cryptography and IT Security Experts," IDRIX, [Tinkle]. Pasiekiama: https://www.idrix.fr/Root/content/category/7/32/60/. [Kreiptasi 01 03 2016].

[18] Dominik Leibenger, Jonas Fortmann, Christoph Sorge, „EncFS goes Multi-User: Adding Access Control to,“ *Communications and Network Security (CNS),* pp. 525-533, 2016.

[19] T. Hornby, „EncFS Security Audit,“ 14 01 2014. [Tinkle]. Pasiekiama: https://defuse.ca/audits/encfs.htm. [Kreiptasi 13 01 2016].

[20] „eCryptfs,“ [Tinkle]. Pasiekiama: http://ecryptfs.org/about.html. [Kreiptasi 05 12 2016].

[21] T. Hornby, „eCryptfs Security Audit,“ 22 01 2014. [Tinkle]. Pasiekiama: https://defuse.ca/audits/ecryptfs.htm. [Kreiptasi 13 01 2016].

[22] R. Laboratories, „PKCS #5 v2.1: Password-Based Cryptography Standard,“ 2012. [Tinkle]. Pasiekiama: https://www.emc.com/collateral/white-papers/h11302-pkcs5v2-1-password-based-cryptography-standard-wp.pdf. [Kreiptasi 15 12 2016].

[23] Meltem Sönmez Turan, Elaine Barker, William Burr, and Lily Chen, „Recommendation for Password-Based Key Derivation,“ 2010. [Tinkle]. Pasiekiama: http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-132.pdf. [Kreiptasi 15 12 2016].

[24] E. Barker, „National Institute of Standards and Technology,“ [Tinkle]. Pasiekiama: http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r4.pdf. [Kreiptasi 15 03 2017].

[25] Elaine Barker, Allen Roginsky, „National Institute of Standards and Technology,“ 11 2015. [Tinkle]. Pasiekiama: http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar1.pdf. [Kreiptasi 15 03 2017].