Kauno technologijos universitetas

Informatikos fakultetas

Kriptografinių raktų valdymo metodas saugiam duomenų ištrynimui

Baigiamasis magistro studijų projektas

Artas Kuzmenko

Projekto autorius

Prof. Algimantas Venčkauskas

Vadovas

Kaunas, 2026

Kauno technologijos universitetas

Informatikos fakultetas

Kriptografinių raktų valdymo metodas saugiam duomenų ištrynimui

Baigiamasis magistro studijų projektas

Informacijos ir informacinių technologijų sauga (6211BX008)

Artas Kuzmenko

Projekto autorius

Prof. Algimantas Venčkauskas

Vadovas

Doc. Rasa Bruzgienė

Recenzentė

Kaunas, 2026

Kauno technologijos universitetas

Informatikos fakultetas

Artas Kuzmenko

Kriptografinių raktų valdymo metodas saugiam duomenų ištrynimui

Akademinio sąžiningumo deklaracija

Patvirtinu, kad:

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

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

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

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

Artas Kuzmenko

Patvirtinta elektroniniu būdu

Artas Kuzmenko. Kriptografinių raktų valdymo metodas saugiam duomenų ištrynimui. Magistro studijų baigiamasis projektas / vadovas prof. VENČKAUSKAS Algimantas; Kauno technologijos universitetas, Informatikos fakultetas.

Studijų kryptis ir sritis (studijų krypčių grupė): Informatikos mokslai, Informacijos ir informacinių technologijų sauga.

Reikšminiai žodžiai: saugus duomenų ištrynimas, kriptografinis duomenų naikinimas, raktų valdymas, DEK, KEK, voko šifravimas, virtualioji infrastruktūra.

Kaunas, 2026. 89 p.

Santrauka

Magistro darbe nagrinėjama saugaus duomenų ištrynimo problema paskirstytoje, debesijos ir virtualiojoje infrastruktūroje. Darbo tikslas – išanalizuoti kriptografinių raktų valdymo metodus, taikomus saugiam duomenų ištrynimui, ir pasiūlyti sprendimą, pagrįstą duomenų nebeiššifruojamumu bei praktiniu jų turinio nebeatkuriamumu. Atlikta analizė parodė, kad tradiciniai metodai, tokie kaip loginis ištrynimas ar perrašymas, tokiose aplinkose neužtikrina pakankamo negrįžtamumo, todėl tinkamiausia kryptis yra kriptografinis duomenų naikinimas.

Darbe pasiūlytas kriptografinių raktų valdymo metodas, kuriame failo turinys šifruojamas duomenų šifravimo raktu DEK, o pats DEK papildomai apsaugomas raktų šifravimo raktu KEK. Siūlomo metodo ypatybė yra ta, kad KEK nėra saugomas kaip vientisas ilgalaikis raktas, bet išvedamas iš naudotojo rakto dalies, serverio rakto dalies ir druskos reikšmės. Toks sprendimas leidžia panaikinti galimybę atkurti duomenis, panaikinant raktų grandinės atkuriamumą, net jei užšifruotas objektas dar išlieka saugykloje.

Metodui praktiškai įvertinti sukurtas funkcinis prototipas, realizuotas naudojant „Node.js“, „Express.js“, „PostgreSQL“ ir „MinIO“. Prototipe failų ir DEK apsaugai naudotas AES-256-GCM algoritmas, o KEK išvesti – HKDF funkcija. Eksperimentinis vertinimas parodė, kad sistema korektiškai atkuria failus iki saugaus ištrynimo momento, aptinka modifikuotą rakto failą, užtikrina šifruotų duomenų vientisumą ir leidžia įgyvendinti atrankinį saugų ištrynimą. Gauti rezultatai rodo, kad pasiūlytas metodas yra praktiškai realizuojamas ir tinkamas saugiam duomenų ištrynimui virtualiojoje infrastruktūroje.

Artas Kuzmenko. Cryptographic Key Management Method for Secure Data Deletion. Master's Final Degree Project / supervisor prof. VENČKAUSKAS Algimantas; Faculty of Informatics, Kaunas University of Technology.

Study field and area (study field group): Informatics Sciences, Information and Information Technology Security.

Keywords: secure data deletion, cryptographic erasure, key management, DEK, KEK, envelope encryption, virtualized infrastructure.

Kaunas, 2026. 89 p.

Summary

This master's thesis addresses the problem of secure data deletion in distributed, cloud-based and virtualized infrastructures. The aim of the thesis is to analyse cryptographic key management methods applied to secure data deletion and to propose a solution based on the inability to decrypt the data and practical unrecoverability. The analysis showed that traditional methods, such as logical deletion or overwriting, do not ensure sufficient irreversibility in such environments; therefore, cryptographic erasure was selected as the most suitable approach.

The thesis proposes a cryptographic key management method in which file content is encrypted using a data encryption key, while the data encryption key itself is additionally protected by a key encryption key. The key feature of the proposed method is that the key encryption key is not stored as a single long-term secret, but is derived from a user-controlled key part, a server-side key part and a salt value. This makes it possible to eliminate data recoverability by removing the recoverability of the key chain, even if the encrypted object still remains in storage.

To evaluate the proposed method in practice, a functional prototype was developed using „Node.js", „Express.js", „PostgreSQL" and „MinIO". AES-256-GCM was used to protect files and the data encryption key, while HKDF was used to derive the key encryption key. The experimental evaluation showed that the system correctly restores files before secure deletion, detects modified key files, ensures the integrity of encrypted data and enables selective secure deletion. The obtained results indicate that the proposed method is practically implementable and suitable for secure data deletion in virtualized infrastructure.

Turinys

Lentelių sąrašas **8**
Paveikslų sąrašas **9**
Santrumpų ir terminų sąrašas **10**
Įvadas **13**
1. Saugaus duomenų ištrynimo problema ir kriptografinių raktų valdymo metodų analizė. **14**
1.1. Probleminė sritis 14
1.1.1. Duomenų saugojimo ir naikinimo ypatumai debesijos ir virtualiojoje infrastruktūroje 14
1.1.2. Duomenų likučių, versijų ir atsarginių kopijų problema 15
1.1.3. Fizinės laikmenos kontrolės nebuvimas 15
1.1.4. Duomenų atkūrimo rizika po ištrynimo 16
1.1.5. Probleminės srities apibendrinimas 16
1.2. Esami duomenų ištrynimo metodai 17
1.2.1. Loginis duomenų ištrynimas 17
1.2.2. Perrašymu pagrįstas duomenų naikinimas 19
1.2.3. Kriptografinis duomenų naikinimas 21
1.2.4. Hibridinis duomenų naikinimo metodas 23
1.2.5. Esamų metodų palyginimas 25
1.2.6. Poskyrio apibendrinimas 27
1.3. Standartai ir metodikos 27
1.3.1. Bendrasis duomenų apsaugos reglamentas ir teisė būti pamirštam 27
1.3.2. NIST SP 800-88 rekomendacijos 28
1.3.3. NIST SP 800-132 ir NIST SP 800-57 raktų išvedimo bei valdymo principai 28
1.3.4. ISO/IEC 27001 ir ISO/IEC 27002 nuostatos 29
1.3.5. Poskyrio apibendrinimas 29
1.4. Esami saugaus duomenų ištrynimo sprendimai 29
1.4.1. Raktų atskyrimu pagrįsti sprendimai 30
1.4.2. Smulkiai valdomo kriptografinio naikinimo sprendimai 31
1.4.3. Verifikuojamo ištrynimo sprendimai 33
1.4.4. Komercinių debesijos platformų saugaus duomenų ištrynimo mechanizmai 34
1.4.5. KEK valdymo kryptys 37
1.4.6. Esamų sprendimų palyginimas 42
1.4.7. Poskyrio apibendrinimas 42
1.5. Analizės išvados 43
1.5.1. Metodų analizės rezultatai 43
1.5.2. Kriptografinių raktų valdymo principų apibendrinimas 43
1.5.3. Analizės išvada 44
2. Kriptografinių raktų valdymo metodas saugiam duomenų ištrynimui **45**
2.1. Metodo paskirtis ir esmė 45
2.2. Metodo veikimo logika 45
2.3. Metodo koncepcinė schema 46
2.4. Kriptografinė schema ir duomenų bei raktų srautai 48
2.4.1. Duomenų šifravimo rakto generavimas ir failo šifravimas 49
2.4.2. Raktų šifravimo rakto išvedimas 50
2.4.3. Duomenų šifravimo rakto apsauga 52

2.4.4. Duomenų ir raktų paskirstymas tarp prototipo komponentų ... 53
2.4.5. Failo atkūrimo srautas ... 54
2.4.6. Saugaus naikinimo srautas ... 55
2.5. Metodo taikymo logika saugaus duomenų ištrynimo atveju ... 57
2.6. Skyriaus apibendrinimas ... 57
3. Prototipo projektavimas ir realizacijos planas ... 58
3.1. Prototipo tikslas, paskirtis ir ribos ... 58
3.2. Prototipo architektūra ir komponentai ... 58
3.3. Sistemos projektiniai modeliai ... 60
3.3.1. Panaudojimo atvejų diagrama ... 60
3.3.2. Konceptinis duomenų modelis ... 62
3.3.3. Diegimo diagrama ... 63
3.4. Naudojamos technologijos ir realizavimo sprendimai ... 64
3.4.1. Taikomosios programos lygmens technologijos ... 64
3.4.2. Kriptografinės realizacijos priemonės ... 64
3.4.3. Failų saugojimo sprendimas ... 65
3.4.4. Metaduomenų ir raktų informacijos saugojimas ... 65
3.4.5. Virtualizacijos ir infrastruktūros sprendimai ... 65
3.4.6. Naudotojo valdomos rakto dalies realizacija ... 66
3.4.7. Audito ir veiksmų registravimo sprendimai ... 66
3.5. Prototipo realizacijos planas ... 66
3.6. Skyriaus apibendrinimas ... 67
4. Prototipo testavimas ir eksperimentinis įvertinimas ... 68
4.1. Kokybinis siūlomo metodo vertinimas ... 68
4.2. Prototipo realizacijos aplinka ... 70
4.3. Testavimo strategija ... 71
4.4. Matuojami rodikliai ir eksperimentinių duomenų rinkimas ... 72
4.5. Testiniai duomenys ir eksperimentinės sąlygos ... 72
4.6. Eksperimentų aprašymai ... 73
4.6.1. Funkcinio korektiškumo eksperimentai ... 73
4.6.2. Rakto failo ir kriptografinio vientisumo eksperimentai ... 73
4.6.3. Kriptografinio naikinimo ir atrankinio naikinimo eksperimentai ... 74
4.6.4. Veikimo trukmės ir dviejų veiksnių eksperimentai ... 74
4.7. Eksperimentų rezultatų apdorojimas ir pateikimas ... 75
4.8. Eksperimentų rezultatai ir jų analizė ... 77
4.8.1. Failo dydžio ir naudotojo rakto ilgio įtaka bendram failo įkėlimo laikui ... 77
4.8.2. Failo dydžio ir naudotojo rakto ilgio įtaka bendram failo atkūrimo laikui ... 78
4.8.3. Failo dydžio ir naudotojo rakto ilgio įtaka KEK išvedimo trukmei ... 79
4.8.4. Failo dydžio ir naudotojo rakto ilgio įtaka failo šifravimo ir iššifravimo trukmėms ... 80
4.8.5. Funkcinių ir saugumo eksperimentų rezultatų analizė ... 82
4.8.6. Bendras eksperimentinių rezultatų apibendrinimas ... 83
4.9. Skyriaus apibendrinimas ... 83
Išvados ... 85
Informacijos šaltinių sąrašas ... 86

Lentelių sąrašas

1 lentelė. Loginio duomenų ištrynimo metodo vertinimas 18
2 lentelė. Perrašymu pagrįsto metodo vertinimas 21
3 lentelė. Kriptografinio duomenų naikinimo metodo vertinimas 23
4 lentelė. Hibridinio duomenų naikinimo metodo vertinimas 25
5 lentelė. Esamų duomenų naikinimo metodų palyginimas 26
6 lentelė. KEK valdymo modelių palyginimas 41
7 lentelė. Esamų saugaus duomenų naikinimo sprendimų palyginimas 42
8 lentelė. Duomenų naikinimo metodų kokybinis palyginimas 69
9 lentelė. Vidutinės failo įkėlimo trukmės priklausomai nuo failo dydžio ir S_{usr} ilgio 75
10 lentelė. Vidutinės failo atkūrimo trukmės priklausomai nuo failo dydžio ir S_{usr} ilgio 76
11 lentelė. Vidutinės KEK išvedimo trukmės priklausomai nuo failo dydžio ir S_{usr} ilgio 76
12 lentelė. Vidutinės failo šifravimo trukmės priklausomai nuo failo dydžio ir S_{usr} ilgio 76
13 lentelė. Vidutinės failo iššifravimo trukmės priklausomai nuo failo dydžio ir S_{usr} ilgio 77
14 lentelė. Funkcinių ir saugumo eksperimentų rezultatų suvestinė 77

Paveikslų sąrašas

1 pav. Duomenų gyvavimo ciklas debesijos ir virtualiojoje infrastruktūroje 14
2 pav. Loginio duomenų ištrynimo veikimo logika paskirstytoje infrastruktūroje..... 18
3 pav. Perrašymu pagrįsto duomenų naikinimo veikimo ir ribotumų schema..... 20
4 pav. DEK ir KEK pagrindu veikiančio voko šifravimo schema..... 22
5 pav. Hibridinio duomenų naikinimo veikimo schema 24
6 pav. FADE (angl. *File Assured Deletion*) architektūra [22] 30
7 pav. Užtikrinto duomenų ištrynimo (angl. *assured deletion*) konceptualioji architektūra [5]..... 31
8 pav. ADAA sistemos modelis [23] 32
9 pav. ATDD sistemos architektūra ir operacijos [24] 32
10 pav. EVSD sistemos modelis [26] 33
11 pav. „Azure“ saugaus duomenų ištrynimo mechanizmų schema 35
12 pav. AWS saugaus duomenų ištrynimo mechanizmų schema..... 36
13 pav. Slaptažodžiu grindžiamos rakto išvedimo schemos principas [19]..... 38
14 pav. Kelių veiksnių pagrindu išvedamo rakto sudarymo principas [33]..... 39
15 pav. Siūlomo kriptografinių raktų valdymo metodo koncepcinė schema..... 47
16 pav. Failo įkėlimo, šifravimo ir išsaugojimo seka 50
17 pav. Raktų išvedimo ir DEK apsaugos seka 52
18 pav. Failo atsisiuntimo seka..... 55
19 pav. Saugaus duomenų ištrynimo seka 56
20 pav. Siūlomo saugaus duomenų naikinimo prototipo koncepcinė architektūra..... 59
21 pav. Saugaus duomenų naikinimo prototipo panaudojimo atvejų diagrama 61
22 pav. Saugaus duomenų naikinimo prototipo konceptinis duomenų modelis..... 62
23 pav. Saugaus duomenų naikinimo prototipo diegimo diagrama..... 63
24 pav. Failo dydžio ir naudotojo rakto ilgio įtaka bendram failo įkėlimo laikui 78
25 pav. Failo dydžio ir naudotojo rakto ilgio įtaka bendram failo atkūrimo laikui 79
26 pav. Failo dydžio ir naudotojo rakto ilgio įtaka KEK išvedimo trukmei..... 80
27 pav. Failo dydžio ir naudotojo rakto ilgio įtaka failo šifravimo trukmei..... 81
28 pav. Failo dydžio ir naudotojo rakto ilgio įtaka failo iššifravimo trukmei 82

Santrumpų ir terminų sąrašas

Santrumpos:

ABE – atributais grindžiamas šifravimas;

ADAA – smulkiai valdomo užtikrinto duomenų naikinimo schema, grindžiama atributų asociacija;

API – taikomųjų programų programavimo sąsaja;

ATDD – smulkiai valdomo laiku pagrįsto užtikrinto duomenų naikinimo schema;

AWS – *Amazon Web Services*;

BYOK – „atsinešk savo raktą“ modelis;

CP-ABE – šifruoto teksto politika grindžiamas atributinis šifravimas;

DEK – duomenų šifravimo raktas;

EDPB – Europos duomenų apsaugos valdyba;

ENISA – Europos Sąjungos kibernetinio saugumo agentūra;

EVSD – efektyvi ir verifikuojama saugaus duomenų naikinimo schema;

FADE – failų užtikrinto duomenų naikinimo sistema;

GDPR – Bendrasis duomenų apsaugos reglamentas;

HOTP – vienkartinis slaptažodis pagal skaitiklį;

HSM – aparatinis saugumo modulis;

ISO – Tarptautinė standartizacijos organizacija;

KEK – raktų šifravimo raktas;

KMS – raktų valdymo paslauga;

MFKDF – kelių veiksnių raktų išvedimo funkcija;

NIST – Nacionalinis standartų ir technologijų institutas;

PBKDF – slaptažodžiu grindžiama rakto išvedimo funkcija;

PPSS – slaptažodžiu apsaugotas paslapties padalijimas;

PUF – fiziškai neklonuojama funkcija;

S3 – *Amazon S3* objektinė saugykla;

SGX – saugiosios vykdymo aplinkos technologija;

SLA – paslaugų lygio susitarimas;

SSD – kietakūnis diskas;

SSE-KMS – serverio pusės šifravimas naudojant *AWS KMS* raktus;

TOTP – vienkartinis slaptažodis pagal laiką;

VM – virtualioji mašina;

WORM – „rašyti vieną kartą, skaityti daug kartų“ principas.

Terminai:

Auditas – veiksmų, įvykių ir valdymo sprendimų registravimas bei tikrinimas, siekiant pagrįsti jų teisėtumą, teisingumą ir atsekamumą.

Atsarginė kopija – duomenų kopija, skirta informacijai atkurti praradimo, sugadinimo ar incidento atveju.

Atsekamumas – galimybė nustatyti, kas, kada ir kokį veiksmą atliko sistemoje.

Daugianuomiškumas – debesijos aplinkos savybė, kai ta pati fizinė ar virtualioji infrastruktūra aptarnauja kelis naudotojus ar organizacijas.

Debesijos platforma – nuotolinių skaičiavimo, saugojimo ir kitų informacinių technologijų paslaugų aplinka, pasiekiama per tinklą.

Duomenų gyvavimo ciklas – nuoseklus duomenų egzistavimo procesas nuo sukūrimo ir saugojimo iki naudojimo, archyvavimo ir pašalinimo.

Duomenų likučiai – po ištrynimo infrastruktūroje likusi informacija ar jos fragmentai, kurie tam tikromis sąlygomis vis dar gali būti atkurti.

Duomenų šifravimas – procesas, kurio metu aiškaus teksto duomenys paverčiami šifruotu tekstu taip, kad be atitinkamo rakto jų nebūtų galima perskaityti.

Duomenų šifravimo raktas – raktas, naudojamas konkretaus failo, objekto ar duomenų bloko turiniui tiesiogiai šifruoti.

Išteklių telkimas – debesijos modelio savybė, kai fiziniai ir virtualieji ištekliai dinamiškai paskirstomi daugeliui naudotojų.

Ištrynimo žymė – žymeklis, rodantis, kad objektas laikomas ištrintu, nors jo ankstesnės versijos ar turinys vis dar gali egzistuoti saugykloje.

Kliento valdomas raktas – šifravimo raktas, kurio valdymą kontroliuoja pats klientas, o ne debesijos paslaugų teikėjas.

Kriptografinis duomenų naikinimas – metodas, kai prieiga prie užšifruotų duomenų panaikinama sunaikinant arba padarant nebepasiekiamą jiems iššifruoti reikalingą raktą.

Laikinas išsaugojimas po ištrynimo – mechanizmas, leidžiantis tam tikrą laiką atkurti objektą po jo ištrynimo.

Metaduomenys – papildoma informacija apie duomenis, pavyzdžiui, jų sukūrimo laikas, versija, savininkas ar susietas rakto identifikatorius.

Nekeičiamasis saugojimas – saugojimo režimas, kai objektų negalima keisti ar ištrinti iki nustatyto laikotarpio pabaigos.

Negrįžtamumas – savybė, rodanti, kad pašalinti ar nebepasiekiami duomenys nebegali būti praktiškai atkurti.

Objektinė saugykla – duomenų saugojimo modelis, kai informacija laikoma kaip atskiri objektai su turiniu, identifikatoriumi ir metaduomenimis.

Objektų užrakinimas – saugojimo mechanizmas, ribojantis objekto ištrynimą ar modifikavimą nustatytą laikotarpį arba iki teisinio sulaikymo pabaigos.

Rakto gyvavimo ciklas – visas rakto egzistavimo procesas nuo sugeneravimo iki naudojimo, rotacijos ir sunaikinimo.

Raktų hierarchija – kelių lygių raktų organizavimo modelis, kuriame vieni raktai naudojami duomenims šifruoti, o kiti – šiems raktams apsaugoti.

Raktų šifravimo raktas – aukštesnio lygio raktas, naudojamas duomenų šifravimo raktui apsaugoti.

Raktų sunaikinimas – procesas, kurio metu raktas pašalinamas arba padaromas nebepasiekiamas, todėl nebegali būti naudojamas duomenims atkurti.

Saugojimo politika – taisyklių rinkinys, nustatantis, kiek laiko ir kokiomis sąlygomis duomenys turi būti laikomi, apsaugomi ar pašalinami.

Smulkiai valdomas naikinimas – galimybė pašalinti tik konkretų objektą, duomenų klasę, atributą ar laikotarpį, nepaveikiant kitų duomenų.

Šifruotas tekstas – užšifruota duomenų forma, kurios be atitinkamo rakto neįmanoma tiesiogiai perskaityti.

Teisinis sulaikymas – teisiniais ar reguliaciniais tikslais taikomas ribojimas, neleidžiantis ištrinti tam tikrų duomenų ar objektų.

Užtikrintas duomenų naikinimas – modelis, kai duomenys tampa nebeprieinami arba nebeatkuriami, o toks rezultatas gali būti pagrįstas techninėmis ir valdymo priemonėmis.

Verifikuojamas ištrynimas – duomenų naikinimo modelis, kuriame sukuriamas patikrinamas įrodymas, kad naikinimo procesas buvo atliktas korektiškai.

Voko šifravimas – šifravimo modelis, kai duomenys šifruojami DEK, o pats DEK papildomai apsaugomas KEK.

Įvadas

Šiuolaikinėje skaitmeninėje aplinkoje duomenų saugumas yra vienas svarbiausių organizacijų veiklos užtikrinimo veiksnių, ypač kai informacija saugoma, perduodama ir apdorojama paskirstytoje, debesijos ar virtualiojoje infrastruktūroje. Tokios aplinkos suteikia lankstumo, mastelio keitimo ir prieinamumo privalumų, tačiau kartu kelia naujų iššūkių, susijusių su duomenų konfidencialumu, prieigos kontrole ir galutiniu jų panaikinimu. Praktikoje duomenų ištrynimas tokiose sistemose ne visada reiškia visišką informacijos pašalinimą, nes duomenys gali išlikti atsarginėse kopijose, versijų kūrimo mechanizmuose, replikose, momentinėse kopijose ar kituose infrastruktūros sluoksniuose. Dėl šios priežasties saugus duomenų ištrynimas turi būti vertinamas ne tik kaip techninis failo pašalinimo veiksmas, bet ir kaip platesnė informacijos saugos bei duomenų valdymo problema.

Darbo naujumas ir aktualumas.

Darbo aktualumą lemia sparčiai augantis paskirstytų ir debesijos technologijų taikymas tiek viešajame, tiek privačiajame sektoriuje. Organizacijos, tvarkančios jautrius ar asmens duomenis, privalo užtikrinti, kad, pasibaigus duomenų gyvavimo ciklui, informacija būtų panaikinta taip, kad jos nebebūtų galima praktiškai atkurti. Šis reikalavimas svarbus ne tik informacijos saugos, bet ir teisinio reguliavimo, atitikties bei atsakomybės požiūriu. Tačiau paskirstytoje infrastruktūroje fizinė laikmenos kontrolė dažniausiai nėra tiesiogiai prieinama, todėl tradiciniai duomenų naikinimo būdai ne visada yra pakankami ar praktiškai įgyvendinami. Dėl to vis didesnę reikšmę įgauna kriptografiniai metodai, kuriuose duomenų negrįžtamumas siejamas su šifravimo raktų valdymu.

Darbo naujumą lemia tai, kad saugaus duomenų ištrynimo problema nagrinėjama per kriptografinių raktų valdymo prizmę. Šiame darbe pagrindinis dėmesys skiriamas ne fiziniam visų duomenų kopijų pašalinimui, bet tokiam raktų valdymo metodui, kuris leistų panaikinti galimybę atkurti duomenų turinį. Analizuojama, kaip duomenų šifravimo rakto ir raktų šifravimo rakto hierarchija, voko šifravimo principas, atskirtas raktų valdymo sluoksnis ir alternatyvūs KEK atkūrimo modeliai gali būti panaudoti saugiam duomenų ištrynimui užtikrinti. Tokiu būdu darbo centre atsiduria ne vien pats ištrynimo faktas, bet kriptografinių raktų gyvavimo ciklo valdymas kaip pagrindinė negrįžtamumo prielaida.

Darbo tikslas – išanalizuoti kriptografinių raktų valdymo metodus, taikomus saugiam duomenų ištrynimui, ir pasiūlyti sprendimą, pagrįstą duomenų nebeiššifruojamumu bei praktiniu jų turinio nebeatkuriamumu.

Darbo uždaviniai:

1. išanalizuoti saugaus duomenų ištrynimo problemą paskirstytoje, debesijos ir virtualiojoje infrastruktūroje;
2. ištirti pagrindinius duomenų ištrynimo metodus ir įvertinti jų tinkamumą nagrinėjamai infrastruktūrai;
3. išanalizuoti kriptografinių raktų valdymo principus, apimančius DEK ir KEK hierarchiją, raktų gyvavimo ciklą, raktų išvedimą ir rakto sunaikinimo logiką;
4. palyginti mokslinėje literatūroje siūlomus ir praktikoje taikomus sprendimus, susijusius su saugiu duomenų ištrynimu ir raktų valdymu;
5. suprojektuoti ir praktiškai įvertinti kriptografinių raktų valdymo metodu pagrįstą sprendimą saugiam duomenų ištrynimui.

1. Saugaus duomenų ištrynimo problema ir kriptografinių raktų valdymo metodų analizė

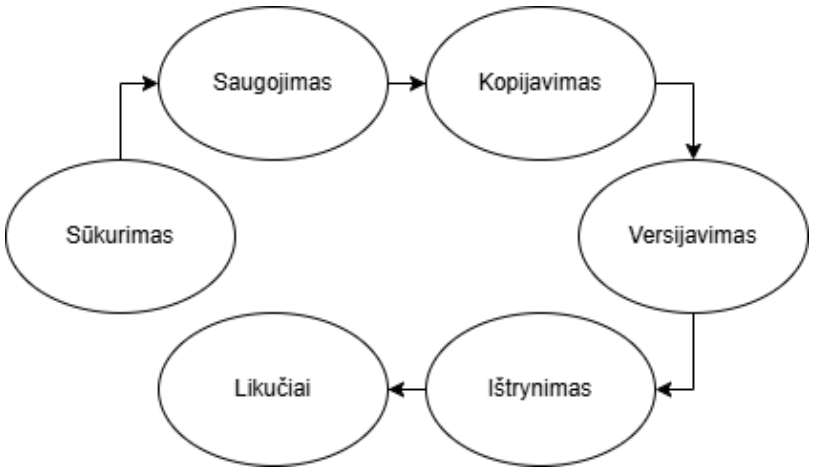
1.1. Probleminė sritis

1.1.1. Duomenų saugojimo ir naikinimo ypatumai debesijos ir virtualiojoje infrastruktūroje

Vienas iš esminių debesijos požymių yra išteklių telkimas (angl. *resource pooling*) ir daugianuomiškumas (angl. *multi-tenancy*), kai fiziniai ir virtualieji ištekliai dinamiškai paskirstomi daugeliui naudotojų, o klientas paprastai neturi tikslios kontrolės ar žinojimo apie realią išteklių vietą [1]. Dėl to duomenų apsauga debesijoje tampa ne vien technologiniu, bet ir valdymo uždaviniu: naudotojas pasitiki paslaugų teikėjo architektūra, izoliavimo mechanizmais ir paslaugų lygio susitarimais (angl. *Service Level Agreements, SLA*), kuriuose turi būti apibrėžti saugos, kokybės ir atsakomybės aspektai [1], [2].

Pagrindinės grėsmės debesijoje saugomiems duomenims apima neautorizuotą prieigą, duomenų nutekėjimą, duomenų pažeidimus, neteisėtą atskleidimą, duomenų modifikavimą ir pasiekiamumo praradimą [2], [3]. Europos Sąjungos kibernetinio saugumo agentūros (ENISA) 2024 m. grėsmių ataskaitoje pabrėžiama, kad grėsmės duomenims dažniausiai nukreiptos į prieigos prie jų blokavimą arba duomenų modifikavimą, siekiant paveikti sistemų veikimą, praktikoje tokios grėsmės dažniausiai pasireiškia duomenų pažeidimais ir nutekėjimais [3]. Naujesni debesijos saugos tyrimai taip pat rodo, kad be išorinių atakų reikšmingą riziką kelia nesąžiningi paslaugų teikėjo veiksmai, kenkėjiški vidiniai naudotojai, klastojimo atakos ir duomenų vientisumo pažeidimai [2].

Papildomą sudėtingumą lemia tai, kad debesijoje duomenų apsauga neapsiriboja vien prieigos kontrole. Nacionalinio standartų ir technologijų instituto (NIST) 2024 m. publikacijoje pažymima, kad vien teisių suteikimo nepakanka, nes duomenų sauga turi apimti ir duomenų prieigos bei nutekėjimo analizę, kai duomenys perduodami tarp skirtingų tarnybų, taikomųjų programų sąsajų (angl. *Application Programming Interfaces, API*) ir tinklo segmentų. Tai ypač svarbu debesijos architektūrose, kuriose plačiai taikomos mikroservisų, taikomųjų programų sąsajų ir tarp paslauginių duomenų srautų schemos [4].



1 pav. Duomenų gyvavimo ciklas debesijos ir virtualiojoje infrastruktūroje

Kaip matyti 1 paveiksle, duomenų gyvavimo ciklas apima ne tik sukūrimą ir saugojimą, bet ir kopijavimą, versijų kūrimą, perkėlimą, ištrynimą bei galimą atkūrimą. Dėl šios priežasties saugaus duomenų naikinimo problema turi būti nagrinėjama plačiau nei vien loginio pašalinimo lygmeniu [4], [5].

1.1.2. Duomenų likučių, versijų ir atsarginių kopijų problema

Viena svarbiausių saugaus duomenų ištrynimo problemų yra duomenų likučių problema (angl. *data remanence*). Ji reiškia, kad net ir po oficialaus ištrynimo informacija arba jos fragmentai gali išlikti infrastruktūroje ir tam tikromis sąlygomis būti atkurti. Mokslinėje literatūroje pabrėžiama, kad užtikrintam duomenų ištrynimui (angl. *assured deletion*) trukdo atsarginės kopijos, daugiasluoksnė infrastruktūra ir paskirstytas duomenų saugojimas [5]. Ramokapane ir bendraautoriai taip pat pažymi, kad visiškas ištrynimas turi apimti visas susijusias duomenų kopijas ir metaduomenis, o paskirstytoje infrastruktūroje tai tampa sudėtinga užduotimi [5].

Praktikoje šią problemą aiškiai iliustruoja debesijos paslaugų teikėjų įgyvendinti mechanizmai. „Microsoft“ dokumentacijoje nurodoma, kad įjungus objektų laikiną išsaugojimą po ištrynimo (angl. *blob soft delete*), ištrinti objektai, jų versijos, momentinės kopijos ir metaduomenys sistemoje išlieka nustatytą laikotarpį, o galutinai pašalinami tik jam pasibaigus [6]. Tai rodo, kad praktikoje ištrynimo veiksmas dažnai reiškia ne faktinį objekto išnykimą, bet jo perėjimą į laikinai atkuriamą būseną [6].

Panaši logika taikoma ir „Amazon S3“ aplinkoje. AWS dokumentacijoje nurodoma, kad paprastas ištrynimo veiksmas (angl. *DELETE*) konteineryje, kuriame įjungtas versijų kūrimas, nesunaikina objekto turinio, o tik sukuria ištrynimo žymę (angl. *delete marker*). Tokiu atveju objektas naudotojui atrodo ištrintas, tačiau ankstesnės jo versijos išlieka saugykloje [8]. Todėl paskirstytoje saugojimo aplinkoje loginis ištrynimas dažnai panaikina tik numatytąją prieigą prie objekto, bet ne visas jo egzistavimo formas [8].

Taigi duomenų likučių, versijų ir atsarginių kopijų problema yra esminė, nes ji atskleidžia skirtumą tarp administracinio objekto pašalinimo ir realaus informacijos nebeatkuriamumo. Kol išlieka versijos, atsarginės kopijos ar kiti sistemos artefaktai, nėra pakankamo pagrindo teigti, kad duomenys buvo saugiai ir negrįžtamai panaikinti [5].

1.1.3. Fizinės laikmenos kontrolės nebuvimas

Kitas svarbus veiksnys, apsunkinantis saugų duomenų ištrynimą, yra fizinės laikmenos kontrolės nebuvimas. NIST debesijos modelis pabrėžia išteklių telkimą ir jų naudojimą keliems klientams, todėl naudotojas paprastai valdo tik loginį paslaugos sluoksnį, bet neturi tiesioginės prieigos prie fizinių saugojimo įrenginių [1]. Analogiška problema kyla ir virtualiojoje infrastruktūroje, kur duomenų saugojimo logika išskaidoma tarp hipervizoriaus, virtualiųjų diskų, momentinių kopijų ir migravimo mechanizmų. Dėl to klientas ar sistemos savininkas dažnai negali savarankiškai patikrinti, kaip ir kur tiksliai jo duomenys buvo įrašyti, perrašyti ar pašalinti fizinėje infrastruktūroje [1].

Mokslinėje literatūroje ši problema siejama su tuo, kad paskirstytoje infrastruktūroje duomenys gali būti laikomi keliuose komponentuose, o jų ištrynimas priklauso nuo architektūros, atsarginių kopijų valdymo ir daugianuomiškumo modelio. Ramokapane ir bendraautoriai pažymi, kad tokioje aplinkoje duomenys gali būti laikomi bendroje arba išskaidytoje infrastruktūroje, todėl visiškas jų pašalinimas tampa sudėtingas, ypač kai informacija pereina tarp skirtingų valdymo sričių [5]. Autoriai taip pat nurodo, kad viena iš pagrindinių kliūčių visiškam ištrynimui yra tai, jog klientas neturi galimybės tiesiogiai veikti visų sluoksnių, kuriuose gali būti saugomi jo duomenys [5].

Dėl šios priežasties tradiciniai duomenų sanitarizavimo metodai, tokie kaip fizinis laikmenos sunaikinimas ar tiesioginis perrašymas, paskirstytose aplinkose dažnai tampa praktiškai

neįgyvendinami. Todėl saugaus duomenų ištrynimo sprendimai tokiose aplinkose paprastai remiasi ne fizinės laikmenos valdymu, o loginėmis ir kriptografinėmis priemonėmis, leidžiančiomis valdyti prieigą prie duomenų ir jų raktų gyvavimo ciklą [5].

1.1.4. Duomenų atkūrimo rizika po ištrynimo

Duomenų atkūrimo rizika po ištrynimo yra tiesioginė paskirstytos architektūros, versijų kūrimo ir duomenų likučių problemų pasekmė. Jei objektas tik pažymimas kaip ištrintas, tačiau jo versijos ar kopijos išlieka sistemoje, išlieka ir galimybė jį atkurti. „Microsoft“ dokumentacijoje aiškiai nurodoma, kad objektų laikinas išsaugojimas po ištrynimo (angl. *blob soft delete*) leidžia atkurti ištrintus objektus, jų versijas ir momentines kopijas per nustatytą laikymo laikotarpį [6]. Tai rodo, kad platforma sąmoningai palaiko atkūrimo funkciją kaip apsaugą nuo atsitiktinio praradimo, tačiau kartu tai reiškia, kad ištrynimas dar nėra galutinis [6].

AWS „S3“ aplinkoje panaši problema kyla dėl ištrynimo žymės (angl. *delete marker*) mechanizmo ir objektų užrakinimo funkcijos (angl. *Object Lock*). Ištrynimo žymė leidžia objektui atrodyti ištrintam, nors jo turinys gali būti išsaugotas ankstesnėse versijose [8]. Tuo tarpu objektų užrakinimas gali neleisti objektų versijų ištrinti ar perrašyti nustatytą laikotarpį arba neribotai, taikant principą „rašyti vieną kartą, skaityti daug kartų“ (angl. *write once, read many, WORM*) [9]. AWS taip pat nurodo, kad net taikant objektų užrakinimą, naujos versijos ir ištrynimo žymės vis tiek gali būti kuriamos virš apsaugotų objektų versijų [9]. Tai reiškia, kad net ir turint administracinį ketinimą ištrinti objektą, saugojimo politika gali neleisti greito galutinio pašalinimo [9].

Ši problema svarbi ne tik techniniu, bet ir informacijos saugos požiūriu. ENISA 2024 m. grėsmių aplinkos apžvalgoje pažymima, kad duomenų pažeidimai, išpirkos reikalaujančios atakos ir kitos grėsmės duomenų prieinamumui bei konfidencialumui išlieka tarp ryškiausių kibernetinio saugumo problemų Europoje [3]. Tai reiškia, kad negrįžtamai nepašalinti duomenys gali tapti papildomu rizikos šaltiniu duomenų pažeidimo, nutekėjimo ar neteisėto atkūrimo atvejais [3].

Todėl vien formalaus ištrynimo nepakanka. Reikalingas toks metodas, kuris sumažintų arba panaikintų pačią galimybę atkurti turinį, net jei tam tikri fiziniai ar loginiai informacijos pėdsakai dar kurį laiką išlieka infrastruktūroje. Būtent ši logika tolesniuose skyriuose pagrindžia perėjimą prie kriptografiniais principais grindžiamo duomenų ištrynimo ir kriptografinių raktų valdymo metodų analizės kaip tinkamesnės krypties paskirstytai infrastruktūrai [5].

1.1.5. Probleminės srities apibendrinimas

Apibendrinant galima teigti, kad saugaus duomenų ištrynimo problema kyla dėl kelių tarpusavyje susijusių veiksnių. Pirma, duomenys paskirstytoje infrastruktūroje egzistuoja ne vienoje vietoje ir ne vienu pavidalu, todėl jų gyvavimo ciklas yra sudėtingesnis nei lokaliai valdomoje aplinkoje [1], [4]. Antra, versijų kūrimas, atsarginės kopijos, ištrynimo žymės (angl. *delete marker*), laikinas išsaugojimas po ištrynimo ir principu „rašyti vieną kartą, skaityti daug kartų“ (angl. *write once, read many, WORM*) grindžiami apsaugos mechanizmai lemia, kad paprastas loginis ištrynimas dažnai nereiškia faktinio informacijos išnykimo [6], [9]. Trečia, klientas ar sistemos savininkas neturi visiškos fizinės laikmenos kontrolės, todėl negali tiesiogiai taikyti tradicinių sanitarizavimo priemonių ir yra priverstas remtis loginėmis bei kriptografinėmis priemonėmis [5].

Dėl šių priežasčių saugaus duomenų ištrynimo problema turi būti nagrinėjama ne kaip vienkartinis trynimo veiksmas, bet kaip platesnė informacijos saugos ir duomenų valdymo problema. Ji apima duomenų likučių kontrolę, atkūrimo galimybių ribojimą, atsakomybės pasidalijimą tarp infrastruktūros valdytojo ir naudotojo bei tokių metodų paiešką, kurie leistų užtikrinti praktinį negrįžtamumą.

Ši išvada tiesiogiai pagrindžia poreikį tolesniame skyriuje analizuoti esamus duomenų ištrynimo metodus ir įvertinti, kokį vaidmenį juose gali atlikti kriptografinių raktų valdymas, siekiant užtikrinti praktinį duomenų turinio nebeatkuriamumą ir panaikinti jų iššifravimo galimybę.

1.2. Esami duomenų ištrynimo metodai

Saugaus duomenų ištrynimo problemos negalima spręsti neįvertinus skirtingų šiam tikslui taikomų metodų. Praktikoje ir mokslinėje literatūroje dažniausiai išskiriami šie metodai: loginis duomenų ištrynimas, perrašymu pagrįstas duomenų naikinimas, kriptografinis duomenų naikinimas (angl. *crypto-shredding, cryptographic erase*) ir hibridinis metodas. Šie metodai skiriasi ne tik veikimo principu, bet ir tuo, kokį negrįžtamumo lygį gali užtikrinti, kiek yra pritaikomi paskirstytoje, debesijos ir virtualiojoje infrastruktūroje bei kiek leidžia pagrįsti duomenų pašalinimo faktą teisiniu ir audito požiūriu [10].

NIST SP 800-88 metodus sistemina pagal tris pagrindines kryptis: išvalymą (angl. *clear*), galutinį pašalinimą (angl. *purge)* ir sunaikinimą (angl. *destroy*). Toks skirstymas rodo, kad skirtingi metodai suteikia nevienodą apsaugą nuo duomenų atkūrimo: vieni metodai apsaugo tik nuo paprasto atkūrimo per standartinę naudotojo sąsają, o kiti siekia padaryti duomenų atkūrimą neįmanomą net taikant pažangesnes priemones [10]. Paskirstytoje infrastruktūroje šis skirtumas ypač svarbus, nes sistemos savininkas paprastai nekontroliuoja visų fizinio saugojimo sluoksnių, todėl ne visi tradiciniai sanitarizavimo metodai gali būti praktiškai įgyvendinti [10].

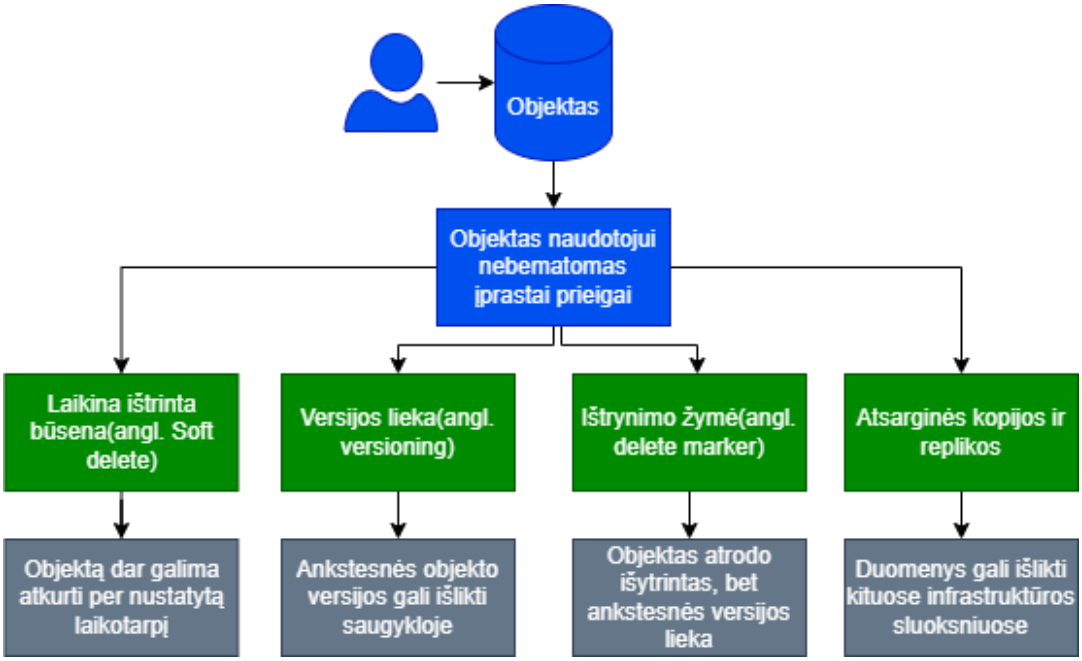
Toliau nagrinėjami pagrindiniai duomenų ištrynimo metodai ir vertinamas jų tinkamumas debesijos bei virtualiajai infrastruktūrai.

1.2.1. Loginis duomenų ištrynimas

Loginis duomenų ištrynimas grindžiamas standartine objekto, failo ar įrašo pašalinimo operacija per taikomosios programos, operacinės sistemos ar saugojimo paslaugos valdymo sąsają. Tokiu atveju duomenys tampa nebepasiekiami įprastam naudojimui, tačiau pats rezultatas priklauso nuo to, kaip sistema realizuoja saugojimo, versijų kūrimo, atkūrimo ir laikymo politikos mechanizmus. NIST SP 800-88 nurodo, kad išvalymas (angl. *clear*) yra sanitarizavimo metodas, taikantis logines priemones naudotojui prieinamose saugojimo vietose ir apsaugantis nuo paprastų, neinvasinių atkūrimo būdų [10].

Vis dėlto loginis ištrynimas savaime nereiškia, kad duomenys tampa negrįžtamai pašalinti. Užtikrintas duomenų ištrynimas (angl. *assured deletion*) turi užtikrinti, kad duomenys taptų nebeatkuriami arba nebeprieinami, o vien loginis pašalinimas tokio rezultato dažnai nesuteikia [5]. Tai aiškiai matyti iš realių saugyklų veikimo. „Microsoft Azure Blob Storage“ atveju objektų laikinas išsaugojimas po ištrynimo (angl. *blob soft delete*) leidžia tam tikrą laiką atkurti ištrintą objektą, o versijų kūrimo mechanizmas išsaugo ankstesnes jo versijas [6], [7]. AWS „S3“ aplinkoje paprastas ištrynimo veiksmas (angl. *DELETE*) konteineryje, kuriame įjungtas versijų kūrimas, sukuria

ištrynimo žymę (angl. *delete marker*), todėl objektas atrodo ištrintas, nors ankstesnės jo versijos išlieka saugykloje [8].



2 pav. Loginio duomenų ištrynimo veikimo logika paskirstytoje infrastruktūroje

Kaip matyti 2 paveiksle, loginio ištrynimo veiksmas dažnai pašalina tik aktyvią objekto būseną arba numatytąją prieigą prie jo, tačiau nebūtinai panaikina versijas, ištrynimo žymes ar laikinas atkūrimo būsenas.

Pagrindinis loginio ištrynimo privalumas yra jo paprastumas, mažas įgyvendinimo sudėtingumas ir geras pritaikomumas kasdienėms administravimo operacijoms. Šis metodas paprastai jau yra integruotas į saugojimo paslaugų valdymo sąsajas, todėl nereikalauja papildomos kriptografinės infrastruktūros [10]. Vis dėlto pagrindinis šio metodo trūkumas yra ribotas negrįžtamumo lygis. Paskirstytoje infrastruktūroje naudotojas nekontroliuoja fizinės laikmenos ir negali patikrinti, ar duomenys nebuvo išsaugoti kituose fiziniuose ar loginiuose sluoksniuose [1]. Todėl loginį ištrynimą tikslinga laikyti baziniu administraciniu veiksmu, tačiau nepakankamu pagrindiniu saugaus duomenų ištrynimo sprendimu [10].

1 lentelė. Loginio duomenų ištrynimo metodo vertinimas

Kriterijus	**Vertinimas**	**Paaiškinimas**
Negrįžtamumo lygis	Žemas	Loginis ištrynimas dažniausiai pašalina tik objekto matomumą arba prieigą prie jo, tačiau nebūtinai panaikina visas versijas, kopijas ar atsargines kopijas.
Pritaikomumas paskirstytoje infrastruktūroje	Aukštas	Metodas lengvai taikomas debesijos ir virtualiojoje infrastruktūroje, nes dažniausiai yra realizuojamas standartinėmis platformos priemonėmis.
Įgyvendinimo sudėtingumas	Žemas	Metodui nereikia papildomos kriptografinės infrastruktūros ar sudėtingo raktų valdymo.
Našumas	Aukštas	Ištrynimo veiksmas paprastai atliekamas greitai ir nereikalauja didelių skaičiavimo ar įvesties ir išvesties išteklių.

Audituojamumas	Vidutinis	Ištrynimo veiksmą galima registruoti, tačiau sunku įrodyti, kad visi duomenų egzemplioriai buvo negrįžtamai pašalinti.
Atitiktis reikalavimams	Žemas	Metodas gali būti tinkamas baziniam administraciniam valdymui, tačiau ne visada pakankamas saugiam ir negrįžtamam duomenų panaikinimui.

Vertinimo reikšmių paaiškinimas: šioje lentelėje vertinimas žemas reiškia, kad loginis ištrynimas silpnai užtikrina konkretų kriterijų, pavyzdžiui, negrįžtamumą ar atitiktį saugaus ištrynimo reikalavimams. Vertinimas vidutinis reiškia, kad kriterijus tenkinamas tik iš dalies, pavyzdžiui, ištrynimo veiksmą galima registruoti, tačiau sunku įrodyti galutinį rezultatą. Vertinimas aukštas reiškia, kad loginis ištrynimas gerai atitinka konkretų kriterijų, pavyzdžiui, yra lengvai pritaikomas ir pasižymi geru našumu.

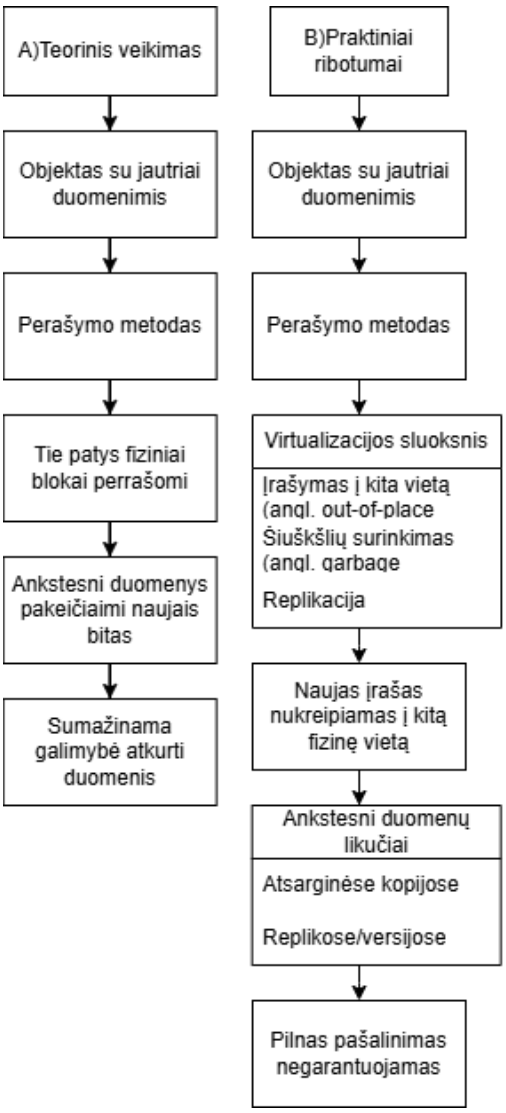
Kaip matyti 1 lentelėje, loginis duomenų ištrynimas yra paprastas ir lengvai pritaikomas metodas, tačiau jo negrįžtamumo lygis yra žemas. Todėl paskirstytoje infrastruktūroje jis gali būti naudojamas kaip administracinis veiksmas, bet nėra pakankamas pagrindinis saugaus duomenų ištrynimo sprendimas.

1.2.2. Perrašymu pagrįstas duomenų naikinimas

Perrašymu pagrįstas metodas remiasi principu, kad anksčiau saugykloje buvę duomenys pakeičiami kitais bitų modeliais, pavyzdžiui, nuliais, vienetais ar pseudoatsitiktinėmis reikšmėmis. Taip siekiama panaikinti galimybę atkurti pradinį turinį iš tų pačių fizinių saugojimo vietų. NIST SP 800-88 Rev. 2 šį metodą sieja su išvalymo tipo sanitarizavimo priemonėmis (angl. *clear*) [10].

Tradicinėje infrastruktūroje, ypač magnetinių diskų atveju, perrašymas ilgą laiką buvo laikomas praktišku sprendimu, kai organizacija pati valdo fizinę laikmeną [10]. Tačiau šio metodo veiksmingumas priklauso nuo prielaidos, kad sistema gali pasiekti tas pačias fizines vietas, kuriose anksčiau buvo įrašyti duomenys. Būtent ši prielaida tampa problemiška šiuolaikinėse saugyklose, ypač kietakūniuose diskuose (angl. *Solid-State Drive, SSD*) ir virtualiojoje paskirstytoje infrastruktūroje, kur realus fizinis įrašymo adresas dažnai nėra tiesiogiai valdomas naudotojo [11].

NIST SP 800-88 Rev. 2 pažymi, kad dėl atsarginių blokų ir dėvėjimosi balansavimo (angl. *wear leveling*) naudotojui gali būti neįmanoma pasiekti visų ankstesnių vietų, kuriose buvo saugoti jautrūs duomenys [10]. Šią problemą patvirtina ir kiti tyrimai, rodantys, kad SSD architektūrose operacinės sistemos ar failų sistemos inicijuotas perrašymas nebūtinai sutampa su tikru ankstesnių fizinių puslapių pakeitimu [11], [12].



3 pav. Perrašymu pagrįsto duomenų naikinimo veikimo ir ribotumų schema

Kaip matyti 3 paveiksle, perrašymu pagrįstas duomenų naikinimas teoriškai remiasi prielaida, kad sistema gali perrašyti tas pačias fizines vietas, kuriose anksčiau buvo saugomi duomenys. Tačiau SSD ir paskirstytoje infrastruktūroje ši prielaida dažnai nėra tenkinama, todėl ankstesni duomenų likučiai gali išlikti kituose fiziniuose ar loginiuose sluoksniuose.

Pagrindinis perrašymo metodo privalumas yra aiškus jo veikimo principas ir galimybė tam tikrais atvejais išsaugoti laikmeną pakartotiniam naudojimui [10]. Tačiau paskirstytoje infrastruktūroje šio metodo trūkumai tampa esminiai: naudotojas nekontroliuoja fizinės laikmenos, negali patvirtinti, kad perrašymas pasiekė visas ankstesnes duomenų saugojimo vietas, o papildomi mechanizmai, tokie kaip versijų kūrimas, replikacija, atsarginės kopijos ar objektų užrakinimas (angl. *Object Lock*), dar labiau sumažina metodo patikimumą [5], [6], [9]. Be to, perrašymas yra lėtas ir imlus ištekliams, nes reikalauja realių įvesties ir išvesties operacijų (angl. *input/output, I/O*) [12].

Dėl šių priežasčių perrašymu pagrįstas metodas šiame darbe vertinamas kaip svarbus palyginamasis atskaitos taškas, tačiau ne kaip tinkamas pagrindinis sprendimas paskirstytai infrastruktūrai.

2 lentelė. Perrašymu pagrįsto metodo vertinimas

Kriterijus	**Vertinimas**	**Paaiškinimas**
Negrįžtamumo lygis	Vidutinis	Tradicinėje infrastruktūroje perrašymas gali sumažinti duomenų atkūrimo galimybę, tačiau paskirstytoje aplinkoje jo veiksmingumas nėra visiškai užtikrinamas.
Pritaikomumas paskirstytoje infrastruktūroje	Žemas	Metodas reikalauja fizinės arba žemo lygmens saugyklos kontrolės, kurios naudotojas debesijos ar virtualiojoje infrastruktūroje dažniausiai neturi.
Įgyvendinimo sudėtingumas	Aukštas	Reikia papildomų sanitarizavimo priemonių, žemo lygmens saugyklos valdymo ir galimybės pasiekti fizinius duomenų blokus.
Našumas	Žemas	Perrašymas reikalauja daug įvesties ir išvesties operacijų, todėl didesnių duomenų kiekių atveju yra lėtas ir imlus ištekliams.
Audituojamumas	Vidutinis	Perrašymo veiksmą galima registruoti, tačiau sunku įrodyti, kad buvo paveiktos visos fizinės ar loginės duomenų kopijos.
Atitiktis reikalavimams	Žemas	Paskirstytoje infrastruktūroje metodas ribotai atitinka saugaus duomenų ištrynimo reikalavimus, nes nėra garantijos, kad pasiektos visos duomenų saugojimo vietos.

Vertinimo reikšmių paaiškinimas: šioje lentelėje vertinimas žemas reiškia, kad perrašymu pagrįstas metodas yra silpnai pritaikomas paskirstytai, debesijos ar virtualiajai infrastruktūrai. Vertinimas vidutinis reiškia, kad metodas gali būti veiksmingas tam tikromis sąlygomis, dažniausiai tradicinėje fizinėje infrastruktūroje, tačiau jo rezultatas nėra visiškai patikimas paskirstytoje aplinkoje. Vertinimas aukštas, vertinant įgyvendinimo sudėtingumą, reiškia ne privalumą, o tai, kad metodui įgyvendinti reikia sudėtingų techninių sąlygų, fizinės laikmenos kontrolės ir papildomų sanitarizavimo priemonių.

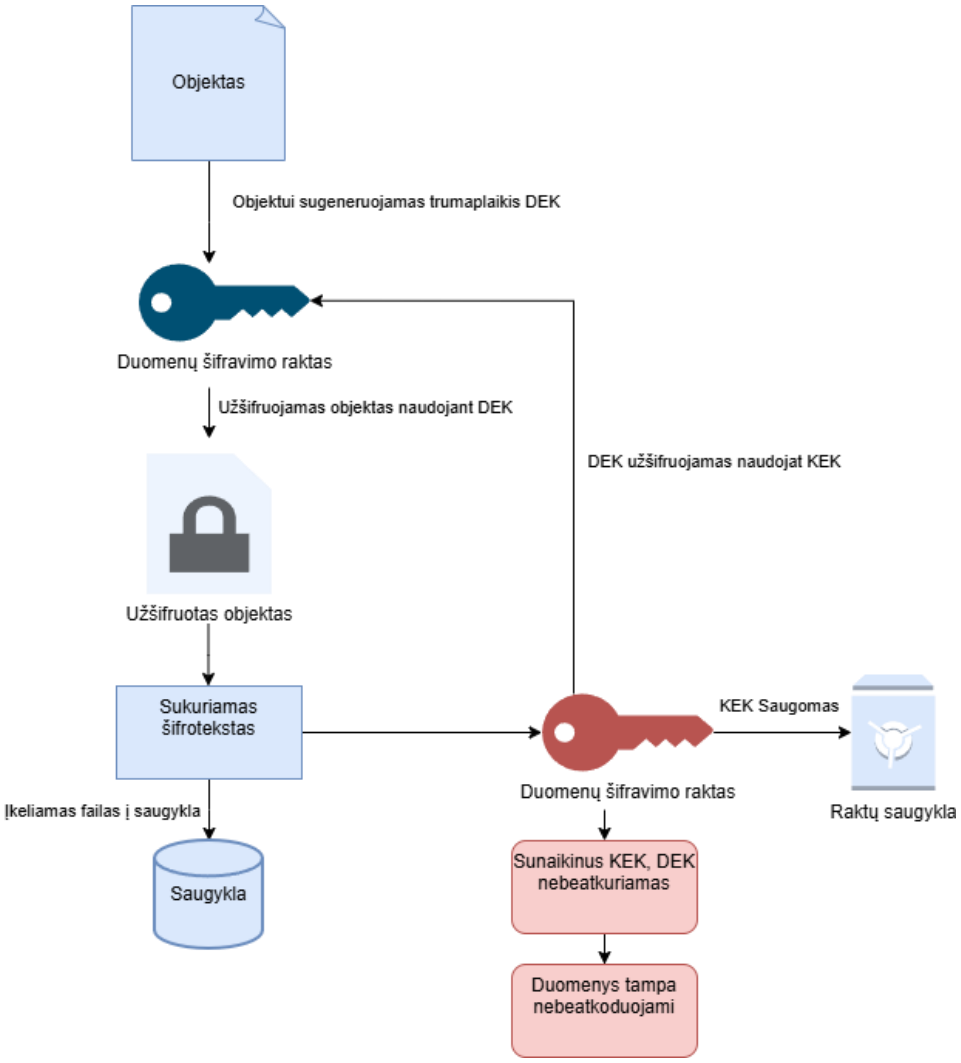
Kaip matyti 2 lentelėje, perrašymu pagrįstas metodas gali būti naudingas tradicinėje infrastruktūroje, tačiau paskirstytoje ir virtualiojoje aplinkoje jo pritaikomumas yra žemas. Pagrindinė problema yra fizinės laikmenos kontrolės nebuvimas.

1.2.3. Kriptografinis duomenų naikinimas

Kriptografinis duomenų naikinimas (angl. *crypto-shredding*, *cryptographic erase*) grindžiamas principu, kad duomenys pirmiausia užšifruojami, o vėliau sunaikinamas arba nebepasiekiamas padaromas jiems iššifruoti reikalingas raktas. Tokiu atveju pats šifruotas tekstas gali likti saugykloje, tačiau be rakto jo turinys tampa praktiškai nebeatkuriamas ir nebeiššifruojamas. NIST SP 800-88 Rev. 2 aiškiai nurodo, kad kriptografinis ištrynimas yra galutinio pašalinimo (angl. *purge*) priemonė, grindžiama raktų sanitarizavimu [10].

Šio metodo esmė yra ne fizinis visų kopijų pašalinimas, o prieigos prie duomenų panaikinimas per raktų valdymą. Dėl to kriptografinis duomenų naikinimas yra ypač tinkamas paskirstytoje, debesijos ir virtualiojoje infrastruktūroje, kur sistemos savininkas paprastai neturi fizinės laikmenos kontrolės [10]. Mokslinėje literatūroje ši kryptis laikoma viena realiausių paskirstytoms architektūroms: užšifruoti duomenys gali likti saugykloje, tačiau pašalinus raktą jie tampa nebeprieinami [5].

Kriptografinio duomenų naikinimo pagrindą sudaro duomenų šifravimo rakto (angl. *Data Encryption Key, DEK*) ir raktų šifravimo rakto (angl. *Key Encryption Key, KEK*) hierarchija. DEK naudojamas konkretaus objekto turiniui šifruoti, o KEK skirtas DEK apsaugoti ir prieigai prie jo valdyti [13]. Tokia schema realizuojama taikant voko šifravimą (angl. *envelope encryption*): objektas šifruojamas DEK, o DEK užšifruojamas KEK ir saugomas kartu su objekto metaduomenimis arba atskirame raktų valdymo sluoksnyje [14], [16].



4 pav. DEK ir KEK pagrindu veikiančio voko šifravimo schema

Kaip matyti 4 paveiksle, voko šifravimo modelyje pats duomenų turinys šifruojamas DEK, o šis DEK apsaugomas KEK. Tokia schema leidžia saugoti šifruotą tekstą atskirai nuo rakto medžiagos ir sudaro prielaidas duomenų turinio nebeatkuriamumui bei nebeiššifruojamumui sunaikinus KEK.

Pagrindinis kriptografinio duomenų naikinimo privalumas yra aukštas negrįžtamumo lygis net tada, kai sistemos savininkas nekontroliuoja fizinės laikmenos. Kitas svarbus privalumas yra geras suderinamumas su paskirstyta architektūra: net jei objektas dėl versijų kūrimo, replikacijos ar atsarginių kopijų išlieka saugykloje, sunaikinus KEK jo turinys tampa nebeiššifruojamas ir praktiškai nebeatkuriamas [22]. Be to, šis metodas yra našus, nes paprastai nereikalauja fiziškai perrašyti visų duomenų kopijų.

Vis dėlto šis metodas turi ir ribotumų. Jo veiksmingumas priklauso nuo tinkamai suprojektuoto raktų valdymo, aiškios DEK ir KEK hierarchijos, prieigos kontrolės ir audituojamų rakto sunaikinimo procedūrų. Taip pat svarbu pažymėti, kad kriptografinis duomenų naikinimas negarantuoja fizinio duomenų egzemplioriaus pašalinimo iš laikmenos – jis užtikrina, kad be reikalingų raktų duomenų turinys tampa nebeiššifruojamas ir praktiškai nebeatkuriamas. Tačiau būtent toks rezultatas paskirstytoje infrastruktūroje yra praktiškai svarbiausias. Atsižvelgiant į tai, šiame darbe kriptografinis duomenų naikinimas vertinamas kaip pagrindinis metodas, nes jis geriausiai atitinka paskirstytos infrastruktūros ribojimus ir suteikia tvirčiausią pagrindą audituojamam bei atsekamam duomenų ištrynimo procesui [10], [13], [16]. Kartu svarbu pažymėti, kad šio metodo veiksmingumas tiesiogiai priklauso nuo kriptografinių raktų valdymo modelio, todėl tolesnėje analizėje ypatingas dėmesys skiriamas būtent raktų hierarchijai, jų saugojimui, išvedimui ir sunaikinimo logikai.

3 lentelė. Kriptografinio duomenų naikinimo metodo vertinimas

Kriterijus	**Vertinimas**	**Paaiškinimas**
Negrįžtamumo lygis	Aukštas	Sunaikinus arba padarius nebepasiekiamą raktą, nebeįmanoma atkurti DEK ir iššifruoti duomenų.
Pritaikomumas paskirstytoje infrastruktūroje	Aukštas	Metodas nereikalauja fizinės laikmenos kontrolės, todėl gerai tinka debesijos, objektinei ir virtualiajai infrastruktūrai.
Įgyvendinimo sudėtingumas	Vidutinis	Reikia tinkamai suprojektuotos DEK ir KEK hierarchijos, raktų saugojimo, išvedimo ir sunaikinimo logikos.
Našumas	Aukštas	Metodui nereikia perrašyti visų duomenų kopijų, todėl raktų sunaikinimas arba atkuriamumo panaikinimas gali būti atliekamas greitai.
Audituojamumas	Aukštas	Galima registruoti raktų generavimo, apvyniojimo, išvyniojimo ir sunaikinimo veiksmus.
Atitiktis reikalavimams	Aukštas	Metodas gerai dera su paskirstytos infrastruktūros ribojimais ir saugaus duomenų valdymo principais.

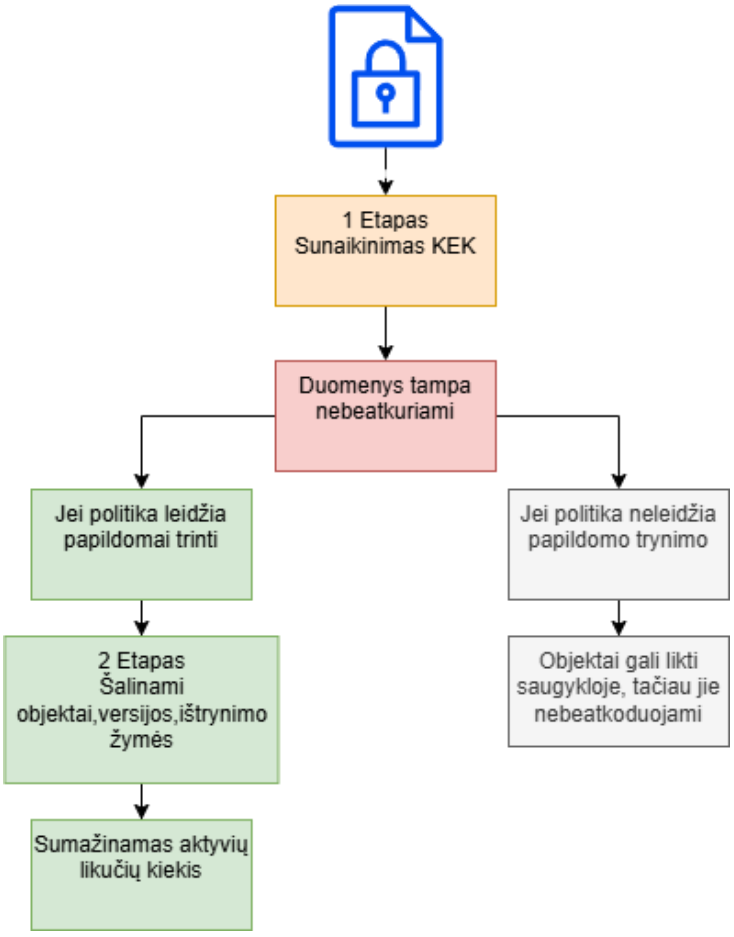
Vertinimo reikšmių paaiškinimas: šioje lentelėje vertinimas žemas reikštų, kad kriptografinis metodas silpnai tenkina konkretų kriterijų, tačiau šiuo atveju toks vertinimas netaikomas, nes metodas gerai atitinka paskirstytos infrastruktūros poreikius. Vertinimas vidutinis reiškia, kad kriterijus įgyvendinamas, tačiau reikalauja papildomo projektavimo, pavyzdžiui, aiškios DEK ir KEK hierarchijos bei raktų valdymo logikos. Vertinimas aukštas reiškia, kad metodas gerai užtikrina konkretų kriterijų, nes duomenų turinio nebeatkuriamumas ir nebeiššifruojamumas pasiekiamas sunaikinant raktus arba panaikinant jų atkuriamumą.

Kaip matyti 3 lentelėje, kriptografinis duomenų naikinimas yra tinkamiausias metodas paskirstytai infrastruktūrai, nes leidžia pasiekti praktinį duomenų turinio nebeatkuriamumą ir nebeiššifruojamumą net tada, kai užšifruotas objektas dar išlieka saugykloje.

1.2.4. Hibridinis duomenų naikinimo metodas

Hibridinis duomenų naikinimo metodas grindžiamas dviem nuosekliais veiksmais: pirmiausia kriptografiškai panaikinama prieiga prie duomenų, sunaikinant raktą arba padarant jį nebepasiekiamą, o po to, kiek leidžia infrastruktūros architektūra ir saugojimo politika, atliekamas papildomas loginis arba galutinis objektų pašalinimas iš aktyvios saugyklos [10].

Toks požiūris leidžia siekti dviejų tikslų: greitai panaikinti duomenų iššifravimo galimybę ir kartu sumažinti aktyvioje infrastruktūroje likusių objektų, jų versijų ar metaduomenų kiekį. Tai ypač svarbu paskirstytoje infrastruktūroje, kur vien papildomas objektų pašalinimas ne visada reiškia galutinį informacijos išnykimą. „Microsoft Azure“ nekeičiamumo politikos (angl. *immutability policies*), teisiniai sulaikymai (angl. *legal holds*), taip pat AWS objektų užrakinimas (angl. *Object Lock*) gali riboti ar atidėti galutinį pašalinimą [9], [27]. Todėl tokiose aplinkose pirminis rezultatas dažniausiai yra kriptografiškai pasiekiamas duomenų turinio nebeiššifruojamumas ir praktinis nebeatkuriamumas.



5 pav. Hibridinio duomenų naikinimo veikimo schema

Kaip matyti 5 paveiksle, hibridinis metodas susideda iš dviejų etapų: pirmiausia panaikinama duomenų iššifravimo galimybė, sunaikinant arba padarant neprieinamą reikiamą raktą, o po to, kiek leidžia infrastruktūros politika, atliekamas papildomas objektų, versijų ar metaduomenų pašalinimas iš aktyvios saugyklos.

Pagrindinis šio metodo privalumas yra tai, kad jis sujungia kriptografinio naikinimo negrįžtamumą ir papildomą aktyvių likučių mažinimą. Jis taip pat suteikia aiškesnį veiksmų sekos pagrindimą audito ar atitikties požiūriu, nes galima registruoti tiek rakto sunaikinimą, tiek papildomą objektų pašalinimą [5], [10].

Vis dėlto hibridinis metodas yra sudėtingesnis už grynąjį kriptografinį naikinimą, nes reikalauja suderinti raktų gyvavimo ciklo valdymą, objektų būsenų analizę ir saugojimo politikų apribojimus. Be to, papildomas objektų pašalinimas ne visada gali būti atliktas iš karto dėl versijų kūrimo,

ištrynimo žymių, teisinio sulaikymo ar nekeičiamumo politikų [8], [9], [27]. Dėl to šiame darbe hibridinis metodas vertinamas kaip praktiškai naudingas papildomas modelis, bet ne kaip pagrindinė sprendimo ašis.

4 lentelė. Hibridinio duomenų naikinimo metodo vertinimas

Kriterijus	**Vertinimas**	**Paaiškinimas**
Negrįžtamumo lygis	Aukštas	Rakto sunaikinimas arba jo atkuriamumo panaikinimas suteikia stiprų pagrindą duomenų turinio nebeiššifruojamumui ir praktiniam nebeatkuriamumui.
Pritaikomumas paskirstytoje infrastruktūroje	Aukštas	Metodas gali būti taikomas paskirstytoje infrastruktūroje, tačiau papildomas objektų pašalinimas priklauso nuo versijų kūrimo, nekeičiamumo ir saugojimo politikų.
Įgyvendinimo sudėtingumas	Aukštas	Reikia suderinti raktų sunaikinimą, objektų šalinimą, versijų valdymą, saugojimo politikų apribojimus ir audito veiksmus.
Našumas	Vidutinis	Kriptografinė dalis yra greita, tačiau papildomas objektų ar versijų šalinimas gali padidinti sistemos apkrovą.
Audituojamumas	Aukštas	Galima registruoti tiek rakto sunaikinimą, tiek papildomus objektų ar metaduomenų pašalinimo veiksmus.
Atitiktis reikalavimams	Aukštas	Metodas gerai dera su saugaus valdymo logika, jei yra tinkamai atsižvelgiama į infrastruktūros politikų ribojimus.

Vertinimo reikšmių paaiškinimas: šioje lentelėje vertinimas žemas reikštų, kad hibridinis metodas silpnai atitinka konkretų kriterijų, tačiau šiame vertinime toks lygis nenaudojamas. Vertinimas vidutinis reiškia, kad kriterijus tenkinamas iš dalies, pavyzdžiui, našumas gali sumažėti dėl papildomo objektų ar versijų šalinimo. Vertinimas aukštas reiškia, kad metodas gerai užtikrina konkretų kriterijų, tačiau įgyvendinimo sudėtingumo atveju aukštas vertinimas reiškia didesnį realizacijos sudėtingumą, o ne metodo pranašumą.

Kaip matyti 4 lentelėje, hibridinis metodas gali sustiprinti bendrą saugaus duomenų naikinimo rezultatą. Vis dėlto dėl didesnio įgyvendinimo sudėtingumo jis šiame darbe vertinamas kaip papildomas, o ne pagrindinis sprendimo modelis.

1.2.5. Esamų metodų palyginimas

Siekiant objektyviai įvertinti loginį duomenų ištrynimą, perrašymu pagrįstą duomenų naikinimą, kriptografinį duomenų naikinimą ir hibridinį metodą, šiame darbe taikomi šeši palyginimo kriterijai: negrįžtamumo lygis, pritaikomumas paskirstytoje infrastruktūroje, įgyvendinimo sudėtingumas, našumas, audituojamumas ir atitiktis reikalavimams [10]. Toks požiūris dera su NIST SP 800-88, kuriame pabrėžiama, kad tinkamo sanitarizavimo metodo pasirinkimas priklauso ne tik nuo techninės galimybės ištrinti duomenis, bet ir nuo konfidencialumo reikalavimų, laikmenos pobūdžio, sąnaudų, trukmės ir rizikos mažinimo efektyvumo [10].

Pagal negrįžtamumą silpniausias yra loginis ištrynimas, nes jis dažniausiai panaikina tik objekto matomumą arba numatytąją prieigą, bet ne visas jo versijas ar kopijas [5]. Perrašymas teoriniu požiūriu gali suteikti stipresnį rezultatą, tačiau jo efektyvumas priklauso nuo laikmenos architektūros ir fizinės kontrolės [10], [11]. Kriptografinis ir hibridinis metodai suteikia aukščiausią praktiškai pasiekiamą negrįžtamumo lygį, nes duomenų neprieinamumą sieja su rakto sunaikinimu [10].

Pagal pritaikomumą paskirstytoje infrastruktūroje lengviausiai realizuojamas loginis ištrynimas, tačiau jis išlieka funkciniu požiūriu ribotas [6]–[8]. Perrašymas yra prasčiausiai pritaikomas, nes reikalauja fizinės arba bent žemo lygmens saugyklos kontrolės [10], [11]. Kriptografinis duomenų naikinimas geriausiai dera su paskirstytos architektūros logika, nes nepriklauso nuo fizinės laikmenos kontrolės [5]. Hibridinis metodas taip pat tinka paskirstytai infrastruktūrai, tačiau jo papildoma fazė priklauso nuo konkrečios platformos politikų [8], [9], [27].

Pagal įgyvendinimo sudėtingumą paprasčiausias yra loginis ištrynimas, o sudėtingiausias – hibridinis metodas. Pagal našumą stipriausi yra loginis ir kriptografinis duomenų naikinimas, nes jiems nereikia didelio masto fizinio perrašymo. Pagal audituojamumą stipriausi yra kriptografinis ir hibridinis metodai, nes jų atveju galima registruoti rakto gyvavimo ciklo ir naikinimo veiksmus [13].

Toliau visi nagrinėti duomenų naikinimo metodai palyginami pagal tuos pačius kriterijus, taikant vienodą trijų lygių kokybinę vertinimo skalę. Toks pateikimas leidžia aiškiai palyginti metodų tinkamumą saugiam duomenų ištrynimui paskirstytoje, debesijos ir virtualiojoje infrastruktūroje.

5 lentelė. Esamų duomenų naikinimo metodų palyginimas

Vertinimo kriterijus	**Loginis ištrynimas**	**Perrašymas**	**Kriptografinis duomenų naikinimas**	**Hibridinis metodas**
Negrįžtamumo lygis	Žemas	Vidutinis	Aukštas	Aukštas
Pritaikomumas paskirstytoje infrastruktūroje	Vidutinis	Žemas	Aukštas	Aukštas
Įgyvendinimo sudėtingumas	Žemas	Aukštas	Vidutinis	Aukštas
Našumas	Aukštas	Žemas	Aukštas	Vidutinis
Audituojamumas	Vidutinis	Vidutinis	Aukštas	Aukštas
Atitiktis reikalavimams	Žemas	Žemas	Aukštas	Aukštas
Tinkamumas pagrindiniam sprendimui	Žemas	Žemas	Aukštas	Vidutinis

Vertinimo reikšmių paaiškinimas: šioje lentelėje vertinimas žemas reiškia, kad metodas turi esminių ribojimų pagal konkretų kriterijų ir negali būti laikomas pakankamu pagrindiniu sprendimu. Vertinimas vidutinis reiškia, kad metodas gali būti taikomas tam tikrais atvejais, tačiau jo veiksmingumas priklauso nuo papildomų sąlygų, pavyzdžiui, infrastruktūros politikų, fizinės laikmenos kontrolės ar raktų valdymo kokybės. Vertinimas aukštas reiškia, kad metodas gerai atitinka konkretų kriterijų ir yra tinkamas saugaus duomenų ištrynimo paskirstytoje infrastruktūroje poreikiams. Įgyvendinimo sudėtingumo kriterijuje žemas reiškia mažą sudėtingumą, o aukštas – didelį sudėtingumą.

Kaip matyti 5 lentelėje, loginis ištrynimas ir perrašymas turi praktinių privalumų, tačiau jų negrįžtamumo lygis arba pritaikomumas paskirstytoje infrastruktūroje yra nepakankami. Kriptografinis duomenų naikinimas pasižymi aukštu negrįžtamumo lygiu, aukštu pritaikomumu paskirstytoje infrastruktūroje ir geru audituojamumu, todėl šiame darbe vertinamas kaip tinkamiausias pagrindinis metodas. Hibridinis metodas taip pat pasižymi aukštais saugumo ir

atitikties vertinimais, tačiau dėl aukšto įgyvendinimo sudėtingumo jis labiau tinka kaip papildomas modelis.

Vertinant metodų tinkamumą, žemas reiškia, kad metodas turi esminių ribojimų ir negali būti laikomas pakankamu pagrindiniu sprendimu. Vidutinis reiškia, kad metodas gali būti taikomas tam tikrais atvejais, tačiau jo veiksmingumas priklauso nuo papildomų sąlygų. Aukštas reiškia, kad metodas gerai atitinka paskirstytos infrastruktūros ribojimus ir gali būti laikomas tinkamu saugaus duomenų ištrynimo sprendimu.

1.2.6. Poskyrio apibendrinimas

Atliktas metodų palyginimas parodė, kad nors loginis ištrynimas ir perrašymu pagrįstas duomenų naikinimas išlieka svarbūs tam tikruose praktiniuose scenarijuose, jų galimybės paskirstytoje infrastruktūroje yra ribotos. Loginis ištrynimas dažniausiai nepanaikina visų versijų, kopijų ar atkūrimo būsenų, o perrašymas stipriai priklauso nuo fizinės laikmenos savybių ir kontrolės, kurios sistemos savininkas tokiose aplinkose paprastai neturi [5], [11].

Kriptografinis duomenų naikinimas, paremtas duomenų šifravimo rakto (angl. *Data Encryption Key*, DEK) ir raktų šifravimo rakto (angl. *Key Encryption Key*, KEK) hierarchija bei voko šifravimu (angl. *envelope encryption*), geriausiai atitinka paskirstytos architektūros logiką, nes leidžia pasiekti duomenų turinio nebeiššifruojamumą ir praktinį nebeatkuriamumą net tada, kai fizinės duomenų kopijos dar kurį laiką išlieka infrastruktūroje [10], [13], [16]. Hibridinis metodas papildo šį požiūrį, tačiau dėl didesnio sudėtingumo ir priklausomybės nuo saugojimo politikų šiame darbe vertinamas kaip papildoma, o ne pagrindinė kryptis [8], [9], [10], [27].

Todėl šiame darbe kaip pagrindinė tolesnės analizės kryptis pasirenkamas kriptografinis duomenų naikinimas. Vis dėlto, kadangi šio metodo veiksmingumas tiesiogiai priklauso nuo to, kaip kuriami, saugomi, išvedami ir sunaikinami raktai, tolesnėje analizėje pagrindinis dėmesys skiriamas kriptografinių raktų valdymo metodams, kurie sudaro praktinį saugaus duomenų ištrynimo pagrindą.

1.3. Standartai ir metodikos

Saugaus duomenų ištrynimo problema negali būti vertinama vien techniniu požiūriu. Ji taip pat turi būti grindžiama teisiniais, norminiais ir metodiniais reikalavimais, kurie apibrėžia, kada duomenų pašalinimas laikomas pakankamu, kokie valdymo principai turi būti taikomi ir kaip turi būti organizuojamas raktų bei susijusių procesų valdymas. Dėl šios priežasties šiame skyriuje aptariami pagrindiniai norminiai ir metodiniai dokumentai, sudarantys teorinį pagrindą tolesnei kriptografinių raktų valdymo metodų analizei.

1.3.1. Bendrasis duomenų apsaugos reglamentas ir teisė būti pamirštam

Duomenų apsaugos požiūriu svarbią vietą užima teisė į duomenų ištrynimą, dar vadinama teise būti pamirštam. Europos duomenų apsaugos valdybos (EDPB) koordinuoto vykdymo ataskaitoje pažymima, kad teisė į ištrynimą pagal Bendrojo duomenų apsaugos reglamento (angl. *General Data Protection Regulation, GDPR*) 17 straipsnį nėra absoliuti ir gali būti įgyvendinama tik esant konkretiems teisiniams pagrindams, pavyzdžiui, kai asmens duomenys nebėra būtini tikslams, kuriems buvo surinkti, kai atšaukiamas sutikimas arba kai duomenys buvo tvarkomi neteisėtai [17]. Toje pačioje ataskaitoje nurodoma, kad egzistuoja ir išimčių, leidžiančių atsisakyti tenkinti ištrynimo

prašymą, pavyzdžiui, kai duomenų tvarkymas būtinas teisinei prievolei vykdyti, viešajam interesui ar archyvavimo tikslams [17].

Praktiniu požiūriu svarbu tai, kad teisės į ištrynimą įgyvendinimas dažnai susiduria su techniniais sunkumais. EDPB priede pažymima, kad kai kuriais atvejais duomenys ištrinami iš aktyvių sistemų, tačiau neištrinami iš atsarginių kopijų [18]. Tai rodo, kad vien formalus ištrynimo veiksmas dar nereiškia, jog duomenys tapo visiškai nebeatkuriami.

1.3.2. NIST SP 800-88 rekomendacijos

Techniniu požiūriu vienas svarbiausių norminių pagrindų saugiam duomenų ištrynimui yra NIST SP 800-88 Rev. 2. Šiame dokumente sanitarizavimas apibrėžiamas kaip procesas, kuriuo prieiga prie tikslinių duomenų laikmenoje padaroma neįmanoma esant tam tikram pastangų lygiui, o metodai skirstomi į išvalymą (angl. *clear*), galutinį pašalinimą (angl. *purge*) ir sunaikinimą (angl. *destroy*) [10]. Toks požiūris svarbus todėl, kad leidžia saugų duomenų ištrynimą vertinti ne pagal vieną komandą ar įrankį, bet pagal realų galimybės atkurti duomenis sumažinimą.

Šis standartas ypač svarbus todėl, kad jame kriptografinis ištrynimas (angl. *cryptographic erase*) aiškiai pristatomas kaip galutinio pašalinimo (angl. *purge*) kategorijos priemonė. NIST nurodo, kad esant tinkamoms sąlygoms ir taikant stiprią kriptografiją, integruotos šifravimo ir raktų valdymo galimybės gali būti naudojamos kaip aukšto užtikrintumo sanitarizavimo priemonė [10]. Tai tiesiogiai pagrindžia, kodėl šiame darbe kriptografinis duomenų naikinimas laikomas svarbia technine kryptimi, o pagrindinis dėmesys skiriamas raktų valdymo metodui, nuo kurio priklauso šio principo praktinis veiksmingumas.

1.3.3. NIST SP 800-132 ir NIST SP 800-57 raktų išvedimo bei valdymo principai

Kriptografinio duomenų naikinimo kontekste svarbus ne tik pats duomenų šifravimas, bet ir tai, kaip generuojami, saugomi, naudojami ir sunaikinami raktai. NIST SP 800-132 pateikia rekomendacijas, kaip iš slaptažodžių ar slaptafrazių išvesti kriptografinius raktus naudojant slaptažodžiu grindžiamas raktų išvedimo funkcijas. Šios rekomendacijos svarbios tais atvejais, kai raktų šifravimo raktas nėra tiesiogiai saugomas kaip ilgalaikis objektas, o išvedamas iš naudotojo paslapties ir papildomų parametrų, pavyzdžiui, druskos ir iteracijų skaičiaus [19]. Toks požiūris leidžia mažinti ilgalaikio rakto saugojimo poreikį, tačiau kartu iškelia reikalavimą užtikrinti pakankamą naudotojo paslapties stiprumą [19], [29].

NIST SP 800-57 pateikia bendruosius raktų valdymo principus per visą rakto gyvavimo ciklą. Šiame dokumente akcentuojama, kad raktų saugumas priklauso ne vien nuo pasirinkto algoritmo, bet ir nuo to, kaip raktai generuojami, paskirstomi, saugomi, naudojami, keičiami ir sunaikinami [13]. Kriptografinio duomenų naikinimo atveju tai ypač svarbu, nes praktinis duomenų turinio nebeiššifruojamumas ir nebeatkuriamumas tiesiogiai priklauso nuo to, ar raktų sunaikinimo procedūros yra patikimos, kontroliuojamos ir audituojamos [13].

Todėl NIST SP 800-132 ir NIST SP 800-57 šiame darbe sudaro svarbų metodinį pagrindą ne tik raktų išvedimui, bet ir visam jų gyvavimo ciklo valdymui. Šie dokumentai padeda pagrįsti, kodėl saugaus duomenų ištrynimo sprendime turi būti aiškiai apibrėžta, kaip raktai kuriami, kaip apsaugomi naudojimo metu ir kaip panaikinama galimybė juos atkurti pasibaigus duomenų gyvavimo ciklui.

1.3.4. ISO/IEC 27001 ir ISO/IEC 27002 nuostatos

Organizaciniu ir valdymo požiūriu svarbus pagrindas yra ISO/IEC 27001 ir ISO/IEC 27002 standartai. ISO/IEC 27001:2022 apibrėžia reikalavimus informacijos saugumo valdymo sistemai, kuri turi būti kuriama, įgyvendinama, palaikoma ir nuolat tobulinama [20]. ISO/IEC 27002:2022 pateikia informacijos saugumo kontrolės priemonių gaires ir gerąją praktiką, susijusią su prieigos valdymu, kriptografija, incidentų valdymu ir kitomis saugumo priemonėmis [21].

Šių standartų reikšmė šiam darbui yra ta, kad jie nenurodo vienos konkrečios duomenų ištrynimo priemonės, tačiau aiškiai parodo, jog saugus duomenų pašalinimas turi būti vertinamas kaip valdomas, atsekamas ir audituojamas procesas. Kitaip tariant, vien techninio veiksmo nepakanka – būtini aiškiai apibrėžti vaidmenys, kontrolės priemonės, procedūros ir dokumentavimas.

1.3.5. Poskyrio apibendrinimas

Apibendrinant galima teigti, kad aptarti standartai ir metodikos sudaro pakankamą norminį pagrindą tolesnei analizei. Bendrasis duomenų apsaugos reglamentas (angl. *General Data Protection Regulation, GDPR*) ir EDPB praktika parodo, kad duomenų ištrynimas turi būti ne tik formaliai inicijuotas, bet ir realiai suvaldytas, įskaitant problemą, susijusią su atsarginėmis kopijomis [17], [18]. NIST SP 800-88 Rev. 2 suteikia techninę logiką, pagal kurią saugus ištrynimas turi būti vertinamas pagal nebeatkuriamumą, o kriptografinis ištrynimas pripažįstamas tinkama galutinio pašalinimo (angl. *purge*) priemone [10]. NIST SP 800-57 ir NIST SP 800-132 suteikia metodinį pagrindą raktų valdymui ir raktų išvedimui, o ISO/IEC 27001 ir ISO/IEC 27002 parodo, kad ši problema turi būti suprantama kaip valdomas ir audituojamas saugumo procesas [13], [19], [21]. Ši norminė bazė leidžia pereiti prie konkrečių kriptografinių raktų valdymo sprendimų analizės, taikomų saugiam duomenų ištrynimui.

1.4. Esami saugaus duomenų ištrynimo sprendimai

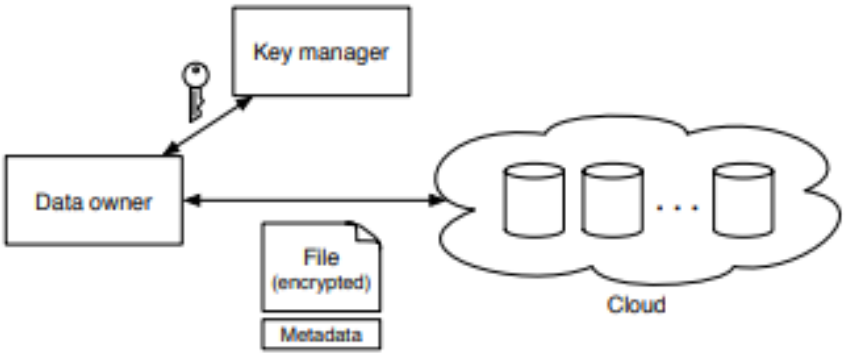
Ankstesniame skyriuje nustatyta, kad loginis ištrynimas ir perrašymu pagrįstas naikinimas paskirstytoje infrastruktūroje turi reikšmingų ribojimų, o kriptografinis duomenų naikinimas geriausiai dera su tokios architektūros logika. Tačiau vien teorinio metodo pasirinkimo nepakanka – būtina įvertinti, kokie konkretūs sprendimai jau yra pasiūlyti mokslinėje literatūroje ir kokius techninius principus jie taiko. Dėl šios priežasties šiame skyriuje analizuojami esami saugaus duomenų ištrynimo sprendimai, akcentuojant raktų atskyrimą, smulkiai valdomą kriptografinį naikinimą, verifikuojamą ištrynimą ir raktų šifravimo rakto (angl. *Key Encryption Key, KEK*) valdymo kryptis.

Mokslinėje literatūroje šiuos sprendimus vienija tai, kad jie nebesiremia fiziniu visų kopijų ištrynimu. Vietoje to duomenų apsauga ir ištrynimas siejami su šifravimu, raktų atskyrimu, politikomis, atributais ar ištrynimo įrodymo mechanizmais. Tokia kryptis ypač svarbi paskirstytoje infrastruktūroje, nes sistemos savininkas paprastai negali patikimai kontroliuoti fizinių kopijų, jų replikavimo ar atsarginių kopijų. Dėl to šiame darbe ypatingas dėmesys skiriamas ne tik bendriems ištrynimo sprendimams, bet ir tiems architektūriniams principams, kurie leidžia valdyti duomenų turinio nebeiššifruojamumą ir praktinį nebeatkuriamumą per raktų gyvavimo ciklą.

1.4.1. Raktų atskyrimu pagrįsti sprendimai

Vienas ankstyviausių sprendimų yra FADE (angl. *File Assured Deletion*). FADE autoriai jį apibrėžia kaip saugią debesijos saugyklos perdangą (angl. *secure overlay cloud storage system*), kuri užtikrina užtikrintą failų ištrynimą (angl. *assured deletion*) ir gali veikti esamų saugyklų pagrindu nekeičiant jų infrastruktūros. Esminė FADE idėja yra raktų ir duomenų atskyrimas: užšifruoti duomenys lieka trečiosios šalies saugykloje, o šifravimo raktai tvarkomi atskirame raktų valdymo sluoksnyje [22].

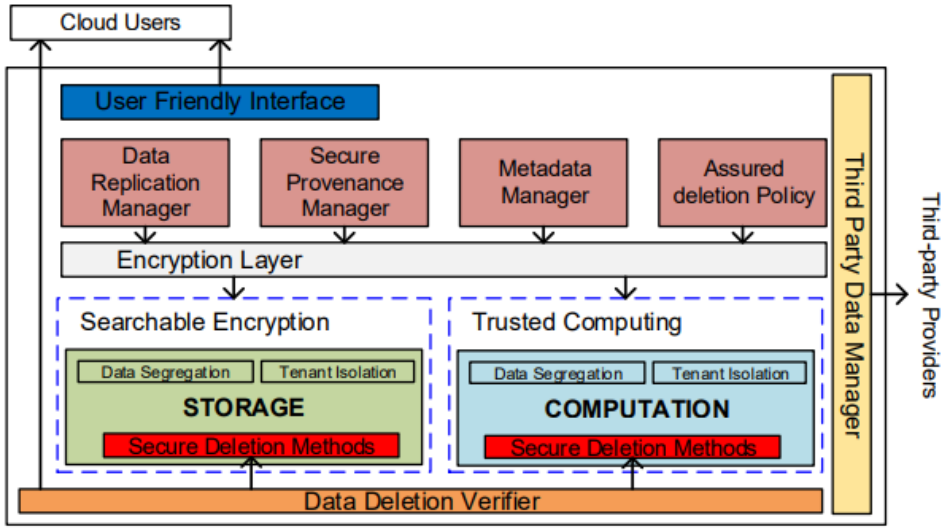
FADE sprendimas atsirado kaip atsakas į problemą, kad saugojimo paslaugų teikėjai gali nepašalinti visų failo kopijų, nes palaiko atsargines kopijas ar paskirsto duomenis tarp kelių vietų. Dėl to FADE architektūra orientuojama ne į fizinių kopijų paiešką, o į raktų valdymą. Svarbu ir tai, kad šis modelis išplečia laiku pagrįstą ištrynimą į politikomis grindžiamą failų naikinimą: kai politika nustoja galioti, su ja susieti raktai panaikinami, todėl užšifruotas failas tampa nebeprieinamas [22].



6 pav. FADE (angl. *File Assured Deletion*) architektūra [22]

Kaip matyti 6 paveiksle, FADE architektūroje duomenų savininkas užšifruoja failą ir perduoda užšifruotą objektą bei metaduomenis į saugyklą, o raktų valdymas realizuojamas atskirame raktų valdymo sluoksnyje. Tokia schema leidžia pasiekti duomenų nebeprieinamumą panaikinus prieigą prie rakto, net jei pats užšifruotas failas išlieka saugykloje.

Vėlesnėje literatūroje FADE tipo sprendimai vertinami kaip svarbi užtikrinto duomenų ištrynimo kryptis, nes parodo, kad net nežinant visų fizinių kopijų vietos galima pasiekti praktinį negrįžtamumą panaikinant prieigą prie rakto. Vis dėlto šių modelių ribotumas yra tas, kad jie turi būti pakankamai tiksliai suprojektuoti, kad palaikytų atrankinį ištrynimą, o ne verstų naikinti per platų duomenų rinkinį vien dėl bendro rakto.



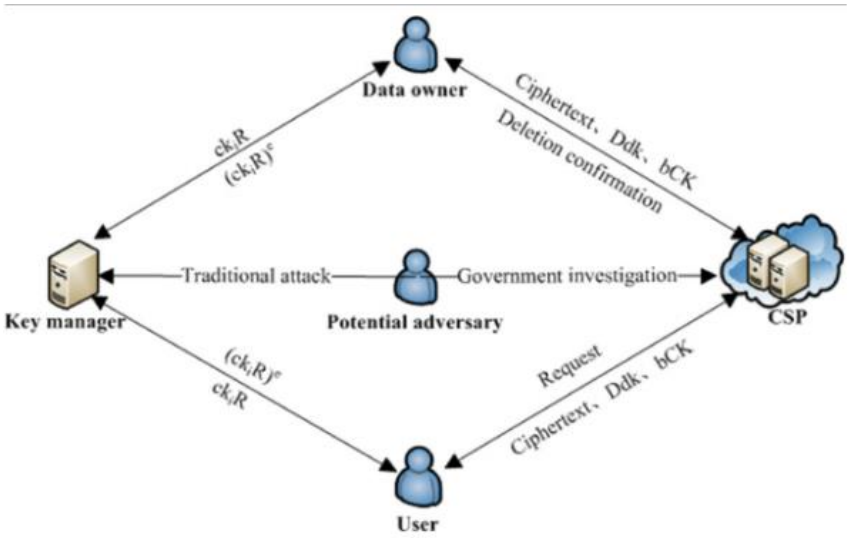
7 pav. Užtikrinto duomenų ištrynimo (angl. *assured deletion*) konceptualioji architektūra [5]

Kaip matyti 7 paveiksle, užtikrinto duomenų naikinimo sprendimai paprastai sudaryti iš kelių tarpusavyje susijusių sluoksnių: naudotojo sąsajos, metaduomenų ir politikų valdymo, šifravimo sluoksnio, saugojimo bei skaičiavimo komponentų ir ištrynimo patikros mechanizmo. Tai rodo, kad saugus duomenų naikinimas nėra pavienė trynimo operacija, bet kompleksinis procesas, apimantis prieigos kontrolę, raktų valdymą ir ištrynimo patikrą.

1.4.2. Smulkiai valdomo kriptografinio naikinimo sprendimai

Tolesnė saugaus duomenų ištrynimo sprendimų raida siejama su smulkiai valdomo naikinimo (angl. *fine-grained deletion*) poreikiu. Ramokapane ir bendraautoriai tarp svarbiausių užtikrinto duomenų ištrynimo reikalavimų išskiria galimybę pašalinti tik konkrečius tikslinius duomenis, nepaveikiant kitų to paties naudotojo ar nuomininko duomenų [5]. Toks požiūris leidžia tiksliau valdyti duomenų gyvavimo ciklą ir sumažina operacijų sąnaudas didelio masto sistemose.

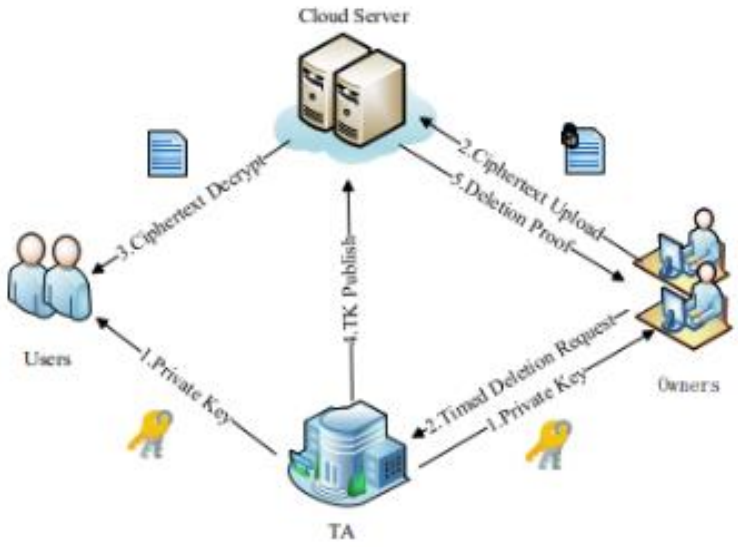
Viena šios krypties realizacijų yra ADAA (angl. *Fine-grained Assured Data Deletion Scheme Based on Attribute Association*). Šio modelio esmė – duomenų prieinamumą susieti su atributais, o naikinimą realizuoti per atributų arba susijusių raktų atšaukimą. Tokiu atveju tik tie naudotojai, kurių atributai atitinka prieigos struktūrą, gali pasiekti duomenis, o panaikinus atitinkamus atributus ar raktus, duomenys tampa nebeprieinami [23].



8 pav. ADAA sistemos modelis [23]

Kaip matyti 8 paveiksle, ADAA modelyje duomenų savininkas, raktų valdytojas, naudotojas ir debesijos paslaugų teikėjas sudaro tarpusavyje susietą sistemą, kurioje duomenų prieinamumas siejamas su atributais, o ištrynimo procesas papildomas patvirtinimo mechanizmu.

Kita svarbi kryptis yra ATDD (angl. *Fine-Grained Assured Time-Sensitive Data Deletion*). Ši schema duomenų prieinamumą papildo laiko sąlyga: pasibaigus duomenų savininko nustatytam galiojimo laikui, autorizuoti naudotojai nebegali gauti prieigos prie duomenų, o duomenų savininkui grąžinamas patvirtinimo duomuo, leidžiantis patikrinti, ar ištrynimas buvo įvykdytas [24].



9 pav. ATDD sistemos architektūra ir operacijos [24]

Kaip matyti 9 paveiksle, ATDD modelyje duomenų prieinamumas papildomai siejamas su galiojimo laiku, o patikimoji šalis dalyvauja valdant laiko parametrus ir ištrynimo procesą. Tokia schema leidžia realizuoti laiku valdomą smulkiai valdomą duomenų nebeprieinamumą.

Šie sprendimai svarbūs tuo, kad siūlo ne vien bendrą rakto sunaikinimo modelį, bet ir tikslesnį ryšį tarp prieigos politikos, atributo ar laiko sąlygos ir konkretaus duomenų rinkinio. Kitaip tariant, jie

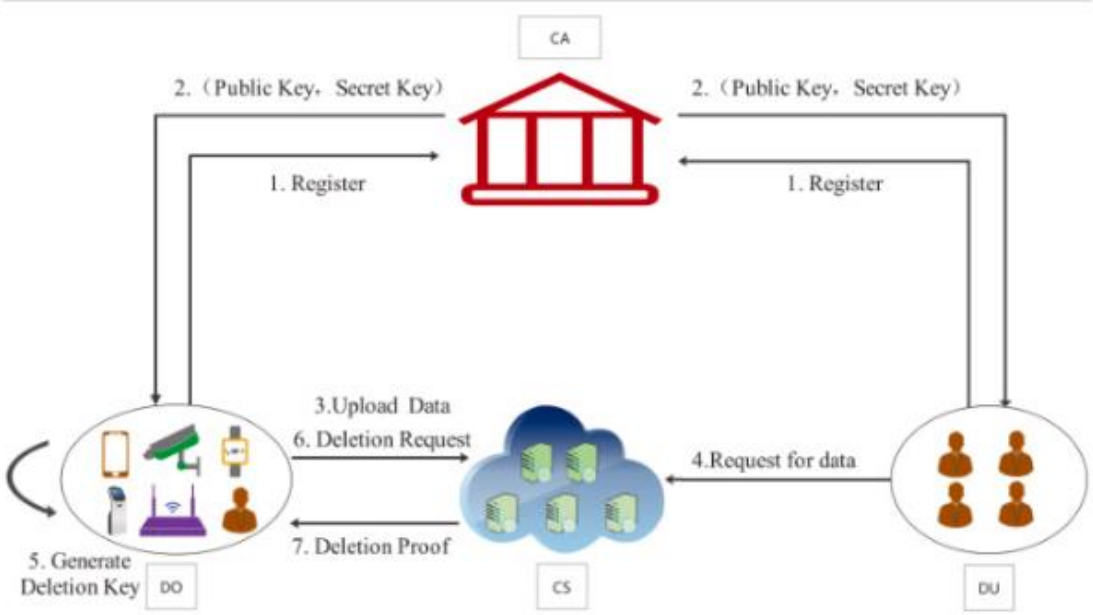
priartina užtikrintą duomenų ištrynimą prie realių duomenų valdymo scenarijų, kuriuose reikia naikinti ne „viską pagal vieną raktą“, o konkretų failą, duomenų klasę ar laikotarpį. Vis dėlto tokios schemos paprastai yra sudėtingesnės, priklauso nuo atributais grindžiamo šifravimo (angl. *attribute-based encryption, ABE*) konstrukcijų ir didina valdymo bei skaičiavimo sąnaudas [23], [24]. Kartu jos svarbios tuo, kad aiškiai parodo, jog saugus ištrynimas tampa tiesiogiai priklausomas nuo to, kaip valdomi prieigos atributai, laiko parametrai ir su jais susiję raktai.

1.4.3. Verifikuojamo ištrynimo sprendimai

Kita svarbi kryptis yra verifikuojamas ištrynimas (angl. *verifiable deletion*), kai nepakanka vien manyti, kad duomenys tapo nebeprieinami – reikia ir galimybės tą faktą patikrinti arba gauti patikimą įrodymą. Tokie sprendimai ypač svarbūs paskirstytoje infrastruktūroje, nes duomenų savininkas tiesiogiai nemato visos saugojimo infrastruktūros ir negali savarankiškai patvirtinti, kas tiksliai įvyko paslaugos teikėjo pusėje.

Vienas naujesnių pavyzdžių yra „SevDel“. Šio darbo autoriai teigia, kad tradiciniai metodai, paremti susiejimo panaikinimu, perrašymu ar paprastu kriptografiniu raktų valdymu, paskirstytoje saugykloje yra arba neveiksmingi, arba remiasi nepraktiškomis prielaidomis. Dėl to siūlomas modelis, kuris naudoja nulinio atskleidimo įrodymą (angl. *zero-knowledge proof*), kad būtų galima patikrinti, ar duomenys buvo korektiškai apdoroti, o raktų panaikinimas grindžiamas saugiosios vykdymo aplinkos mechanizmu [25].

Kita kryptis yra EVSD (angl. *An Efficient and Verifiable Scheme for Secure Data Deletion*). Šio modelio autoriai pabrėžia, kad daugelis ankstesnių kriptografiškai pagrįstų sprendimų, ypač pagrįstų atributais grindžiamu šifravimu, yra skaičiavimo požiūriu sudėtingi, todėl siūlo efektyvesnę schemą, jungiančią šifravimą, smulkiai valdomą ištrynimo politiką ir greitą patikros algoritmą [26].



10 pav. EVSD sistemos modelis [26]

Kaip matyti 10 paveiksle, EVSD modelyje duomenų savininkas įkelia duomenis į debesijos serverį, duomenų naudotojas gali prašyti prieigos, o centrinė institucija dalyvauja registracijos ir raktų paskirstymo procese. Ištrynimo atveju duomenų savininkas inicijuoja trynimo užklausą, o debesijos

serveris grąžina ištrynimo įrodymą, leidžiantį patikrinti, ar duomenų pašalinimo procesas buvo atliktas korektiškai.

Lyginant su ankstesniais modeliais, verifikuojamo ištrynimo sprendimai suteikia papildomą pasitikėjimo sluoksnį. Jei FADE ir panašūs modeliai daugiausia sprendžia raktų atskyrimo ir politikomis valdomo nebeprieinamumo problemą, tai verifikuojamo ištrynimo sprendimai sprendžia ir įrodymo spragą – kaip duomenų savininkas gali pagrįstai tikėti, kad ištrynimo procesas tikrai buvo atliktas korektiškai. Tai ypač svarbu šiame darbe, nes jame akcentuojamas ne tik techninis sunaikinimas, bet ir atsekamumas bei audituojamumas [25], [26]. Kartu šie modeliai parodo, kad raktų valdymo veiksmai gali būti ne tik atliekami, bet ir patikrinami, o tai svarbu vertinant praktinį tokio metodo patikimumą.

1.4.4. Komercinių debesijos platformų saugaus duomenų ištrynimo mechanizmai

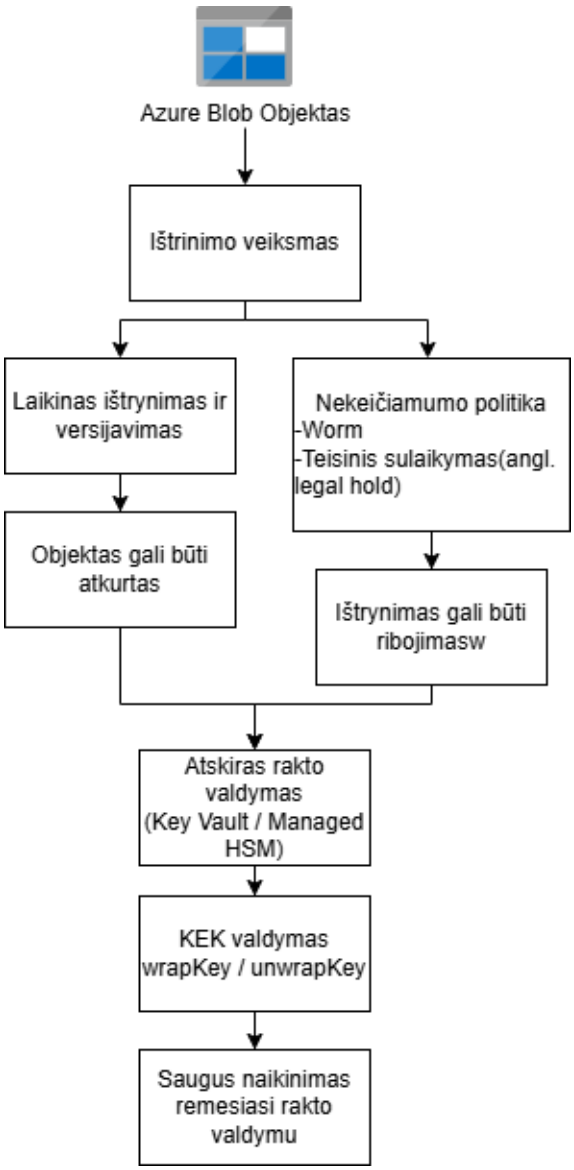
Komercinių debesijos platformų mechanizmų analizė svarbi ne todėl, kad jie būtinai bus tiesiogiai naudojami siūlomame sprendime, bet todėl, kad jie parodo, kaip brandžiose sistemose praktiškai realizuojamas ryšys tarp saugojimo būsenų, raktų valdymo ir duomenų turinio nebeiššifruojamumo bei praktinio nebeatkuriamumo. Tokie mechanizmai padeda suprasti, kurios funkcijos yra reikalingos saugiam duomenų ištrynimui, net jei vėliau jos bus realizuojamos savarankiškai valdomoje virtualiojoje infrastruktūroje.

1.4.4.1. „Microsoft Azure“ mechanizmai

„Microsoft Azure“ aplinkoje saugaus duomenų ištrynimo analizė turi būti grindžiama ne vien bendromis saugyklos galimybėmis, bet būtent tais mechanizmais, kurie lemia, ar objektas gali būti atkurtas, ar jo ištrynimas yra atidėtas, ir kaip raktų valdymo sluoksnis gali būti panaudotas duomenų turinio nebeiššifruojamumui ir praktiniam nebeatkuriamumui pasiekti. „Azure Blob Storage“ dokumentacijoje nurodoma, kad objektų laikinas išsaugojimas po ištrynimo (angl. *blob soft delete*) apsaugo objektą, momentinę kopiją arba versiją nuo atsitiktinio ištrynimo ar perrašymo, išlaikydamas ištrintus duomenis sistemoje nustatytą laikotarpį; per šį laikotarpį objektą galima atkurti, o tik jam pasibaigus objektas pašalinamas galutinai [6]. Be to, versijų kūrimo mechanizmas leidžia išlaikyti ankstesnes objekto versijas, o jų šalinimas yra atskiras veiksmas [7]. Tai reiškia, kad vien loginis ištrynimas „Azure“ aplinkoje dažnai nekeičia to fakto, kad su tuo pačiu objektu susiję duomenų egzemplioriai vis dar egzistuoja saugykloje [6], [7].

Papildomą ribojimą sukuria nekeičiamumo mechanizmai (angl. *immutability*). „Azure“ dokumentacijoje nurodoma, kad nekeičiamoji saugykla (angl. *immutable storage*) leidžia saugoti objektų duomenis principu „rašyti vieną kartą, skaityti daug kartų“ (angl. *write once, read many, WORM*), kai duomenys negali būti modifikuoti ar ištrinti naudotojo nustatytą laikotarpį, o tokios politikos gali būti pagrįstos laiku arba teisiniu sulaikymu [27]. Todėl objektų pašalinimas „Azure“ aplinkoje gali būti atidėtas ar ribojamas net tada, kai naudotojas inicijuoja trynimą [27].

Dėl šios priežasties „Azure“ aplinkoje ypač svarbus tampa raktų valdymo sluoksnis. „Managed HSM“ dokumentacijoje parodyta, kad galima kurti raktus, skirtus tik apvyniojimo ir išvyniojimo operacijoms (angl. *wrapKey ir unwrapKey*), o tokio ištrinto rakto būsena saugoma dėl visada įjungto laikino išsaugojimo po ištrynimo, po kurio raktas galutinai pašalinamas [16]. Tai palaiko KEK vaidmenį voko šifravimo architektūroje, nes KEK gali būti naudojamas DEK apsaugoti jo neeksportuojant į taikomąją sistemą [16].



11 pav. „Azure“ saugaus duomenų ištrynimo mechanizmų schema

Kaip matyti 11 paveiksle, „Azure“ aplinkoje objekto ištrynimo rezultatas priklauso ne tik nuo pačios trynimo komandos, bet ir nuo laikino išsaugojimo po ištrynimo, versijų kūrimo, nekeičiamumo politikų bei atskiro raktų valdymo sluoksnio. Dėl to saugus duomenų naikinimas šioje aplinkoje turi būti vertinamas kaip objektų būsenų ir raktų valdymo sąveika [6], [7], [16], [27].

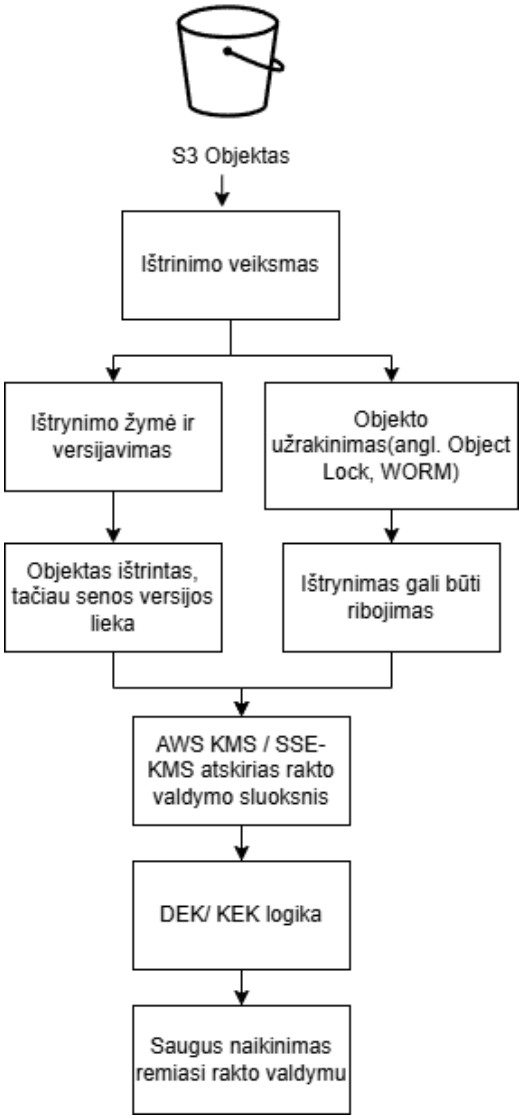
1.4.4.2. „Amazon Web Services“ mechanizmai

AWS aplinkoje pagrindiniai saugaus duomenų ištrynimo aspektai yra susiję su „S3“ versijų kūrimu, ištrynimo žyme (angl. *delete marker*), objektų užrakinimu (angl. *Object Lock*) ir „AWS KMS“ mechanizmais. AWS dokumentacijoje nurodoma, kad „S3“ konteineryje, kuriame įjungtas versijų kūrimas, paprasta ištrynimo užklausa nesunaikina objekto, o sukuria ištrynimo žymę, dėl kurios sistema elgiasi taip, lyg objektas būtų ištrintas. Tuo pačiu metu ankstesnės objekto versijos išlieka saugykloje ir gali būti pasiektos per versijos identifikatorių [8].

Papildomą ribojimą sukuria „*S3 Object Lock*“. AWS dokumentacijoje nurodoma, kad ši funkcija gali neleisti objektų ištrinti ar perrašyti nustatytą laiką arba neribotai, naudodama WORM modelį [9]. Dar svarbiau tai, kad net jei objekto versijos yra apsaugotos, ištrynimo žymė vis tiek gali būti sukurta,

todėl objektas gali atrodyti ištrintas, nors jo apsaugotos versijos išlieka saugykloje [9]. Tai rodo, kad AWS aplinkoje „ištrintas“ nebūtinai reiškia „galutinai pašalintas“ [8], [9].

Raktų valdymo požiūriu „AWS KMS“ suteikia aiškų voko šifravimo modelį. AWS dokumentacijoje nurodoma, kad „Amazon S3“, naudodamas serverio pusės šifravimą su KMS raktais (angl. *SSE-KMS*), šifruoja duomenis duomenų šifravimo raktu (angl. *data key*), po naudojimo kuo greičiau pašalina jo neužšifruotą reikšmę iš atminties, o užšifruotą duomenų raktą išsaugo kaip metaduomenis kartu su užšifruotais duomenimis [14], [28]. Kai reikia duomenis iššifruoti, „S3“ siunčia užšifruotą duomenų raktą į „AWS KMS“, gauna jo neužšifruotą reikšmę ir tik tada atkuria objekto turinį [14], [28]. Tokia schema realizuoja DEK ir KEK logiką.



12 pav. AWS saugaus duomenų ištrynimo mechanizmų schema

Kaip matyti 12 paveiksle, AWS aplinkoje objekto ištrynimo rezultatas priklauso ne tik nuo ištrynimo veiksmo, bet ir nuo ištrynimo žymės, versijų kūrimo, objektų užrakinimo bei „AWS KMS“ raktų valdymo sluoksnio. Dėl to saugus duomenų naikinimas šioje aplinkoje turi būti vertinamas kaip saugyklos būsenų ir rakto gyvavimo ciklo sąveika.

1.4.5. KEK valdymo kryptys

KEK valdymo kryptys yra viena svarbiausių šio skyriaus dalių, nes jos parodo, kaip praktiškai gali būti realizuojamas duomenų turinio nebeiššifruojamumas ir praktinis nebeatkuriamumas. Kadangi šiame darbe pagrindinis dėmesys skiriamas kriptografinių raktų valdymo metodui saugiam duomenų ištrynimui, būtent KEK valdymas tampa centrine analitine ašimi. Analitiškai šias kryptis tikslinga grupuoti ne pagal kiekvieną atskirą priemonę, o pagal bendrą rakto kilmės, saugojimo ir atkūrimo logiką. Toks grupavimas leidžia išlaikyti visus svarbius sprendimus, tačiau pateikti juos nuoseklesne forma.

1.4.5.1. Atskirtame KMS arba HSM sluoksnyje saugomas ir importuotas KEK

Brandžiausia ir praktiškai plačiausiai taikoma kryptis yra modelis, kuriame KEK saugomas atskirame raktų valdymo sluoksnyje, o pats duomenų turinys saugomas kitoje sistemoje. Tokiu atveju duomenų šifravimo raktas (angl. *Data Encryption Key, DEK*) naudojamas objekto ar failo turiniui apsaugoti, o DEK savo ruožtu apsaugomas KEK. Ši logika atitinka voko šifravimo principą ir leidžia atskirti duomenų saugojimą nuo pagrindinio rakto valdymo proceso [22], [16], [28].

Toks modelis aiškiai matomas tiek mokslinėje literatūroje, tiek praktiniuose komerciniuose sprendimuose. FADE sistemoje užšifruoti duomenys saugomi trečiosios šalies saugykloje, o raktų valdymas perduodamas atskiram rakto valdytojui [22]. „Azure“ atveju kliento valdomi raktai gali būti laikomi „Azure Key Vault“ arba „Azure Managed HSM“, o „Managed HSM“ palaiko raktų naudojimą apvyniojimo ir išvyniojimo operacijoms [16]. AWS aplinkoje analogiška logika įgyvendinama per „AWS KMS“ ir „SSE-KMS“ modelį [28].

Prie šios grupės priskirtinas ir importuotas KEK, dar vadinamas „atsinešk savo raktą“ modeliu (angl. *Bring Your Own Key, BYOK*). Šiuo atveju raktas nėra generuojamas pačioje debesijos aplinkoje, bet sukuriamas išorinėje organizacijos valdomoje infrastruktūroje ir vėliau importuojamas į raktų valdymo sluoksnį. Toks modelis leidžia organizacijai išlaikyti didesnę rakto kilmės ir pirminio valdymo kontrolę, nors pats KEK vėliau vis tiek gali būti naudojamas debesijos KMS ar HSM sluoksnyje [40].

Apibendrinant galima teigti, kad atskirtame KMS arba HSM sluoksnyje saugomas, taip pat importuotas KEK yra brandžiausia ir praktiškai geriausiai realizuota raktų valdymo kryptis. Jos pranašumas yra aiški DEK ir KEK hierarchija, geras audituojamumas ir aukštas suderinamumas su paskirstytos infrastruktūros architektūra.

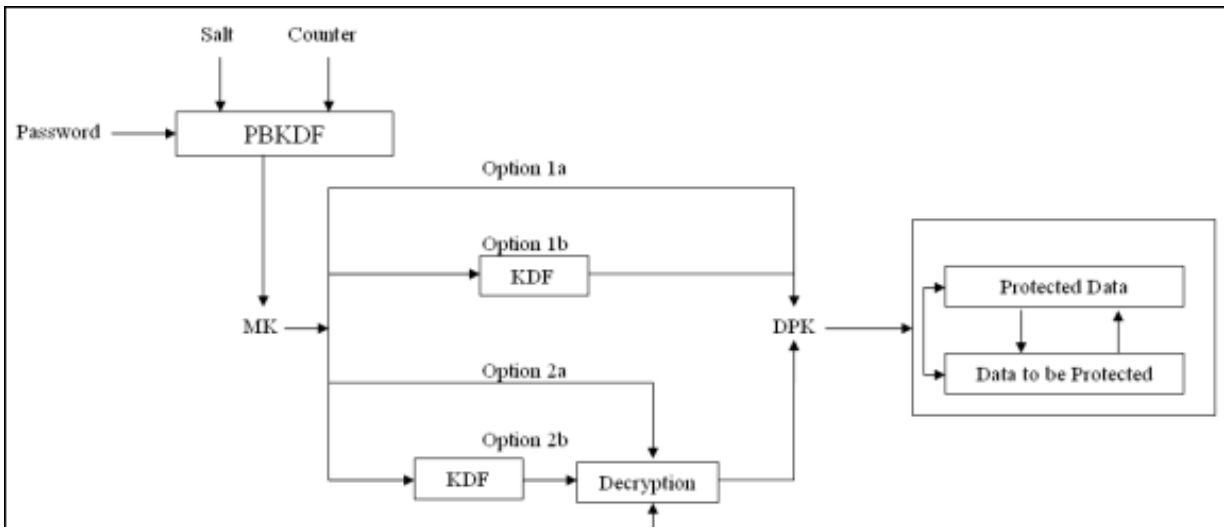
1.4.5.2. Politikomis, atributais ar laiku valdomas KEK

Kita svarbi kryptis yra modeliai, kuriuose prieiga prie KEK arba su juo susieto valdymo rakto siejama su politika, atributais ar galiojimo laiku. Tokiu atveju duomenų nebeprieinamumas pasiekiamas ne vien rankiniu rakto sunaikinimu, bet ir automatiškai pasikeitus arba nustojus galioti taisyklėms, nuo kurių priklauso rakto panaudojimas [22], [24].

Vienas ankstyviausių tokio modelio pavyzdžių yra FADE, o vėlesniuose darbuose ši kryptis išplėtota į atributais valdomą ADAA modelį ir laiku valdomą ATDD modelį. Tokios schemos svarbios tuo, kad priartina kriptografinį duomenų ištrynimą prie realių duomenų valdymo scenarijų, kuriuose reikia valdyti ne visos saugyklos, o konkrečių failų, duomenų klasių ar laikotarpių prieinamumą [22], [24].

1.4.5.3. Iš naudotojo paslapties arba biometrikos išvedamas KEK

Viena iš alternatyvių krypčių yra KEK valdymas be ilgalaikio rakto saugojimo. Paprasčiausia šios krypties forma yra slaptažodžiu išvedamas KEK. NIST SP 800-132 nurodo, kad slaptažodžiais grindžiamos raktų išvedimo funkcijos leidžia iš slaptažodžių ar slaptafrazių išvesti kriptografinius raktus elektroninių duomenų ar duomenų apsaugos raktų apsaugai [19]. Tai reiškia, kad teoriškai KEK nebūtina saugoti kaip ilgalaikio objekto – jis gali būti išvedamas pagal poreikį iš slaptažodžio ir viešų parametrų, tokių kaip druska ir iteracijų skaičius [19].



13 pav. Slaptažodžiu grindžiamos rakto išvedimo schemos principas [19]

Kaip matyti 13 paveiksle, slaptažodis kartu su druska ir iteracijų skaičiumi panaudojamas slaptažodžiu grindžiamoje rakto išvedimo funkcijoje (angl. *Password-Based Key Derivation Function, PBKDF*) pagrindiniam raktui išvesti. Tolesnėje schemos dalyje šis išvestas raktas naudojamas duomenų apsaugos raktui arba patiems saugomiems duomenims apsaugoti.

Vis dėlto tokia kryptis turi aiškų ribotumą. NIST SP 800-133 Rev. 2 pažymi, kad raktų generavimas iš slaptažodžių yra problemiškas, nes slaptažodžiai paprastai suteikia mažai entropijos ir gali būti lengvai atspėjami [29]. Todėl slaptažodžiu išvedamas KEK mažina ilgalaikio rakto saugojimo problemą, tačiau jo saugumas labai priklauso nuo naudotojo paslapties kokybės [19], [29].

Artima kryptis yra biometriniais duomenimis grindžiamas KEK. Dodis ir bendraautoriai nurodo, kad neaiškių duomenų išgavimo metodas (angl. *fuzzy extractor*) leidžia iš triukšmingos įvesties, tokios kaip biometriniai duomenys, išgauti beveik atsitiktinę reikšmę, kuri gali būti naudojama kaip raktas, o saugioji santrauka (angl. *secure sketch*) leidžia iš viešos pagalbinės informacijos ir artimos įvesties atkurti originalią reikšmę neatskleidžiant pačios biometrikos [30]. Toks modelis teoriškai tinka KEK be nuolatinio saugojimo idėjai, nes raktas rekonstruojamas iš biometrinės įvesties ir pagalbinės informacijos pagal poreikį. Vis dėlto naujesni darbai rodo, kad vien biometrika grindžiami modeliai gali būti pažeidžiami, jei pati biometrika nuteka arba gali būti imituojama [31].

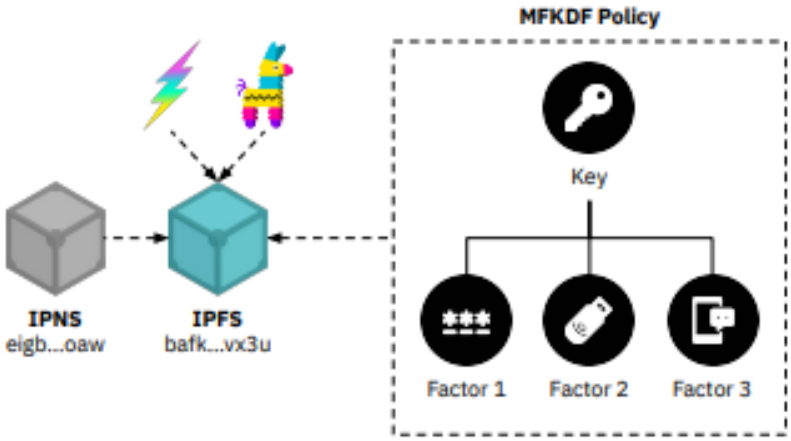
Papildoma šios grupės kryptis yra elgsenos biometrika grindžiamas KEK. Skirtingai nei klasikinė biometrika, paremta piršto atspaudu ar kitais fiziologiniais požymiais, elgsenos biometrika remiasi naudotojo veiksenos modeliais, pavyzdžiui, klaviatūros dinamika. Tyrimai rodo, kad iš tokių požymių gali būti generuojama kriptografiškai naudinga rakto medžiaga [34]. Ši kryptis svarbi tuo,

kad išplečia biometrinių sprendimų spektrą ir parodo, jog KEK atkūrimas gali būti siejamas ne tik su fiziologiniais, bet ir su elgsenos požymiais [34].

1.4.5.4. Kelių veiksnių ir regeneruojami KEK modeliai

Dar stipresnė kryptis yra kelių veiksnių pagrindu išvedamas KEK. Vienas ankstyvesnių šios krypties pavyzdžių yra slaptažodžiu sustiprintas neaiškių duomenų seifas (angl. *password-hardened fuzzy vault*), kuriame pirštо atspaudo pagrindu veikianti schema papildomai sustiprinama slaptažodžiu [32]. Toks požiūris padeda sumažinti vien biometrikai būdingas rizikas ir sustiprina autentifikavimo pagrindą.

Dar bendresnį modelį siūlo kelių veiksnių raktų išvedimo funkcija (angl. *Multi-Factor Key Derivation Function, MFKDF*). Nair ir Song aprašo konstrukciją, kuri išplečia slaptažodžiu grindžiamą raktų išvedimą įtraukdama papildomus autentifikavimo faktorius, tokius kaip vienkartiniai slaptažodžiai pagal laiką (angl. *TOTP*), vienkartiniai slaptažodžiai pagal skaitiklį (angl. *HOTP*), išoriniai autentifikavimo kanalai ar aparatiniai žetonai [33].



14 pav. Kelių veiksnių pagrindu išvedamo rakto sudarymo principas [33]

Kaip matyti 14 paveiksle, keli autentifikavimo veiksniai kartu apibrėžia prieigos politiką ir leidžia išvesti bendrą rakto medžiagą. Šio darbo kontekste tokia logika gali būti interpretuojama kaip kelių veiksnių pagrindu išvedamo KEK modelis.

Prie šios grupės taip pat priskirtinas dviejų veiksnių regeneruojamas KEK, kai raktas gali būti atkuriamas pagal poreikį iš dviejų nepriklausomų faktorių, pavyzdžiui, biometrikos ir slaptažodžio [35]. Artima kryptis yra kelių veiksnių neaiškių duomenų išgavimo metodas (angl. *multi-factor fuzzy extractor*), kuriame biometrika ir papildoma naudotojo paslaptis sujungiamos į vieną rakto generavimo ir rekonstrukcijos procesą [31]. Tokie modeliai sumažina priklausomybę nuo vieno autentifikavimo faktoriaus ir leidžia rakto atkūrimą susieti su kelių nepriklausomų sąlygų įvykdymu [31], [35].

1.4.5.5. Paskirstyti, padalyti, aparatiniai ir „be rakto saugojimo“ modeliai

Kita grupė apima modelius, kuriuose KEK nėra saugomas kaip vienas ilgalaikis objektas vienoje vietoje. Viena iš svarbių krypčių yra slenkstinis arba padalytas KEK. Tokiuose modeliuose vienas KEK suskaidomas į kelias dalis, kurios saugomos atskirai ir gali būti rekonstruojamos tik surinkus

pakankamą jų skaičių. Tokie sprendimai mažina vieno kompromitavimo taško riziką ir leidžia paskirstyti pasitikėjimą tarp kelių saugojimo ar atkūrimo dalyvių [36].

Prie aparatinėmis savybėmis grindžiamų krypčių priskirtinas fiziškai neklonuojama funkcija pagrįstas KEK (angl. *Physically Unclonable Function, PUF*). PUF literatūroje pabrėžiama, kad fizinės įrenginio savybės gali būti naudojamos unikaliam rakto atsakui generuoti, o pagalbiniai duomenys ir klaidų taisymo mechanizmai leidžia tokį raktą atkurti be nuolatinio jo saugojimo [37]. Ši kryptis svarbi tuo, kad raktas siejamas su pačiu fiziniu įrenginiu, todėl gali būti laikoma aparatine alternatyva tradiciniam KEK saugojimui [37].

Papildoma paskirstyto atkūrimo kryptis yra slaptažodžiu apsaugotas paslapties padalijimas (angl. *Password-Protected Secret Sharing, PPSS*). Ši schema leidžia aukštos entropijos paslaptį paskirstyti keliems serveriams ir vėliau ją atkurti naudojant slaptažodį, nepasikliaujant vienu centriniu ilgalaikio rakto saugojimo tašku [38].

Galiausiai literatūroje aptinkama ir šifravimo be ilgalaikio rakto saugojimo kryptis (angl. *keyless encryption*). Tai kol kas gana nišinis ir mažiau brandus požiūris, kuriuo siekiama sumažinti arba visai pašalinti ilgalaikės rakto informacijos saugojimą per visą šifravimo ir iššifravimo procesą, dažnai pasitelkiant PUF tipo mechanizmus ar kitus aparatinės prigimties atsakus [39]. Nors ši kryptis šiuo metu labiau vertintina kaip eksperimentinė, ji svarbi analizei, nes rodo kraštutinę alternatyvą klasikiniam KEK modeliui [39].

1.4.5.6. Šių krypčių privalumai ir ribotumai

Aptartos KEK valdymo kryptys rodo, kad rakto saugojimo problema gali būti sprendžiama keliais skirtingais būdais. Atskirtame KMS arba HSM sluoksnyje saugomas ir importuotas KEK pasižymi brandumu, aiškia DEK ir KEK hierarchija bei geru audituojamumu [22], [16], [28], [40]. Politikomis, atributais ar laiku valdomi modeliai leidžia tiksliau ir lanksčiau valdyti duomenų nebeprieinamumą, susiejant rakto prieinamumą su konkrečiomis taisyklėmis [22]–[24]. Tuo tarpu slaptažodžiu, biometrika, kelių veiksnių, PUF ar PPSS grindžiami modeliai mažina ilgalaikio rakto saugojimo problemą ir leidžia KEK atkurti tik tada, kai jo iš tikrųjų reikia [19]–[39].

Vis dėlto kiekviena iš šių krypčių turi ir ribotumų. KMS arba HSM modeliai vis dar remiasi tuo, kad KEK egzistuoja kaip ilgalaikis objektas atskirame sluoksnyje. Politikomis ir atributais grindžiami modeliai tampa sudėtingesni dėl papildomų valdymo taisyklių, atributais grindžiamo šifravimo konstrukcijų ar laiko logikos [23], [24]. Slaptažodžiu išvedami raktai priklauso nuo žmogaus pasirenkamų paslapčių kokybės [19], [29], biometriniai raktai – nuo biometrikos vientisumo, neatšaukiamumo ir apsaugos nuo imitavimo [30], [31], o kelių veiksnių, PUF ir PPSS modeliai – nuo sudėtingesnės architektūros, atkūrimo procedūrų ir praktinės brandos stokos [31]–[39].

Todėl nė viena kryptis negali būti laikoma universaliu sprendimu visiems scenarijams. Tačiau jų palyginimas leidžia geriau suprasti kompromisus tarp praktinio brandumo, architektūrinio sudėtingumo, saugumo stiprumo ir priklausomybės nuo ilgalaikio rakto saugojimo.

6 lentelė. KEK valdymo modelių palyginimas

KEK valdymo modelis	Esmė	Privalumas	Ribotumas
KEK saugomas atskirame KMS / HSM sluoksnyje	KEK laikomas atskiroje raktų valdymo sistemoje, o DEK apsaugomas šiuo raktu	Brandus ir praktiškai lengvai pritaikomas modelis	Raktas vis tiek ilgai saugomas sistemoje
Importuotas KEK / BYOK	KEK sukuriamas už debesijos ribų ir importuojamas į raktų valdymo sistemą	Didesnė rakto kilmės ir valdymo kontrolė	Išlieka priklausomybė nuo raktų valdymo sluoksnio
Politikomis, atributais arba laiku valdomas KEK	Prieiga prie KEK valdoma pagal taisykles, atributus arba laiką	Leidžia lanksčiai valdyti prieigą prie duomenų	Sudėtingesnis valdymas ir realizacija
Iš slaptažodžio arba biometrikos išvedamas KEK	KEK nėra nuolat saugomas, o atkuriamas iš naudotojo paslapties	Mažina ilgalaikio rakto saugojimo poreikį	Saugumas priklauso nuo naudotojo paslapties patikimumo
Kelių veiksnių regeneruojamas KEK	KEK atkuriamas panaudojus kelis autentifikavimo faktorius	Stipresnė apsauga nei naudojant vieną faktorių	Sudėtingesnis atkūrimo procesas
Padalytas arba paskirstytas KEK	KEK suskaidomas į dalis arba atkuriamas iš kelių šaltinių	Mažina vieno pažeidimo taško riziką	Sudėtingesnis diegimas ir mažesnis praktinis brandumas

Kaip matyti 6 lentelėje, brandžiausi yra KMS / HSM ir BYOK modeliai, o išvedami, regeneruojami ir paskirstyti KEK modeliai suteikia daugiau lankstumo, tačiau yra sudėtingesni praktiniam įgyvendinimui.

1.4.5.7. Tinkamumas siūlomam sprendimui

Vertinant aptartas KEK valdymo kryptis siūlomo sprendimo kontekste, matyti, kad praktiškai patikimiausias pagrindas išlieka modelis, kuriame duomenų saugojimo ir raktų valdymo sluoksniai yra aiškiai atskirti. Toks modelis geriausiai dera su kriptografinio duomenų ištrynimo logika, nes leidžia duomenų turinio nebeiššifruojamumą ir praktinį nebeatkuriamumą pasiekti per DEK ir KEK hierarchiją bei rakto prieigos panaikinimą [22], [16], [28].

Kita vertus, vien ilgalaikio KEK saugojimo modelis neatskleidžia visų galimų projektinių alternatyvų. Politikomis, atributais ar laiku valdomi KEK modeliai rodo, kad prieiga prie rakto gali būti valdoma gerokai smulkiau, o iš slaptažodžio, biometrikos, kelių veiksnių, PUF ar PPSS išvedami modeliai parodo, kad KEK nebūtinai turi būti laikomas kaip statinė paslaptis [19]–[24], [30]–[39]. Tai ypač svarbu šiam darbui, nes leidžia svarstyti originalesnį sprendimą, kuriame rakto atkūrimas galėtų būti susietas ne tik su infrastruktūros komponentu, bet ir su naudotojo valdomu išoriniu veiksniu.

Todėl analitinio naujumo požiūriu vertingiausia kryptis yra derinys tarp klasikinio atskiro raktų valdymo sluoksnio ir alternatyvaus, pagal poreikį išvedamo ar regeneruojamo KEK modelio. Toks požiūris leidžia išlaikyti praktinį pagrįstumą, remiantis DEK ir KEK hierarchija, ir kartu suteikia erdvės originalesniam sprendimui, kuriame bent dalis raktų valdymo funkcijų nebūtų priklausomos nuo nuolat saugomo KEK.

1.4.6. Esamų sprendimų palyginimas

Siekiant aiškiai parodyti skirtumus tarp mokslinėje literatūroje siūlomų sprendimų ir realių komercinių mechanizmų, tikslinga atlikti bendrą palyginimą pagal keturis kriterijus: pagrindinę ištrynimo logiką, smulkaus ištrynimo galimybes, įrodomumą ir praktinį pritaikomumą. Iš ankstesnės analizės matyti, kad FADE ir panašūs raktų atskyrimo modeliai yra labai gerai suderinami su paskirstytos infrastruktūros logika, tačiau jų smulkaus ištrynimo galimybės yra ribotos; atributais ir laiku grindžiamos schemos suteikia daugiau tikslinio valdymo, tačiau didina sudėtingumą; verifikuojamo ištrynimo sprendimai suteikia papildomą patikros sluoksnį, bet reikalauja papildomos kriptografinės ar aparatinės architektūros; o komerciniai mechanizmai pasižymi labai geru praktiniu brandumu, tačiau patys savaime nepašalina visų versijų kūrimo, ištrynimo žymių, nekeičiamumo ar laikino išsaugojimo po ištrynimo ribojimų.

7 lentelė. Esamų saugaus duomenų naikinimo sprendimų palyginimas

Sprendimas	**Esmė**	**Privalumas**	**Ribotumas**
Raktų atskyrimu pagrįsti sprendimai	Duomenys ir raktai laikomi atskirai	Gerai sprendžia kopijų ir fizinės kontrolės stokos problemą	Ribotos smulkaus ištrynimo galimybės
Atributais ir laiku grindžiami sprendimai	Prieiga valdoma atributais arba galiojimo laiku	Leidžia atrankinį duomenų ištrynimą	Didesnis sudėtingumas
Verifikuojamo ištrynimo sprendimai	Ištrynimas papildomas įrodymo mechanizmu	Sustiprina audituojamumą	Reikalauja sudėtingesnės infrastruktūros
„Azure“ mechanizmai	Ištrynimas grindžiamas saugyklos būsenomis ir raktų valdymu	Didelis praktinis brandumas	Vien trynimo komandos nepakanka negrįžtamumui užtikrinti
AWS mechanizmai	Ištrynimas grindžiamas versijų kūrimu, ištrynimo žyme, „Object Lock“ ir KMS	Brandūs praktiniai mechanizmai	Objektas gali atrodyti ištrintas, nors versijos išlieka

Kaip matyti 7 lentelėje, brandžiausi praktinio diegimo požiūriu yra komerciniai mechanizmai, tačiau mokslinėje literatūroje siūlomi sprendimai suteikia svarbių papildomų įžvalgų: raktų atskyrimo modeliai padeda spręsti kopijų problemą, smulkiai valdomi sprendimai – atrankinio ištrynimo problemą, o verifikuojamo ištrynimo sprendimai – įrodomumo problemą. Tuo tarpu atskirai išanalizuotos KEK valdymo kryptys suteikia papildomą architektūrinį pagrindą vertinti, kokiu būdu praktiniame sprendime galėtų būti organizuojamas rakto atkuriamumas arba jo panaikinimas.

1.4.7. Poskyrio apibendrinimas

Atlikta esamų sprendimų analizė parodė, kad saugaus duomenų ištrynimo literatūroje pastebimas perėjimas nuo bandymo fiziškai ištrinti visas duomenų kopijas prie modelių, kuriuose duomenų nebeprieinamumas pasiekiamas per šifravimą, raktų atskyrimą, politikų valdymą ir papildomą įrodomumą. FADE tipo sprendimai parodė, kad paskirstytoje infrastruktūroje praktiška atskirti duomenų saugojimą ir raktų valdymą; ADAA ir ATDD schemos parodė, kad galima siekti smulkiai valdomo ir laiku reguliuojamo ištrynimo; o verifikuojamo ištrynimo sprendimai išplėtė šią logiką į auditą ir patikrą.

„Azure“ ir AWS analizė parodė, kad realūs komerciniai sprendimai jau turi daug mechanizmų, reikalingų tokiam modeliui įgyvendinti: versijų kūrimą, laikiną išsaugojimą po ištrynimo,

nekeičiamumą, ištrynimo žymes, objektų užrakinimą, duomenų rakto logiką ir atskirą raktų valdymo sluoksnį. Tačiau taip pat paaiškėjo, kad vien platformos teikiamų trynimo mechanizmų nepakanka, jei siekiama ne tik administracinio pašalinimo, bet ir praktiškai negrįžtamo bei įrodomo duomenų panaikinimo.

Galiausiai KEK valdymo krypčių analizė parodė, kad egzistuoja ne tik ilgalaikio KEK saugojimo modeliai, bet ir alternatyvios kryptys, kuriose raktas išvedamas iš slaptažodžio, biometrikos, kelių veiksnių, PUF ar paskirstyto paslapties padalijimo mechanizmo. Ši analizė yra ypač vertinga šiam darbui, nes leidžia ne tik remtis egzistuojančiais mechanizmais, bet ir kritiškai svarstyti originalesnes raktų valdymo architektūras. Dėl to šio skyriaus išvada yra tokia: saugiam duomenų ištrynimui paskirstytoje infrastruktūroje tinkamiausias pagrindas yra kriptografinis metodas, tačiau jo praktinį veiksmingumą lemia pasirinktas kriptografinių raktų valdymo modelis. Todėl tolesnėje darbo dalyje pagrindinis dėmesys skiriamas tokiai raktų valdymo schemai, kuri leistų patikimai įgyvendinti duomenų turinio nebeiššifruojamumą ir praktinį nebeatkuriamumą per DEK ir KEK hierarchiją bei kontroliuojamą rakto atkuriamumo panaikinimą.

1.5. Analizės išvados

1.5.1. Metodų analizės rezultatai

Atlikta analizė parodė, kad paskirstytoje, debesijos ir virtualiojoje infrastruktūroje saugaus duomenų ištrynimo problema negali būti patikimai sprendžiama vien loginiu ištrynimu ar perrašymu pagrįstais metodais. Loginis ištrynimas dažniausiai pašalina tik aktyvią objekto būseną arba numatytąją prieigą prie jo, tačiau dėl versijų kūrimo, ištrynimo žymių, atsarginių kopijų ir laikino išsaugojimo po ištrynimo ne visada panaikina visas su objektu susijusias būsenas. Perrašymu pagrįstas metodas teoriškai gali suteikti stipresnį rezultatą, tačiau kietakūnių diskų architektūra, dėvėjimosi balansavimas ir fizinės laikmenos kontrolės nebuvimas reikšmingai sumažina jo praktinį veiksmingumą paskirstytoje infrastruktūroje.

Analizė taip pat parodė, kad kriptografinis duomenų ištrynimas sudaro stipriausią pagrindą praktiniam negrįžtamumui užtikrinti. Šis metodas grindžiamas ne fiziniu visų kopijų pašalinimu, o galimybės atkurti duomenų iššifravimo raktą panaikinimu. Dėl to jis geriau dera su paskirstytos infrastruktūros logika, kurioje fizinės laikmenos kontrolė dažniausiai nėra tiesiogiai pasiekiama. NIST SP 800-88 kriptografinį ištrynimą priskiria galutinio pašalinimo (angl. *purge*) kategorijai, o mokslinėje literatūroje ši kryptis vertinama kaip viena tinkamiausių užtikrinto duomenų ištrynimo sprendimų paskirstytoje aplinkoje.

Svarbu ir tai, kad analizė parodė ryškų skirtumą tarp metodo teorinio stiprumo ir jo praktinio įgyvendinamumo. Nors hibridinis metodas gali sustiprinti bendrą ištrynimo rezultatą, jo realizacija yra sudėtingesnė, nes priklauso nuo saugojimo politikų, versijų kūrimo, nekeičiamumo ir kitų infrastruktūrinių apribojimų. Todėl metodų palyginimas leidžia teigti, kad kriptografinis duomenų ištrynimas yra labiausiai su paskirstyta infrastruktūra suderinamas požiūris, o kiti metodai gali būti vertinami kaip papildomi arba riboto pritaikomumo.

1.5.2. Kriptografinių raktų valdymo principų apibendrinimas

Esamų sprendimų analizė parodė, kad praktiškai stipriausi saugaus duomenų ištrynimo modeliai remiasi duomenų saugojimo ir raktų valdymo atskyrimu. Raktų atskyrimu pagrįsti sprendimai, tokie

kaip FADE, parodė, kad net ir nežinant visų fizinių duomenų kopijų vietos galima pasiekti praktinį negrįžtamumą panaikinant prieigą prie rakto. Smulkiai valdomi sprendimai papildomai parodė, kad ši logika gali būti pritaikyta atrankiniam konkrečių objektų, atributų ar laiko sąlygų valdymui, o verifikuojamo ištrynimo sprendimai atskleidė atsekamumo ir patikros svarbą.

Komercinių mechanizmų analizė taip pat patvirtino, kad saugus duomenų ištrynimas praktikoje grindžiamas ne vien trynimo komanda, bet saugojimo būsenų ir raktų valdymo sąveika. Tiek „Microsoft Azure“, tiek AWS aplinkoje objekto pašalinimo rezultatas priklauso nuo versijų kūrimo, laikino išsaugojimo po ištrynimo, užrakinimo ar nekeičiamumo politikų bei atskiro raktų valdymo sluoksnio. Tai rodo, kad paskirstytoje infrastruktūroje architektūriškai svarbiausi tampa ne tik saugojimo mechanizmai, bet ir raktų hierarchija, jų apsauga bei sunaikinimo valdymas.

KEK valdymo krypčių analizė papildomai parodė, kad raktų valdymas gali būti organizuojamas skirtingais būdais: naudojant atskirtą KMS arba HSM sluoksnį, importuotus raktus, politikomis ar laiku valdomą prieigą, taip pat išvedamus ar regeneruojamus raktus. Tai leidžia daryti išvadą, kad kriptografinių raktų valdymo metodo požiūriu svarbiausi principai yra duomenų šifravimas, aiški DEK ir KEK hierarchija, atskirtas raktų valdymas, atrankinio ištrynimo galimybė ir pakankamas audituojamumas.

1.5.3. Analizės išvada

Apibendrinant galima teigti, kad saugus duomenų ištrynimas paskirstytoje infrastruktūroje yra ne vien techninė ištrynimo komandos problema, bet daugiasluoksnė architektūrinė, organizacinė ir norminė užduotis. Atlikta analizė parodė, kad duomenys gali išlikti versijose, momentinėse kopijose, atsarginėse kopijose, laikino išsaugojimo po ištrynimo būsenose ar kituose infrastruktūros sluoksniuose, todėl tikrasis saugaus ištrynimo kriterijus turi būti siejamas ne su objekto dingimu iš naudotojo rodinio, o su praktiniu duomenų nebeatkuriamumu.

Taip pat nustatyta, kad skirtingi sprendimai sprendžia skirtingas tos pačios problemos dalis. Raktų atskyrimo modeliai padeda spręsti kopijų ir kontrolės problemą, smulkiai valdomi sprendimai geriau atitinka atrankinio ištrynimo poreikį, verifikuojamo ištrynimo kryptis papildo procesą patikros ir audito sluoksniu, o KEK valdymo alternatyvos išplečia raktų valdymo galimybes. Vis dėlto kuo daugiau papildomų politikų, atributinių konstrukcijų ar patikros sluoksnių taikoma, tuo didesnis tampa sprendimo architektūrinis sudėtingumas.

Todėl atlikta analizė leidžia teigti, kad paskirstytai infrastruktūrai labiausiai pagrįstas požiūris yra kriptografinis duomenų ištrynimas, grindžiamas DEK ir KEK hierarchija, voko šifravimu ir atskirtu raktų valdymo sluoksniu. Kartu analizė parodė, kad praktinį tokio metodo veiksmingumą lemia ne vien pats šifravimo faktas, bet pasirinktas kriptografinių raktų valdymo modelis: kaip raktai kuriami, saugomi, išvedami, atkuriami ir kaip panaikinama galimybė juos atkurti. Tokia išvada sudaro loginį pagrindą pereiti prie kito darbo skyriaus, kuriame nagrinėjamas siūlomas kriptografinių raktų valdymo metodas ir jo realizavimas.

2. Kriptografinių raktų valdymo metodas saugiam duomenų ištrynimui

2.1. Metodo paskirtis ir esmė

Šiame darbe siūlomas metodas skirtas saugiam duomenų ištrynimui paskirstytoje, debesijos ir virtualiojoje infrastruktūroje, kai duomenų negrįžtamumas užtikrinamas ne vien failo pašalinimu iš saugyklos, bet kriptografiškai panaikinant galimybę atkurti jo turinį. Metodo paskirtis yra sudaryti tokias sąlygas, kad net ir tuo atveju, jei užšifruota failo kopija tam tikrą laiką išliktų saugojimo infrastruktūroje, pats duomenų turinys būtų praktiškai neatkuriamas, nes nebebūtų įmanoma atkurti raktų grandinės, reikalingos jam iššifruoti.

Siūlomo metodo esmė grindžiama duomenų šifravimo rakto ir raktų šifravimo rakto hierarchija. Kiekvieno failo turinys apsaugomas atskiru duomenų šifravimo raktu, o šis raktas papildomai apsaugomas naudojant raktų šifravimo raktą. Tokia schema leidžia atskirti duomenų turinio apsaugą nuo aukštesnio lygmens raktų valdymo proceso. Dėl to pagrindinis saugaus ištrynimo rezultatas šiame metode yra ne objekto išnykimas iš sistemos rodinio, bet duomenų turinio nebeiššifruojamumas ir praktinis nebeatkuriamumas, kylantis iš nebegalimos rakto rekonstrukcijos.

Svarbi šio metodo savybė yra ta, kad raktų šifravimo raktas nėra laikomas kaip vientisas ilgalaikis objektas vienoje vietoje. Jo atkūrimas priklauso nuo kelių atskirtų komponentų: serverio aplinkoje saugomos rakto dalies, naudotojo valdomos papildomos rakto dalies ir raktų išvedimo parametrų. Toks sprendimas reiškia, kad nei duomenų saugojimo sluoksnis, nei raktų ir metaduomenų sluoksnis atskirai nesudaro pakankamo pagrindo pradiniam failo turiniui atkurti.

Metodas orientuotas ne į fizinį visų duomenų kopijų sunaikinimą, bet į praktiškai svarbiausią rezultatą paskirstytoje infrastruktūroje – nebeįmanomą duomenų atkūrimą. Tokia logika atitinka kriptografinio duomenų ištrynimo principą, kai saugumo pagrindas perkeliamas nuo fizinės laikmenos valdymo prie kontroliuojamo raktų gyvavimo ciklo valdymo. Dėl šios priežasties siūlomas metodas gali būti vertinamas kaip kriptografinių raktų valdymo metodas, skirtas saugiam duomenų ištrynimui tais atvejais, kai fizinė laikmenos kontrolė yra ribota arba neprieinama.

2.2. Metodo veikimo logika

Siūlomo metodo veikimo logika grindžiama nuoseklia duomenų šifravimo, rakto apsaugos, atskirto saugojimo ir rakto atkuriamumo kontrolės seka. Pirmiausia į sistemą pateiktam failui sugeneruojamas atskiras duomenų šifravimo raktas. Šiuo raktu užšifruojamas pradinis failo turinys, todėl aktyvioje saugykloje nebesaugomas atviras duomenų pavidalas, o paliekamas tik šifruotas tekstas.

Sugeneruotas duomenų šifravimo raktas nėra saugomas atviru pavidalu. Jis papildomai apsaugomas raktų šifravimo raktu, kuris savo ruožtu nėra nuolat saugomas kaip vientisas ilgalaikis objektas. Vietoje to jis išvedamas iš kelių atskirtų įvesčių: serverio pusėje saugomos rakto dalies, naudotojo valdomos rakto dalies ir papildomų raktų išvedimo parametrų. Tokiu būdu realizuojamas ne tik raktų apsaugos, bet ir jų kontroliuojamo atkūrimo principas.

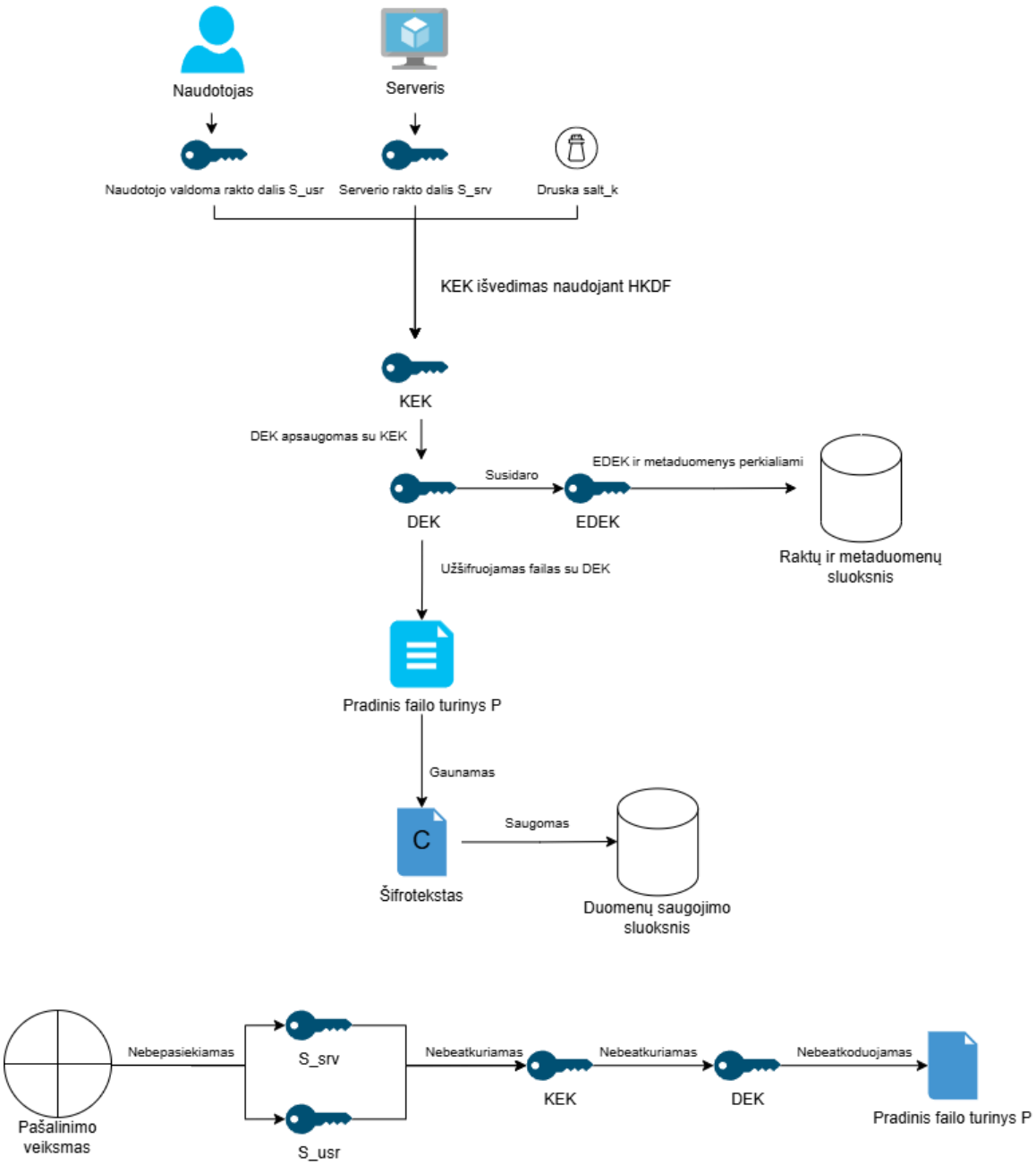
Kai failas turi būti atkurtas, sistema turi gauti ne tik užšifruotą failą ir užšifruotą duomenų šifravimo raktą, bet ir visas būtinas raktų išvedimo sąlygas. Tik iš šių komponentų atkūrus raktų šifravimo raktą tampa įmanoma atkurti duomenų šifravimo raktą, o tada iššifruoti patį failą. Tai reiškia, kad failo atkūrimas priklauso ne nuo vienos centralizuotos saugojimo vietos, bet nuo kelių logiškai atskirtų komponentų sąveikos.

Saugaus duomenų ištrynimo scenarijuje pagrindinis veiksmas yra ne užšifruoto failo pašalinimas iš saugyklos, o raktų grandinės atkuriamumo panaikinimas. Jei bent viena kritinė rakto išvedimo arba atkūrimo sudedamoji dalis tampa nepasiekiama arba sunaikinama, raktų šifravimo raktas nebegali būti atkurtas. Tokiu atveju nebegalima atkurti duomenų šifravimo rakto, o be jo nebegalima iššifruoti ir pradinio failo turinio. Tik po to, kaip papildoma priemonė, gali būti atliekamas užšifruoto failo pašalinimas iš aktyvios saugyklos bei atnaujinama su failu susijusių metaduomenų būsena.

Taigi siūlomo metodo veikimo logika aiškiai parodo, kad saugus duomenų ištrynimas šiuo atveju suprantamas kaip kriptografiškai valdomas duomenų turinio nebeiššifruojamumas ir praktinis nebeatkuriamumas. Pagrindinis rezultatas pasiekiamas ne vien technine trynimo komanda, bet tokia raktų valdymo struktūra, kuri panaikina galimybę atkurti iššifravimui reikalingą raktų grandinę.

2.3. Metodo koncepcinė schema

Siūlomo metodo koncepcinė schema grindžiama principu, kad duomenų turinys, raktų informacija ir papildoma naudotojo valdoma rakto medžiaga negali būti laikomi vienoje vietoje. Toks atskyrimas mažina riziką, kad vieno komponento kompromitavimas sudarytų pakankamą pagrindą pradiniams duomenims atkurti. Dėl šios priežasties metodo logika orientuota ne į vieną centralizuotą saugojimo tašką, bet į kelių logiškai atskirtų sluoksnių sąveiką.



15 pav. Siūlomo kriptografinių raktų valdymo metodo koncepcinė schema

Kaip matyti 15 paveiksle, pradinis failo turinys P pirmiausia apsaugomas duomenų šifravimo raktu DEK. Užšifravus failą gaunamas šifruotas tekstas C, kuris saugomas duomenų saugojimo sluoksnyje. Pats DEK nėra saugomas atviru pavidalu – jis papildomai apsaugomas raktų šifravimo raktu KEK, todėl susidaro užšifruoto duomenų šifravimo rakto reikšmė EDEK. Ši reikšmė kartu su susijusiais metaduomenimis laikoma atskirame raktų valdymo sluoksnyje.

Svarbi šios schemos savybė yra ta, kad KEK nėra laikomas kaip vienas vientisas ilgalaikis objektas. Jo atkūrimas priklauso nuo dviejų įvesčių: serverio pusėje saugomos rakto dalies S_{srv} ir naudotojo valdomos rakto dalies S_{usr}. Šios reikšmės kartu su papildomais parametrais, pavyzdžiui, druska $salt_k$, naudojamos raktų išvedimo funkcijoje. Tokiu būdu visas metodas grindžiamas ne vien rakto saugojimu, bet jo kontroliuojamu atkūrimu.

Schemoje galima išskirti tris pagrindinius loginius sluoksnius. Pirmasis sluoksnis yra duomenų sluoksnis, kuriame saugomas tik užšifruotas failo turinys C. Antrasis sluoksnis yra raktų ir metaduomenų sluoksnis, kuriame saugomas užšifruotas duomenų šifravimo raktas EDEK, serverio rakto dalis S_{srv}, druska $salt_k$ ir kita su rakto būsena susijusi informacija. Trečiasis sluoksnis yra naudotojo valdomas rakto komponentas S_{usr}, kuris nelaikomas serverio aplinkoje ir naudojamas tik tada, kai reikia atkurti prieigą prie failo.

Tokioje schemoje duomenų atkūrimas galimas tik tada, kai kartu prieinami visi būtini elementai: užšifruotas failas C, apsaugotas duomenų šifravimo raktas EDEK, serverio rakto dalis S_{srv} ir naudotojo rakto dalis S_{usr}. Iš šių komponentų atkuriamas KEK, tada atgaunamas DEK, ir tik po to gali būti iššifruotas pradinis turinys P. Nė vienas komponentas atskirai nesudaro pakankamo pagrindo duomenims atkurti.

Saugaus duomenų ištrynimo scenarijuje ši schema leidžia pirmiausia panaikinti rakto atkuriamumą. Jei pašalinama arba nebepasiekiama tampa bent viena kritinė reikšmė, pavyzdžiui, S_{srv} arba S_{usr}, tuomet KEK nebegali būti atkurtas. Dėl to nebegalima atkurti DEK, o tai reiškia, kad nebegalima iššifruoti ir atkurti pradinio failo turinio P. Tokiu būdu pagrindinis saugaus duomenų ištrynimo rezultatas pasiekiamas per kriptografiškai valdomą duomenų turinio nebeiššifruojamumą ir praktinį nebeatkuriamumą, o ne vien per loginį objekto pašalinimą.

Taigi metodo koncepcinė schema parodo, kad siūlomas sprendimas grindžiamas duomenų šifravimu, DEK ir KEK hierarchija, atskirtu raktų valdymu ir naudotojo valdoma papildoma rakto dalimi. Ši schema sudaro pagrindą detalesniam kriptografinės struktūros ir metodo srautų aprašymui.

2.4. Kriptografinė schema ir duomenų bei raktų srautai

Šiame metode saugus duomenų ištrynimas realizuojamas taikant kriptografinio duomenų ištrynimo principą, paremtą voko šifravimu ir iš kelių dalių išvedamu raktų šifravimo raktu. Pagrindinė schema remiasi tuo, kad failo turinys šifruojamas atskiru duomenų šifravimo raktu, o pats šis raktas papildomai apsaugomas raktų šifravimo raktu. Skirtingai nei tradiciniuose modeliuose, šiame metode raktų šifravimo raktas nėra nuolat saugomas kaip vientisas objektas vienoje vietoje. Jo atkūrimas priklauso nuo serverio pusėje saugomos rakto dalies ir naudotojo valdomos išorinės rakto dalies. Dėl to failo atkūrimas galimas tik tada, kai prieinamos abi šios sudedamosios dalys.

Toliau naudojami tokie žymėjimai:

P – pradinis failo turinys;
C – užšifruotas failo turinys;
DEK – duomenų šifravimo raktas;
KEK – raktų šifravimo raktas;
S_{srv} – serverio pusėje saugoma rakto dalis;
S_{usr} – naudotojo valdoma rakto dalis;
$salt_k$ – druska, naudojama raktui išvesti;
N_f – failo šifravimo vienkartinė reikšmė;
N_k – duomenų šifravimo rakto apsaugos vienkartinė reikšmė;
Tag_f ir Tag_k – vientisumo žymos;
AAD_f ir AAD_k – papildomi autentifikuojami duomenys.

2.4.1. Duomenų šifravimo rakto generavimas ir failo šifravimas

Kiekvienam į sistemą pateiktam failui pirmiausia sugeneruojamas naujas atsitiktinis duomenų šifravimo raktas. Daroma prielaida, kad naudojamas 256 bitų ilgio raktas:

$$DEK = Random(32\ bytes); \quad (2.1)$$

čia *DEK* – duomenų šifravimo raktas;
Random(32 bytes) – atsitiktinių 32 baitų reikšmės generavimo funkcija.

Toks pasirinkimas reiškia, kad kiekvienas failas šifruojamas atskiru raktu. Dėl to vieno failo kompromitavimas tiesiogiai nelemia kitų failų kompromitavimo, nes tarp jų nėra bendro duomenų šifravimo rakto.

Po to sugeneruojama failo šifravimo vienkartinė reikšmė:

$$N_f = Random(12\ bytes); \quad (2.2)$$

čia N_f – failo šifravimo vienkartinė reikšmė;
Random(12 bytes) – atsitiktinių 12 baitų reikšmės generavimo funkcija.

Ši reikšmė naudojama simetrinio šifravimo algoritme tam, kad net ir šifruojant skirtingus duomenis tuo pačiu raktu būtų gaunami skirtingi rezultatai. Šiame metode failo turinys šifruojamas naudojant AES-256-GCM algoritmą, kuris vienu metu užtikrina duomenų konfidencialumą ir vientisumo kontrolę.

Failo šifravimo operacija apibrėžiama taip:

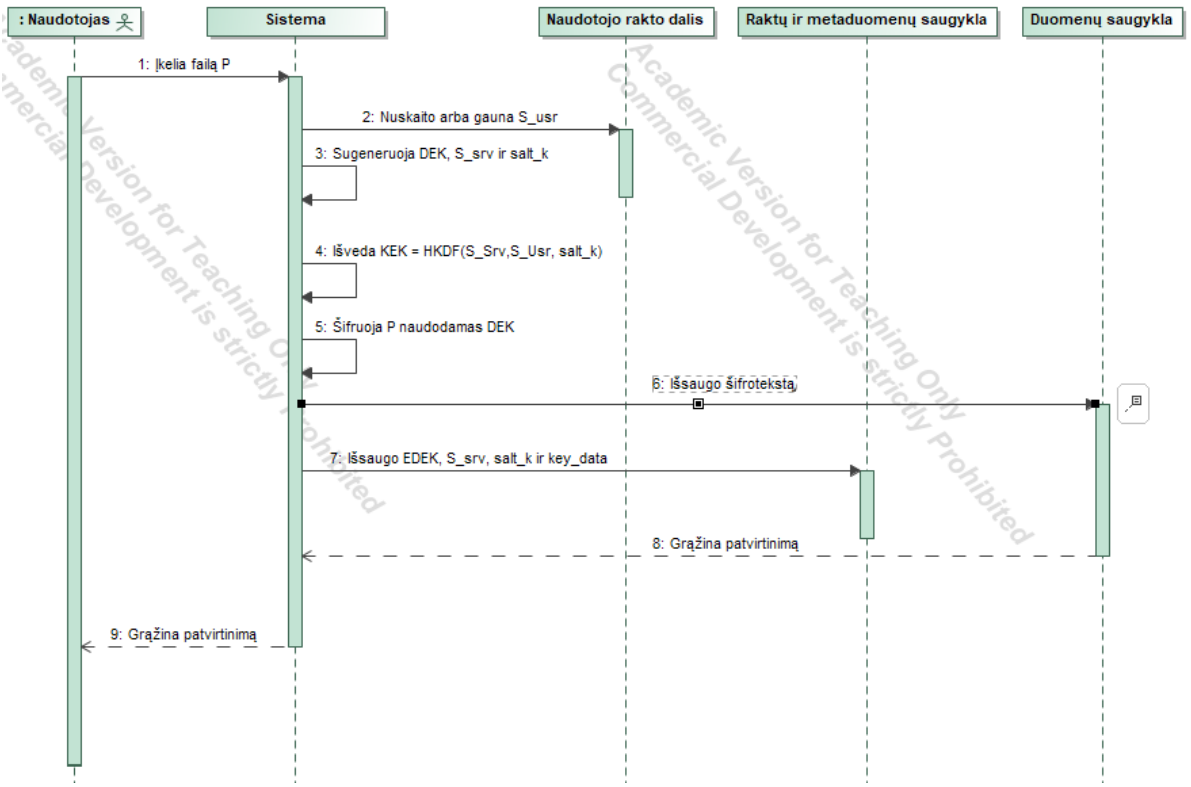
$$\left(C, Tag_f\right) = AES\text{-}GCM\text{-}Enc\left(DEK, N_f, P, AAD_f\right); \quad (2.3)$$

čia C – užšifruotas failo turinys;
Tag_f – failo vientisumo žyma;
AES-GCM-Enc – AES-GCM šifravimo funkcija;
DEK – duomenų šifravimo raktas;
N_f – failo šifravimo vienkartinė reikšmė;
P – pradinis failo turinys;
AAD_f – papildomi autentifikuojami failo duomenys.

Papildomais autentifikuojamais duomenimis AAD_f gali būti, pavyzdžiui, failo identifikatorius, savininko identifikatorius ar kiti metaduomenys, kurie nėra šifruojami, tačiau turi būti apsaugoti nuo nepastebimos modifikacijos.

Po šios operacijos pradinis failo turinys sistemoje nebelaikomas aktyvia forma. Į duomenų saugojimo sluoksnį perduodamas tik užšifruotas turinys, vientisumo žyma ir su šifravimu susiję techniniai parametrai. Tokiu būdu saugykloje nėra laikomas aiškus failo turinys, o saugomas tik jo užšifruotas turinys.

Siekiant parodyti praktinę failo įkėlimo, šifravimo ir išsaugojimo seką, toliau pateikiama šio proceso sekų diagrama.



16 pav. Failo įkėlimo, šifravimo ir išsaugojimo seka

Kaip matyti 16 paveiksle, naudotojas inicijuoja failo įkėlimą, sistema sugeneruoja reikalingus parametrus, užšifruoja pradinį turinį ir išsaugo užšifruotą failą bei su juo susijusius raktų duomenis atskiruose sluoksniuose. Taip jau įkėlimo metu realizuojamas pagrindinis metodo principas – duomenų ir raktų atskyrimas.

2.4.2. Raktų šifravimo rakto išvedimas

Sugeneruotas duomenų šifravimo raktas negali būti saugomas atviru pavidalu, todėl jis turi būti papildomai apsaugotas. Tam naudojamas raktų šifravimo raktas. Tačiau, siekiant išvengti situacijos, kai visas KEK būtų laikomas vienoje vietoje, šis raktas nėra nuolat saugomas kaip vientisas objektas. Vietoje to jo atkūrimas priklauso nuo dviejų dalių.

Pirmoji dalis sugeneruojama serverio pusėje:

$$S_{srv} = Random(32\ bytes); \qquad (2.4)$$

čia S_{srv} – serverio pusėje saugoma rakto dalis;
Random(32 bytes) – atsitiktinių 32 baitų reikšmės generavimo funkcija.

Antroji dalis priklauso naudotojui ir saugoma jo aplinkoje:

$$S_{usr} = Random(32\ bytes)\ ; \qquad (2.5)$$

čia S_{usr} – naudotojo valdoma rakto dalis;
Random(32 bytes) – atsitiktinių 32 baitų reikšmės generavimo funkcija.

Tam, kad iš šių dviejų dalių būtų galima deterministiškai išvesti tą patį raktų šifravimo raktą, papildomai sugeneruojama druska:

$$salt_k = Random(16\ bytes); \quad (2.6)$$

čia $salt_k$ – druska, naudojama raktų šifravimo raktui išvesti;
Random(16 bytes) – atsitiktinių 16 baitų reikšmės generavimo funkcija.

Tuomet taikoma raktų išvedimo funkcija:

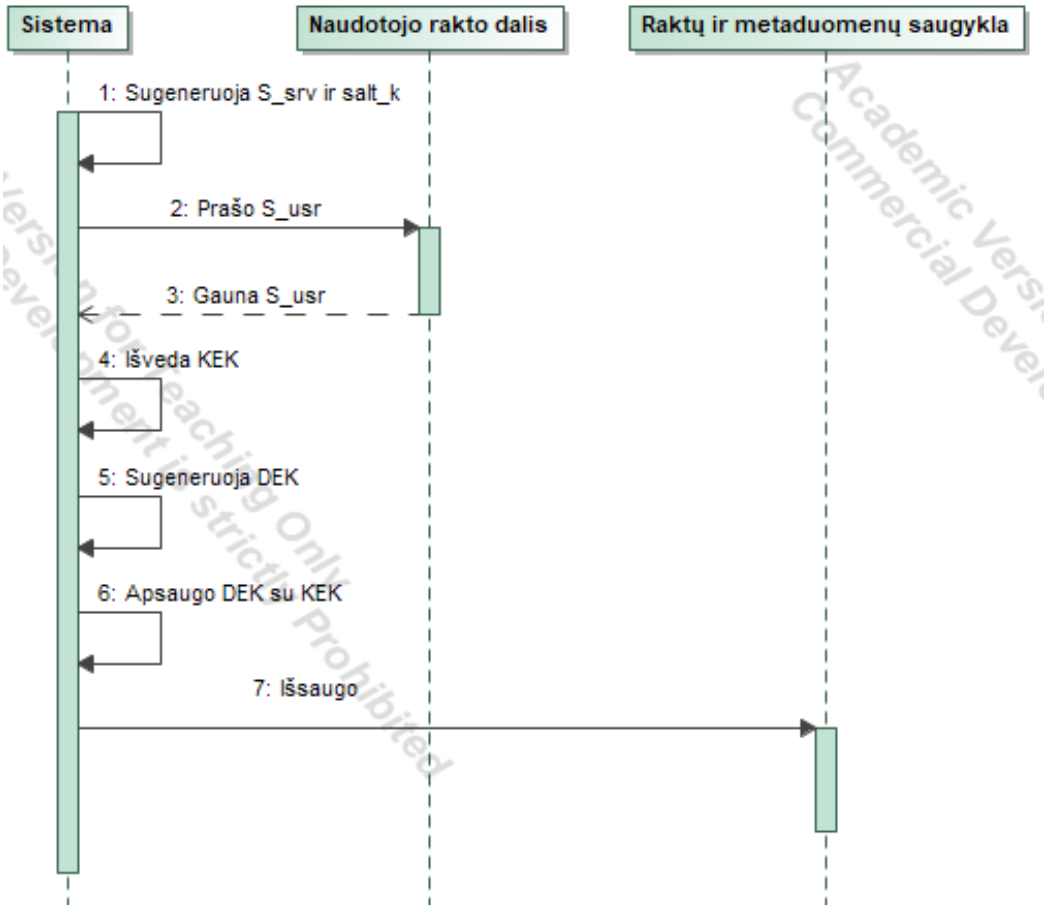
$$KEK = HKDF(salt_k, S_{srv} \parallel S_{usr}, info, 32); \quad (2.7)$$

čia *KEK* – raktų šifravimo raktas;
HKDF – raktų išvedimo funkcija;
$salt_k$ – druska, naudojama raktui išvesti;
S_{srv} – serverio pusėje saugoma rakto dalis;
$\parallel$ – reikšmių sujungimo operacija;
S_{usr} – naudotojo valdoma rakto dalis;
info – papildoma kontekstinė reikšmė;
32 – išvedamo rakto ilgis baitais.

Čia ‖ reiškia dviejų reikšmių sujungimą, o info žymi papildomą kontekstinę eilutę. Gauto rezultato ilgis yra 32 baitai, todėl KEK gali būti naudojamas kaip 256 bitų raktas.

Druskos naudojimas yra svarbus tuo, kad net ir turint tą pačią įvesties struktūrą raktų išvedimo rezultatas tampa susietas su konkrečiu failu ar konkrečiu kontekstu. Tai mažina pasikartojančių ir prognozuojamų reikšmių naudojimo riziką. Kartu svarbu pabrėžti, kad serverio rakto dalis viena pati nėra pakankama KEK atkurti, nes tam būtina ir naudotojo valdoma rakto dalis.

Kadangi raktų išvedimas yra centrinė siūlomo metodo dalis, šio proceso eiga papildomai iliustruojama sekų diagrama.



17 pav. Raktų išvedimo ir DEK apsaugos seka

Kaip matyti 17 paveiksle, raktų išvedimo ir DEK apsaugos procesas prasideda nuo serverio rakto dalies ir naudotojo rakto dalies parengimo. Šios dalys kartu su druska perduodamos raktų išvedimo funkcijai, kuri sugeneruoja KEK. Gavus KEK, juo apsaugomas duomenų šifravimo raktas DEK, todėl sistemoje toliau saugoma tik užšifruota DEK reikšmė EDEK, o neužšifruotas DEK nėra laikomas. Tokia seka leidžia atskirti failo šifravimui naudojamą raktą nuo aukštesnio lygmens rakto atkūrimo proceso ir sudaro pagrindą vėlesniam saugiam duomenų ištrynimui, kai panaikinus bent vieną KEK atkūrimui reikalingą dalį DEK nebegali būti atkurtas.

2.4.3. Duomenų šifravimo rakto apsauga

Kai KEK jau išvestas, juo apsaugomas duomenų šifravimo raktas. Pirmiausia sugeneruojama atskira vienkartinė reikšmė, naudojama būtent DEK apsaugai:

$$N_k = Random(12\ bytes); \qquad (2.8)$$

čia N_k – duomenų šifravimo rakto apsaugos vienkartinė reikšmė;
Random(12 bytes) – atsitiktinių 12 baitų reikšmės generavimo funkcija.

Tada pats DEK užšifruojamas naudojant tą patį AES-256-GCM algoritmą:

$$(EDEK, Tag_k) \leftarrow AES\text{-}GCM\text{-}Enc(KEK, N_k, DEK, AAD_k); \qquad (2.9)$$

čia *EDEK* – užšifruotas duomenų šifravimo raktas;
Tag_k – duomenų šifravimo rakto vientisumo žyma;
AES-GCM-Enc – AES-GCM šifravimo funkcija;
KEK – raktų šifravimo raktas;
N_k – duomenų šifravimo rakto apsaugos vienkartinė reikšmė;
DEK – duomenų šifravimo raktas;
AAD_k – papildomi autentifikuojami rakto duomenys.

Papildomi autentifikuojami duomenys AAD_k gali būti susieti su failo identifikatoriumi ar kitais raktų valdymo kontekstiniais duomenimis.

Po šio žingsnio sistemoje nebesaugomas neužšifruotas DEK. Vietoje jo saugoma tik užšifruota DEK reikšmė, kurio neįmanoma panaudoti neatkūrus KEK. Tokiu būdu realizuojamas voko šifravimo principas: pats failas šifruojamas DEK, o DEK papildomai apsaugomas aukštesnio lygio raktu.

2.4.4. Duomenų ir raktų paskirstymas tarp prototipo komponentų

Po šifravimo ir rakto apsaugos duomenys paskirstomi tarp kelių metodo komponentų. Toks paskirstymas yra viena svarbiausių architektūrinių saugumo prielaidų, nes nė vienas atskiras komponentas neturi visos informacijos, reikalingos failui atkurti.

Duomenų saugojimo sluoksnyje saugomi šie duomenys:

$$\{file_{id}, C, N_f, Tag_f\}; \qquad (2.10)$$

čia $file_{id}$ – failo identifikatorius;
C – užšifruotas failo turinys;
N_f – failo šifravimo vienkartinė reikšmė;
Tag_f – failo vientisumo žyma.

Tai reiškia, kad duomenų saugojimo sluoksnyje laikomas tik užšifruotas failas ir jo techniniai parametrai, tačiau nėra rakto, kuriuo būtų galima iššifruoti turinį.

Raktų ir metaduomenų mazge saugomi šie duomenys:

$$\{file_{id}, EDEK, N_k, Tag_k, salt_k, S_{srv}, key_{state}\}; \qquad (2.11)$$

čia $file_{id}$ – failo identifikatorius;
EDEK – užšifruotas duomenų šifravimo raktas;
N_k – duomenų šifravimo rakto apsaugos vienkartinė reikšmė;
Tag_k – duomenų šifravimo rakto vientisumo žyma;
$salt_k$ – druska, naudojama raktų šifravimo raktui išvesti;
S_{srv} – serverio pusėje saugoma rakto dalis;
key_{state} – rakto būsenos informacija.

Taigi šiame sluoksnyje laikomas užšifruotas DEK, serverio rakto dalis, druska ir rakto būsenos informacija, tačiau viena pati ši informacija taip pat nėra pakankama failui atkurti, nes trūksta naudotojo rakto dalies.

Naudotojo aplinkoje saugoma:

$\{S_{usr}\}$; (2.12)

čia S_{usr} – naudotojo valdoma rakto dalis.

Tokiu būdu naudotojo valdoma rakto dalis lieka už serverio infrastruktūros ribų. Būtent tai sudaro papildomą saugumo sluoksnį ir leidžia teigti, kad failo atkūrimas priklauso ne tik nuo serverio infrastruktūros, bet ir nuo naudotojo pusėje saugomos paslapties.

2.4.5. Failo atkūrimo srautas

Kai naudotojas nori atkurti failą, pirmiausia iš failų saugyklos mazgo gaunamas užšifruotas failas ir su juo susiję parametrai. Tuo pat metu iš raktų ir metaduomenų mazgo gaunami užšifruotas duomenų šifravimo raktas, serverio rakto dalis, druska ir kiti būtini metaduomenys.

Naudotojui pateikus savo rakto dalį, pakartotinai išvedamas raktų šifravimo raktas:

$$KEK = HKDF(salt_k, S_{srv}, \| S_{usr}, info, 32); \quad (2.13)$$

čia KEK – raktų šifravimo raktas;
$HKDF$ – raktų išvedimo funkcija;
$salt_k$ – druska, naudojama raktui išvesti;
S_{srv} – serverio pusėje saugoma rakto dalis;
S_{usr} – naudotojo valdoma rakto dalis;
// – reikšmių sujungimo operacija;
$info$ – papildoma kontekstinė reikšmė;
32 – išvedamo rakto ilgis baitais.

Tuomet atkuriamas duomenų šifravimo raktas:

$$DEK = AES\text{-}GCM\text{-}Dec(KEK, N_k, EDEK, Tag_k, AAD_k); \quad (2.14)$$

čia DEK – atkurtas duomenų šifravimo raktas;
$AES\text{-}GCM\text{-}Dec$ – AES-GCM iššifravimo funkcija;
KEK – raktų šifravimo raktas;
N_k – duomenų šifravimo rakto apsaugos vienkartinė reikšmė;
$EDEK$ – užšifruotas duomenų šifravimo raktas;
Tag_k – duomenų šifravimo rakto vientisumo žyma;
AAD_k – papildomi autentifikuojami rakto duomenys.

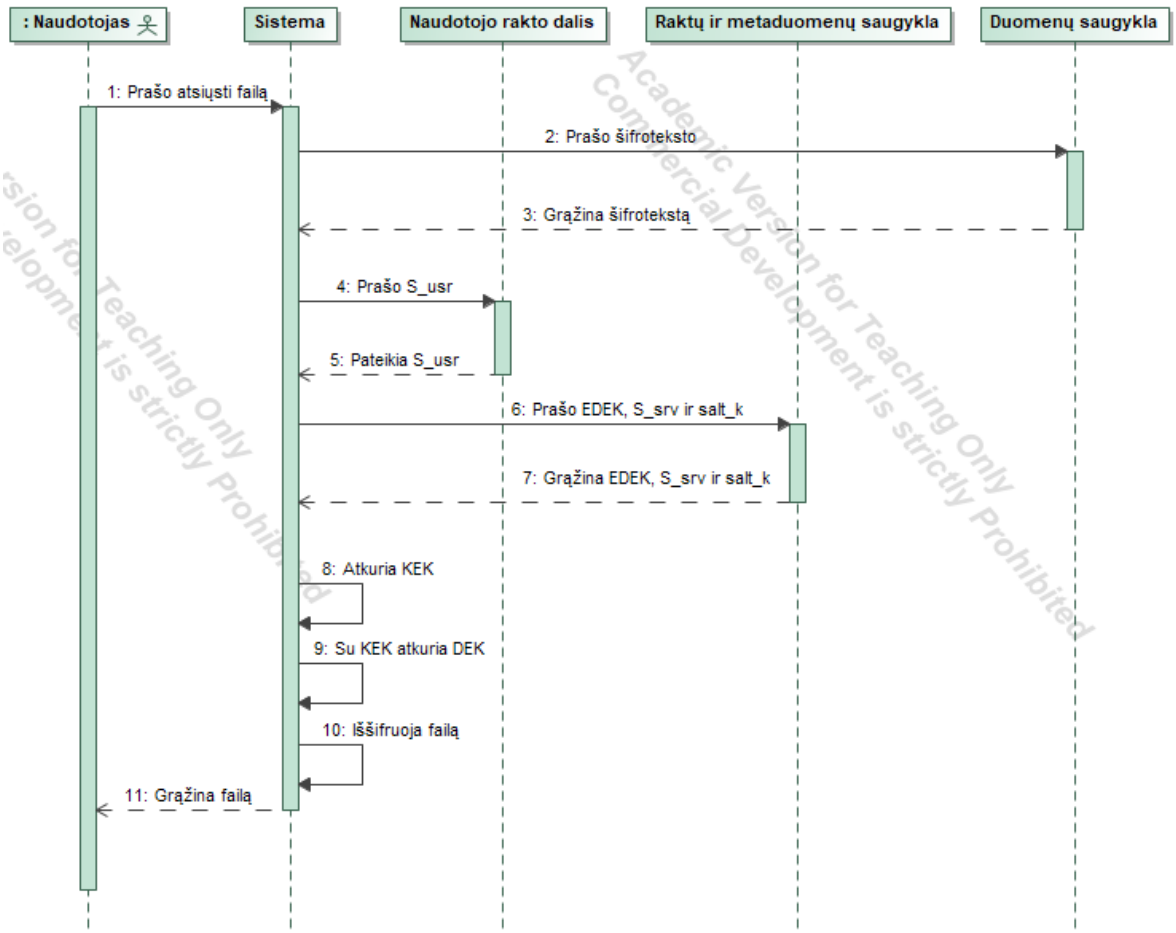
Jei ši operacija sėkminga ir vientisumo žyma atitinka, laikoma, kad DEK atkurtas korektiškai. Po to iššifruojamas pats failas:

$$P = AES\text{-}GCM\text{-}Dec(DEK, N_f, C, Tag_f, AAD_f); \quad (2.15)$$

čia P – pradinis failo turinys;
$AES\text{-}GCM\text{-}Dec$ – AES-GCM iššifravimo funkcija;
DEK – duomenų šifravimo raktas;
N_f – failo šifravimo vienkartinė reikšmė;
C – užšifruotas failo turinys;
Tag_f – failo vientisumo žyma;
AAD_f – papildomi autentifikuojami failo duomenys.

Tik po šių dviejų nuoseklių veiksmų naudotojas gali gauti pradinį failo turinį. Tai reiškia, kad failo atkūrimas galimas tik tuo atveju, kai kartu prieinami užšifruotas failas iš duomenų saugojimo sluoksnio, užšifruotas DEK ir serverio rakto dalis iš raktų ir metaduomenų sluoksnio bei naudotojo valdoma rakto dalis. Jei bent vienas iš šių komponentų neprieinamas, failo turinio atkurti neįmanoma.

Siekiant nuosekliai parodyti, kaip praktikoje vyksta šis procesas, toliau pateikiama failo atkūrimo sekų diagrama.



18 pav. Failo atsisiuntimo seka

Kaip matyti 18 paveiksle, failo atkūrimo metu sistema turi gauti šifruotą tekstą iš duomenų saugojimo sluoksnio, raktų duomenis iš raktų ir metaduomenų sluoksnio bei naudotojo valdomą rakto dalį. Tik po to galima atkurti KEK, iššifruoti DEK ir galiausiai atkurti pradinį failo turinį.

2.4.6. Saugaus naikinimo srautas

Saugaus ištrynimo metu svarbiausias veiksmas yra ne vien užšifruoto failo pašalinimas iš duomenų saugojimo sluoksnio, bet būtent rakto atkuriamumo panaikinimas. Kadangi KEK išvedamas iš dviejų dalių, pakanka pašalinti bent vieną iš jų, kad raktų grandinė taptų neatkuriama.

Tai galima išreikšti taip:

$$S_{srv} = \emptyset \; arba \; S_{usr} = \emptyset \Rightarrow KEK \; nebeatkuriamas$$

čia S_{srv} – serverio pusėje saugoma rakto dalis;
S_{usr} – naudotojo valdoma rakto dalis;
KEK – raktų šifravimo raktas;

Kadangi nebeatkuriamas KEK, nebegali būti iššifruotas užšifruotas duomenų šifravimo raktas:

$$KEK\ nebeatkuriamas\ \Rightarrow DEK\ nebeatkuriamas$$

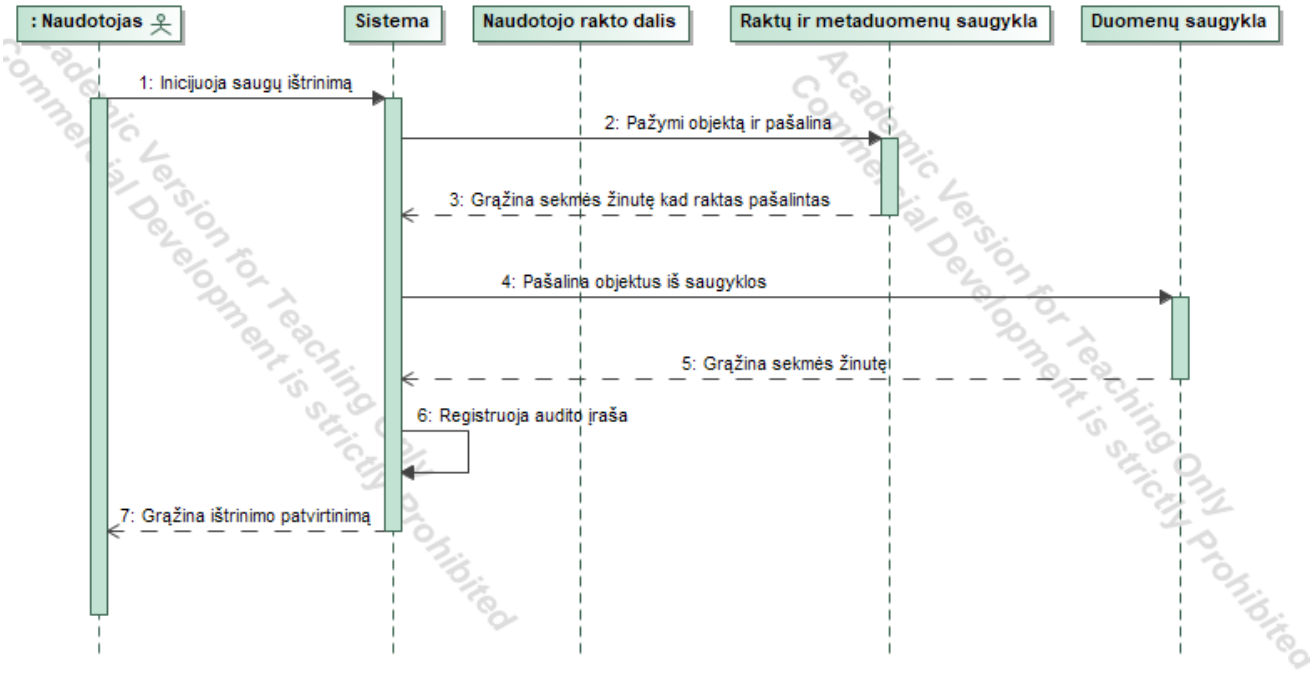
O tai reiškia, kad nebegali būti iššifruotas ir pats failo turinys:

$$DEK\ nebeatkuriamas\ \Rightarrow P\ nebeatkoduojamas$$

Ši logika ir sudaro kriptografinio duomenų ištrynimo esmę. Pirminis saugaus ištrynimo rezultatas pasiekiamas ne tada, kai fizinis failo objektas pašalinamas iš saugyklos, bet tada, kai tampa nebeįmanoma atkurti rakto, nuo kurio priklauso duomenų atkoduojamumas.

Praktinėje realizacijoje po rakto atkuriamumo panaikinimo papildomai atliekami ir antriniai veiksmai: užšifruotas failas pašalinamas iš aktyvios failų saugyklos, raktų ir metaduomenų sluoksnyje atnaujinama rakto būsena ir užregistruojamas audito įrašas apie atliktą ištrynimą. Tokiu būdu derinamos dvi logikos: pirminė kriptografinė nebeatkoduojamumo logika ir papildoma aktyvių duomenų likučių mažinimo logika.

Toliau pateikiama sekų diagrama, parodanti saugaus ištrynimo veiksmų seką.



19 pav. Saugaus duomenų ištrynimo seka

Kaip matyti 19 paveiksle, naudotojas inicijuoja ištrynimo veiksmą, po kurio sistema pirmiausia pažymi rakto duomenis kaip pašalintus arba nebeatkuriamus, vėliau pašalina aktyvų objektą iš duomenų saugyklos ir užregistruoja audito įrašą. Taip pirmiausia panaikinama duomenų iššifravimo

galimybė ir pasiekiamas praktinis jų turinio nebeatkuriamumas, o tik po to mažinami aktyvūs duomenų likučiai.

2.5. Metodo taikymo logika saugaus duomenų ištrynimo atveju

Pateikta kriptografinė schema rodo, kad siūlomas metodas duomenų apsaugą grindžia ne vien failo paslėpimu ar loginio ištrynimo operacija, bet kriptografiškai valdomu prieigos panaikinimu. Duomenų šifravimo raktas apsaugo konkretų failo turinį, o raktų šifravimo raktas apsaugo patį DEK. Tuo tarpu naudotojo valdoma rakto dalis užtikrina, kad raktų grandinės atkuriamumas nepriklausytų vien nuo serverio infrastruktūros.

Toks metodas tinkamas paskirstytai, debesijos ir virtualiajai infrastruktūrai dėl kelių priežasčių. Pirma, jis aiškiai realizuoja kriptografinio duomenų ištrynimo principą. Antra, jis leidžia parodyti praktinį skirtumą tarp paprasto failo pašalinimo ir failo turinio nebeiššifruojamumo bei praktinio nebeatkuriamumo. Trečia, jis suteikia pakankamai aiškią techninę schemą tolesniam sprendimo projektavimui, realizacijai ir eksperimentiniam tikrinimui.

Siūlomo metodo reikšmė saugiam duomenų ištrynimui yra ta, kad kritinis veiksmas tampa ne fizinis visų kopijų pašalinimas, bet galimybės atkurti raktų grandinę panaikinimas. Toks požiūris leidžia perkelti saugumo akcentą iš fiziškai sunkiai kontroliuojamos infrastruktūros į valdomą kriptografinį raktų gyvavimo ciklą. Dėl to metodas gali būti laikomas pagrįstu sprendimo branduoliu tais atvejais, kai siekiama ne formalaus ištrynimo, o praktinio duomenų nebeatkuriamumo.

2.6. Skyriaus apibendrinimas

Šiame skyriuje apibrėžtas siūlomas kriptografinių raktų valdymo metodas saugiam duomenų ištrynimui, jo paskirtis, veikimo logika, koncepcinė schema, kriptografinė struktūra ir pagrindiniai duomenų bei raktų srautai. Parodyta, kad metodas grindžiamas atskirtu duomenų saugojimu, DEK ir KEK hierarchija, iš kelių komponentų išvedamu raktų šifravimo raktu ir naudotojo valdoma papildoma rakto dalimi.

Atliktas metodo aprašymas leidžia teigti, kad pagrindinis saugaus duomenų ištrynimo rezultatas šiame darbe siejamas ne su vien loginio pašalinimo veiksmu, bet su kriptografiškai valdomu duomenų turinio nebeiššifruojamumu ir praktiniu nebeatkuriamumu. Būtent rakto atkuriamumo panaikinimas sudaro pagrindinę prielaidą tam, kad duomenų turinys taptų praktiškai neatkuriamas net ir tuo atveju, jei užšifruota jo kopija tam tikrą laiką išlieka saugojimo infrastruktūroje.

Šis skyrius sudaro loginį pagrindą pereiti prie kito darbo skyriaus, kuriame nagrinėjamas siūlomo metodo pagrindu projektuojamas prototipas, jo architektūra, projektiniai modeliai ir realizacijos planas.

3. Prototipo projektavimas ir realizacijos planas

3.1. Prototipo tikslas, paskirtis ir ribos

Šio prototipo tikslas – realizuoti ir praktiškai patikrinti siūlomą kriptografinių raktų valdymo metodą saugiam duomenų ištrynimui virtualiojoje infrastruktūroje. Prototipas turi parodyti, kad duomenų negrįžtamumas gali būti užtikrinamas ne vien failo pašalinimu iš aktyvios saugyklos, bet kriptografiškai panaikinant galimybę atkurti raktų grandinę, reikalingą pradiniam failo turiniui iššifruoti. Tokiu būdu prototipo paskirtis yra praktiškai patikrinti, ar siūlomas metodas gali būti realizuotas kaip funkciškai veikiantis sprendimas, kuriame užšifruoto failo egzemplioriaus išlikimas saugykloje nebėra lemiamas veiksnys duomenų atkuriamumui.

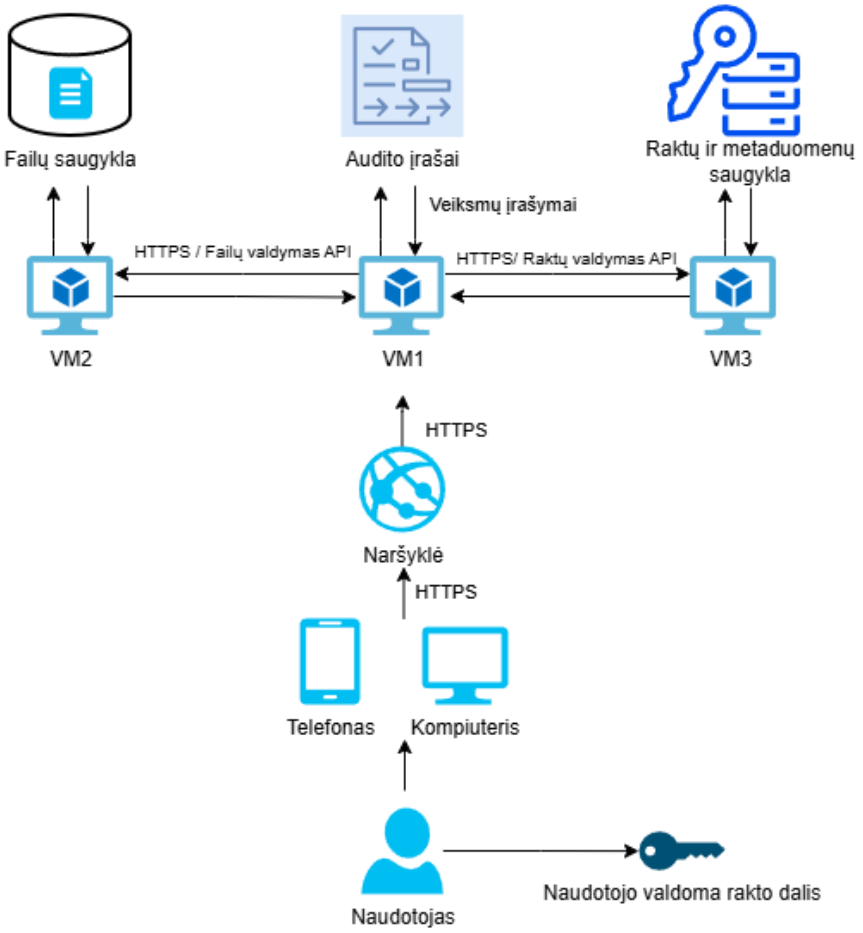
Prototipas skirtas pagrindiniams siūlomo metodo principams realizuoti. Jame turi būti įgyvendintas failo šifravimas atskiru duomenų šifravimo raktu, šio rakto apsauga raktų šifravimo raktu, atskirtas failų ir raktų metaduomenų saugojimas bei naudotojo valdomos papildomos rakto dalies panaudojimas. Be to, prototipas turi sudaryti galimybę praktiškai patikrinti keturis pagrindinius scenarijus: failo įkėlimą, užšifruoto failo saugojimą, failo atkūrimą ir saugų failo ištrynimą.

Šiame darbe kuriamas sprendimas yra funkcinis prototipas, todėl jo apimtis ir paskirtis yra ribotos. Jis nėra skirtas atkartoti viso komercinės debesijos paslaugos funkcionalumo, aukšto prieinamumo architektūros ar išplėstinio organizacinio valdymo modelio. Prototipe nekuriami pertekliniai komponentai, susiję su kelių duomenų centrų replikacija, automatiniu horizontaliuoju mastelio keitimu, išplėstiniu administravimo vaidmenų valdymu ar išsamia saugumo stebėsena. Pagrindinis dėmesys skiriamas metodo branduoliui – kriptografiškai valdomam duomenų turinio nebeiššifruojamumui ir praktiniam nebeatkuriamumui, realizuojamam per atskirtą duomenų saugojimą, raktų valdymą ir naudotojo kontroliuojamą rakto dalį.

Taigi šiame darbe prototipas suprantamas kaip eksperimentinė, tačiau funkciškai veikianti sistema, skirta siūlomam metodui pagrįsti ir sąlygoms tolesniam jo architektūriniam, technologiniam bei eksperimentiniam vertinimui sudaryti.

3.2. Prototipo architektūra ir komponentai

Prototipo architektūra grindžiama principu, kad failo turinys, raktų informacija ir naudotojo valdoma papildoma rakto dalis neturi būti laikomi vienoje vietoje. Toks atskyrimas leidžia sumažinti riziką, kad vieno komponento kompromitavimas sudarytų pakankamą pagrindą pradiniams duomenims atkurti. Dėl to siūloma architektūra orientuota ne į vieną centralizuotą saugojimo tašką, bet į kelių logiškai atskirtų komponentų sąveiką.



20 pav. Siūlomo saugaus duomenų naikinimo prototipo koncepcinė architektūra

Kaip matyti 20 paveiksle, prototipas sudarytas iš trijų pagrindinių komponentų ir naudotojo valdomos papildomos rakto dalies. Naudotojas prie sistemos jungiasi per žiniatinklio sąsają, naudodamas kompiuterį arba kitą kliento įrenginį. Pagrindinė naudotojo sąveika vyksta su taikomosios programos komponentu, kuris koordinuoja failų įkėlimo, šifravimo, atkūrimo ir ištrynimo procesus.

Pirmasis architektūros komponentas yra taikomosios programos sluoksnis. Šis komponentas veikia kaip pagrindinis prototipo valdymo centras. Jis priima naudotojo užklausas, vykdo sąveikos su žiniatinklio sąsaja logiką, inicijuoja failų šifravimą ir iššifravimą, koordinuoja raktų išvedimą bei valdo saugaus duomenų ištrynimo procesą. Taikomosios programos sluoksnis nėra skirtas ilgalaikiam failų ar visų raktų saugojimui. Jo paskirtis – vykdyti sprendimo logiką ir užtikrinti kontroliuojamą ryšį tarp kitų architektūros komponentų.

Antrasis komponentas yra duomenų saugojimo sluoksnis. Šis sluoksnis skirtas tik užšifruotiems failų objektams saugoti. Jame nelaikomas atviras failo turinys ir nėra saugomi visi raktai ar pakankama informacija failui iššifruoti. Duomenų saugojimo sluoksnio paskirtis yra saugoti užšifruotus failų objektus bei techninę informaciją, reikalingą jiems identifikuoti ir pasiekti. Toks sprendimas leidžia logiškai atskirti failo turinį nuo raktų valdymo informacijos.

Trečiasis komponentas yra raktų ir metaduomenų sluoksnis. Šiame sluoksnyje saugoma su failu susijusi valdymo informacija: užšifruotas duomenų šifravimo raktas, serverio pusėje naudojama rakto dalis, druska, failo identifikatoriai, rakto būsena ir kiti metaduomenys. Be to, šiame sluoksnyje

kaupiami pagrindiniai audito įrašai apie failo įkėlimą, atkūrimą ir ištrynimą. Vis dėlto vien šio sluoksnio nepakanka failui atkurti, nes jame nėra naudotojo valdomos papildomos rakto dalies.

Papildomas architektūros komponentas yra naudotojo valdoma rakto dalis. Ji nėra nuolat integruota į serverio infrastruktūrą ir naudojama tik tada, kai reikia atkurti prieigą prie failo. Tokia rakto dalis gali būti saugoma naudotojo pasirinktoje saugioje aplinkoje, pavyzdžiui, kaip apsaugotas rakto failas asmeninėje saugykloje. Šis komponentas yra viena svarbiausių prototipo savybių, nes failo atkūrimas priklauso ne vien nuo serverio infrastruktūros, bet ir nuo naudotojo kontroliuojamos papildomos paslapties.

Ryšys tarp naudotojo ir taikomosios programos sluoksnio vyksta per HTTPS. Vidinė komunikacija tarp taikomosios programos sluoksnio, duomenų saugojimo sluoksnio ir raktų bei metaduomenų sluoksnio vyksta saugiais ryšio kanalais. Toks modelis leidžia apsaugoti tiek perduodamą failo turinį, tiek raktų valdymo informaciją komunikacijos metu.

Architektūros veikimo logika grindžiama tuo, kad failo atkūrimas galimas tik tada, kai vienu metu prieinami trys pagrindiniai elementai: užšifruotas failas iš duomenų saugojimo sluoksnio, užšifruotas duomenų šifravimo raktas ir susiję metaduomenys iš raktų bei metaduomenų sluoksnio, taip pat naudotojo valdoma papildoma rakto dalis. Nė vienas iš šių komponentų atskirai nesudaro pakankamo pagrindo failo turiniui atkurti.

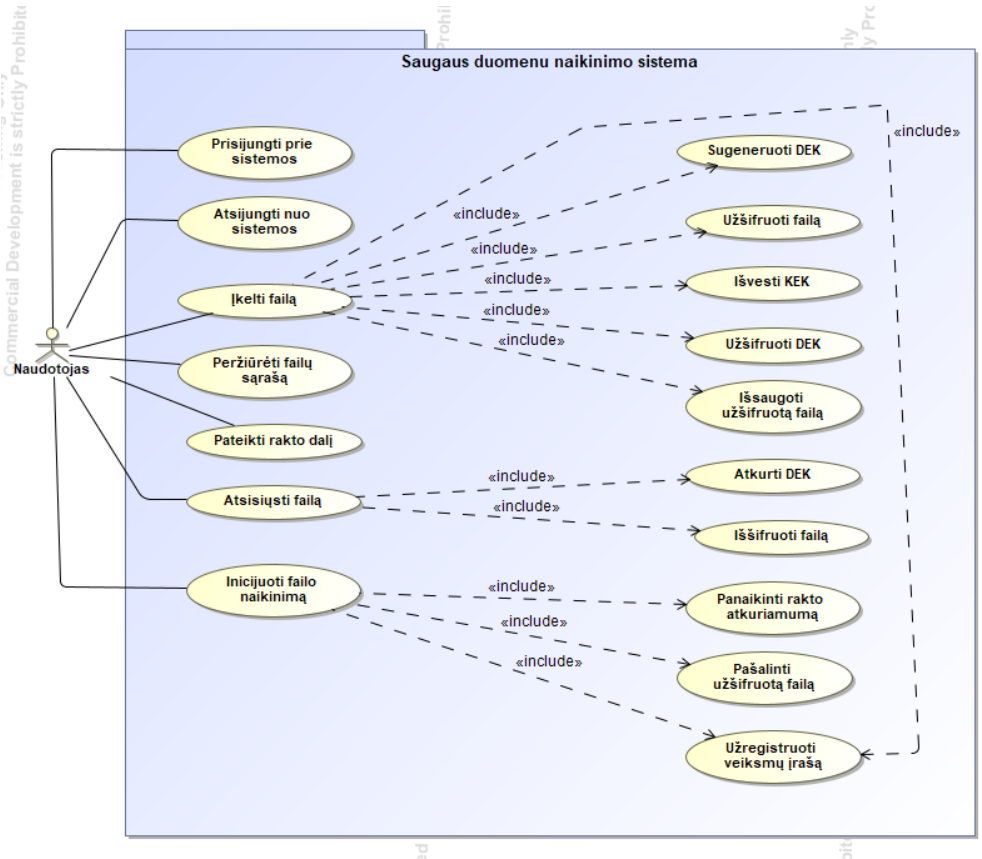
Ištrynimo scenarijuje ši architektūra leidžia pirmiausia panaikinti rakto atkuriamumą, o tik po to pašalinti aktyvius failo ir metaduomenų likučius. Dėl to net ir tuo atveju, jei užšifruoto failo kopija tam tikrą laiką išliktų saugykloje, jos turinys nebegalėtų būti iššifruotas ir praktiškai atkurtas į pradinę formą. Tokiu būdu koncepcinė prototipo architektūra tiesiogiai įgyvendina pagrindinę darbo idėją: saugus duomenų ištrynimas pirmiausia pasiekiamas per kriptografiškai valdomą duomenų turinio nebeiššifruojamumą ir praktinį nebeatkuriamumą, o ne vien per loginį objekto pašalinimą.

3.3. Sistemos projektiniai modeliai

Siekiant aiškiai apibrėžti prototipo veikimą, duomenų struktūrą ir komponentų išdėstymą, šiame poskyryje pateikiami pagrindiniai sistemos projektiniai modeliai. Jie leidžia nuosekliai parodyti, kaip naudotojas sąveikauja su sistema, kokie pagrindiniai duomenų objektai joje egzistuoja ir kaip prototipo komponentai yra paskirstyti virtualiojoje infrastruktūroje. Kadangi šiame darbe kuriamas ribotos apimties funkcinis prototipas, modeliuojami tik tie aktoriai, procesai ir komponentai, kurie tiesiogiai dalyvauja prototipo veikime.

3.3.1. Panaudojimo atvejų diagrama

Panaudojimo atvejų diagrama parodo pagrindines prototipo funkcijas ir naudotojo sąveiką su sistema. Kadangi šiame darbe realizuojamas ribotos apimties funkcinis prototipas, diagramoje modeliuojamas vienas pagrindinis aktorius – naudotojas. Toks sprendimas pasirinktas todėl, kad prototipe nerealizuojami išplėstiniai administravimo, audito ar atitikties valdymo vaidmenys, o pagrindinis dėmesys skiriamas failo įkėlimo, atkūrimo ir saugaus ištrynimo scenarijams.



21 pav. Saugaus duomenų naikinimo prototipo panaudojimo atvejų diagrama

Kaip matyti 21 paveiksle, naudotojas gali prisijungti prie sistemos, atsijungti nuo jos, įkelti failą, peržiūrėti failų sąrašą, pateikti rakto dalį, atsisiųsti failą ir inicijuoti failo ištrynimą. Šie panaudojimo atvejai atspindi pagrindinę prototipo paskirtį – sudaryti galimybę saugiai įkelti duomenis, kontroliuoti jų prieinamumą ir užtikrinti jų turinio nebeiššifruojamumą bei praktinį nebeatkuriamumą ištrynimo atveju.

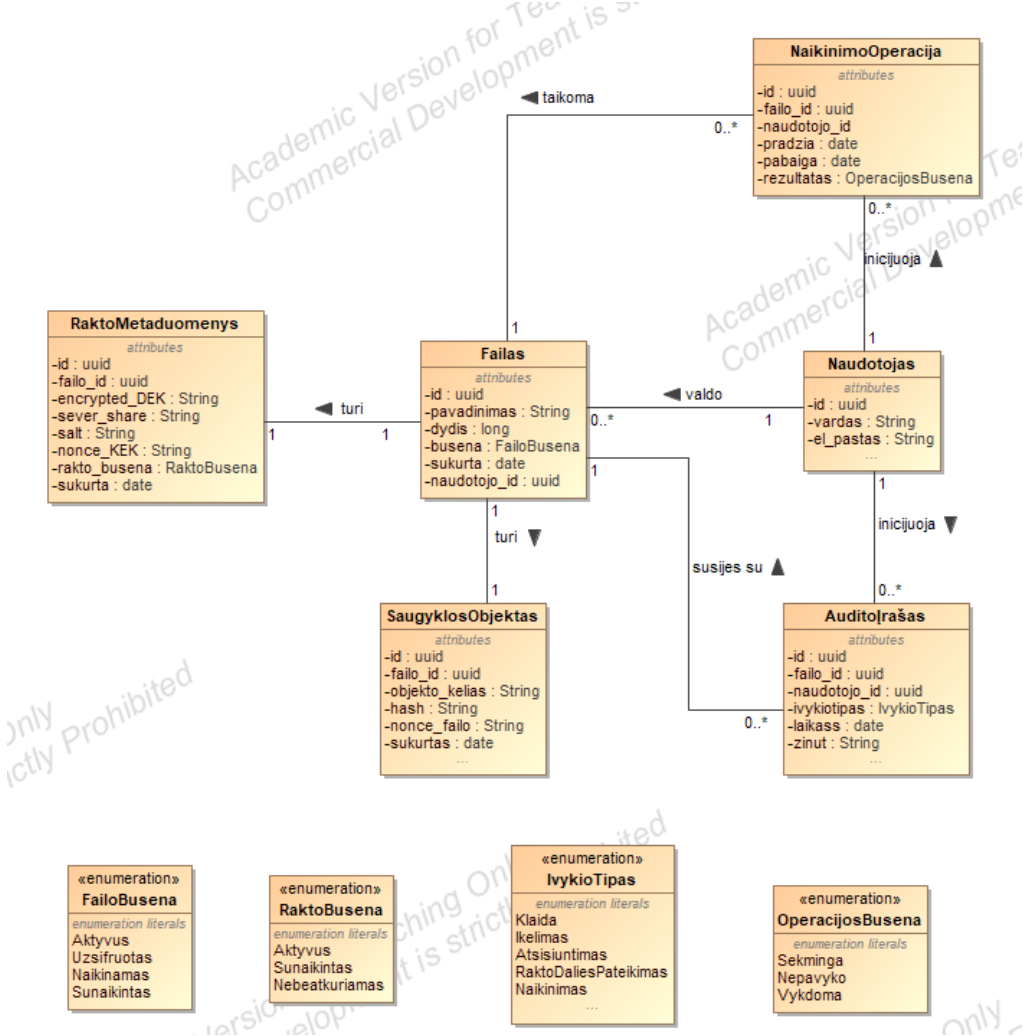
Failo įkėlimo panaudojimo atvejis susijęs su vidiniais sistemos veiksmais, kurie naudotojui nėra tiesiogiai matomi. Įkeliant failą sistema sugeneruoja duomenų šifravimo raktą, užšifruoja failo turinį, apsaugo šį raktą raktų šifravimo raktu ir išsaugo užšifruotą failą bei susijusius metaduomenis atskiruose komponentuose. Tai rodo, kad naudotojo veiksmas „įkelti failą“ iš tikrųjų inicijuoja sudėtinį saugojimo ir apsaugos procesą.

Failo atsisiuntimo panaudojimo atvejis susijęs su failo atkūrimo logika. Naudotojas negali tiesiogiai gauti failo iš saugyklos, nes pirmiausia turi būti atkurta reikalinga raktų grandinė. Dėl šios priežasties diagramoje išskiriamas papildomas panaudojimo atvejis „pateikti rakto dalį“. Jis atspindi svarbią prototipo savybę: failo atkūrimas priklauso ne tik nuo serverio infrastruktūroje saugomų duomenų, bet ir nuo naudotojo valdomos papildomos rakto dalies.

Failo ištrynimo panaudojimo atvejis apima ne tik objekto pašalinimą iš aktyvios saugyklos, bet ir rakto atkuriamumo panaikinimą. Būtent ši logika yra svarbiausia prototipo kontekste, nes ji leidžia pasiekti failo turinio nebeiššifruojamumą ir praktinį nebeatkuriamumą net tuo atveju, jei užšifruota jo kopija dar kurį laiką išlieka infrastruktūroje. Todėl failo ištrynimas šiame darbe suprantamas kaip kriptografiškai valdomas procesas, o ne vien paprasta trynimo operacija.

3.3.2. Konceptinis duomenų modelis

Konceptinis duomenų modelis apibrėžia pagrindines prototipo duomenų esybes, jų atributus ir tarpusavio ryšius. Kadangi šiame darbe kuriamas ribotos apimties funkcinis prototipas, modelyje paliekamos tik tos esybės, kurios tiesiogiai reikalingos failo įkėlimo, saugojimo, atkūrimo ir ištrynimo scenarijams realizuoti. Dėl šios priežasties modelyje atsisakoma perteklinių administravimo, atitikties ar politikų valdymo objektų ir koncentruojamasi į failą, jo saugojimo objektą, raktų metaduomenis, naudotoją, ištrynimo operaciją ir audito įrašus.



22 pav. Saugaus duomenų naikinimo prototipo konceptinis duomenų modelis

Kaip matyti 22 paveiksle, pagrindinė modelio esybė yra failas. Ji aprašo naudotojo įkeltą objektą ir saugo bendrą informaciją apie jo pavadinimą, dydį, būseną, sukūrimo laiką ir savininką. Su kiekvienu failu susiejamas vienas saugojimo objektas, kuris apibrėžia techninę užšifruoto failo saugojimo informaciją, tokią kaip objekto kelias, kontrolinė reikšmė ir su šifravimu susijęs pradinis vektorius arba vienkartinė reikšmė. Toks skaidymas leidžia atskirti loginį failo įrašą nuo fizinio jo saugojimo objekto.

Kita svarbi esybė yra raktų metaduomenys. Ji susieta su failu santykiu „vienas su vienu“ ir skirta raktų valdymo informacijai, reikalingai failui atkurti arba jo turinio nebeiššifruojamumui ir praktiniam nebeatkuriamumui užtikrinti, saugoti. Šioje esybėje laikomas užšifruotas duomenų šifravimo raktas, serverio pusėje naudojama rakto dalis, druska, papildoma su rakto išvedimu susijusi

reikšmė ir rakto būsena. Toks atskyrimas atitinka pagrindinį prototipo principą, kad užšifruotas failas ir jo raktų informacija neturi būti laikomi toje pačioje vietoje.

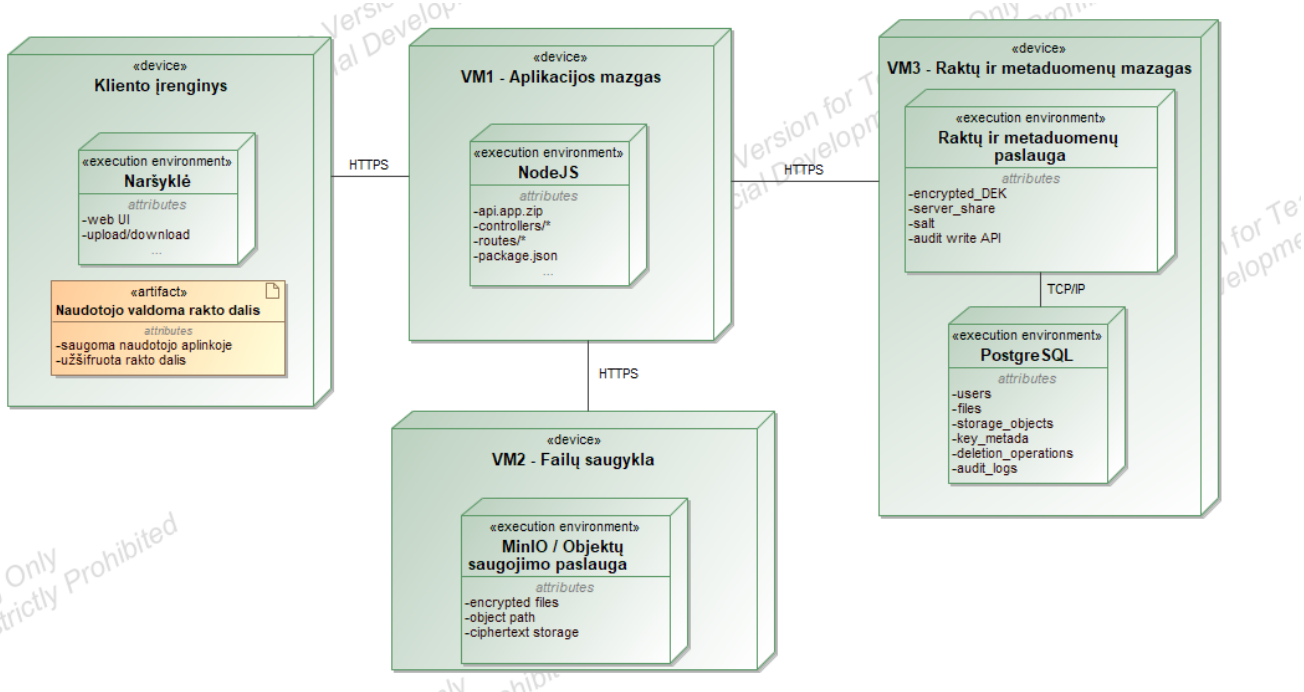
Modelyje taip pat išskiriama naudotojo esybė. Ji aprašo sistemos naudotoją ir saugo jo identifikatorių, vardą bei elektroninio pašto adresą. Vienas naudotojas gali valdyti kelis failus, todėl tarp naudotojo ir failo nustatomas „vienas su daugeliu“ ryšys. Be to, naudotojas inicijuoja ištrynimo operacijas ir generuoja audito įrašus, todėl jo identifikatorius naudojamas ir kitose modelio esybėse.

Failo ištrynimo procesui modeliuoti naudojama ištrynimo operacijos esybė. Ji saugo informaciją apie konkretų ištrynimo veiksmą: susieto failo identifikatorių, operaciją inicijavusio naudotojo identifikatorių, pradžios ir pabaigos laiką bei operacijos rezultatą. Tarp failo ir ištrynimo operacijos nustatomas „vienas su daugeliu“ ryšys, nes tam pačiam failui teoriškai gali būti registruojama daugiau nei viena su jo būsena susijusi operacija. Analogiškai vienas naudotojas gali inicijuoti daugiau nei vieną ištrynimo operaciją.

Svarbi modelio dalis yra ir audito įrašo esybė. Ji naudojama sistemos veiksmų atsekamumui užtikrinti ir saugo įvykio identifikatorių, susieto failo identifikatorių, naudotojo identifikatorių, įvykio tipą, laiką ir tekstinį pranešimą. Audito įrašai gali būti susiję tiek su failo įkėlimu, tiek su rakto dalies pateikimu, atsisiuntimu, ištrynimu ar klaidos fiksavimu. Tokiu būdu modelyje sukuriamas pagrindas minimaliam, bet pakankamam prototipo audituojamumui.

3.3.3. Diegimo diagrama

Diegimo diagrama parodo fizinį prototipo komponentų išdėstymą virtualiojoje infrastruktūroje ir jų tarpusavio ryšius. Skirtingai nuo konceptinio duomenų modelio, ši diagrama orientuota ne į duomenų esybes, bet į realius vykdymo mazgus, jų vidines aplinkas ir komunikacijos kanalus.



23 pav. Saugaus duomenų naikinimo prototipo diegimo diagrama

Kaip matyti 23 paveiksle, naudotojas prie sistemos jungiasi naudodamas kliento įrenginį, kuriame veikia naršyklė. Per šią naršyklę naudotojas pasiekia sistemos žiniatinklio sąsają, įkelia failus,

peržiūri jų sąrašą, atsisiunčia duomenis ir pateikia naudotojo valdomą rakto dalį, kai reikia atkurti prieigą prie failo. Ryšys tarp kliento įrenginio ir taikomosios programos sluoksnio vyksta per HTTPS, taip užtikrinant perduodamų duomenų konfidencialumą ir vientisumą.

Pagrindinis prototipo vykdymo mazgas yra taikomosios programos serveris. Jame realizuojama pagrindinė sistemos logika, atsakinga už naudotojo užklausų apdorojimą, failų įkėlimą, šifravimo ir iššifravimo operacijas, komunikaciją su failų saugykla ir raktų bei metaduomenų saugykla, taip pat saugaus ištrynimo proceso inicijavimą.

Duomenų saugojimo mazgas skirtas užšifruotų failų objektams saugoti. Šiame mazge laikomas tik užšifruotas failo turinys, objekto kelias ir kita techninė failo saugojimo informacija. Raktų ir metaduomenų mazgas skirtas raktų informacijai ir su failais susijusiems metaduomenims saugoti. Jame laikomas užšifruotas DEK, serverio rakto dalis, druska, rakto būsena ir susiję audito įrašai. Toks paskirstymas leidžia aiškiai atskirti failų saugojimo ir raktų valdymo sluoksnius.

3.4. Naudojamos technologijos ir realizavimo sprendimai

Prototipo realizacijai pasirinktos technologijos turi užtikrinti tris pagrindinius tikslus: galimybę praktiškai realizuoti kriptografinį failų apsaugos ir ištrynimo modelį, aiškų sistemos komponentų atskyrimą virtualiojoje infrastruktūroje ir pakankamai paprastą įgyvendinimą magistro darbo apimtyje. Dėl šios priežasties šiame darbe pasirenkamos ne gamybinio lygio debesijos paslaugos, bet savarankiškai diegiami ir valdomi sprendimai, tinkami funkciniam prototipui kurti.

3.4.1. Taikomosios programos lygmens technologijos

Pagrindinis prototipo valdymo komponentas realizuojamas „Node.js“ aplinkoje. Ši technologija pasirinkta todėl, kad leidžia patogiai kurti žiniatinklio pagrindu veikiančias taikomąsias sistemas, apdoroti failų įkėlimo užklausas, valdyti maršrutizaciją ir realizuoti ryšį su kitais sistemos komponentais per taikomųjų programų sąsajas. Be to, „Node.js“ turi pakankamai brandžią bibliotekų ekosistemą darbui su kriptografija, HTTP užklausomis, failų įkėlimu ir duomenų bazėmis.

Taikomosios programos logika įgyvendinama naudojant „Express.js“ arba analogišką lengvą serverio karkasą. Toks sprendimas leidžia aiškiai atskirti maršrutus, valdiklius, paslaugų sluoksnį ir kriptografines funkcijas. Prototipo kontekste tai svarbu todėl, kad sistema turi ne tik priimti naudotojo užklausas, bet ir koordinuoti failų šifravimą, raktų apsaugą, failų atkūrimą ir saugaus ištrynimo veiksmus.

Naudotojo sąveikai su sistema naudojama žiniatinklio sąsaja, pasiekiama per naršyklę. Kadangi prototipas skirtas funkcinei logikai pademonstruoti, o ne visiškai išvystytai naudotojo patirčiai sukurti, sąsaja gali būti realizuojama naudojant paprastus HTML, CSS ir JavaScript komponentus arba lengvą serverio generuojamą puslapių logiką.

3.4.2. Kriptografinės realizacijos priemonės

Kriptografinė logika prototipe realizuojama naudojant standartines „Node.js“ kriptografines priemones arba papildomas patikimas bibliotekas. Failų turinio apsaugai pasirinktas AES-256-GCM algoritmas. Šis algoritmas pasirinktas todėl, kad vienu metu užtikrina duomenų konfidencialumą ir vientisumo kontrolę. Tai ypač svarbu prototipui, nes sistema turi ne tik užšifruoti failą, bet ir užtikrinti, kad jo turinys nebūtų nepastebimai pakeistas.

Raktų išvedimui naudojama HKDF arba kita lygiavertė raktų išvedimo funkcija. Šis pasirinkimas pagrįstas tuo, kad prototipe raktų šifravimo raktas nėra nuolat laikomas kaip vientisas objektas, bet išvedamas iš kelių įvesties komponentų. Druskos naudojimas šiame procese leidžia susieti išvestą raktą su konkrečiu failu ar konkrečiu kontekstu ir taip sumažinti pasikartojančių reikšmių naudojimo riziką.

Atsitiktinėms reikšmėms generuoti naudojamas kriptografiškai saugus pseudoatsitiktinių skaičių generatorius, kurį suteikia „Node.js“ aplinka. Ši priemonė naudojama duomenų šifravimo raktui, serverio rakto daliai, druskai ir vienkartinėms reikšmėms generuoti.

3.4.3. Failų saugojimo sprendimas

Užšifruotiems failams saugoti pasirenkama atskira objektų saugojimo paslauga, šiame prototipe realizuojama naudojant „MinIO“. Šis sprendimas pasirinktas todėl, kad „MinIO“ suteikia objektinės saugyklos veikimo logiką, yra lengvai diegiamas virtualiojoje mašinoje ir leidžia saugoti užšifruotus failus kaip atskirus objektus. Toks modelis yra artimas realioms debesijos objektinėms saugykloms, tačiau nereikalauja naudoti išorinių komercinių debesijos paslaugų.

„MinIO“ naudojimas šiame darbe pagrįstas tuo, kad jis leidžia aiškiai atskirti failų saugojimo sluoksnį nuo taikomosios programos ir raktų valdymo sluoksnio. Failų saugykla laiko tik užšifruotą turinį, objekto kelią ir techninę saugojimo informaciją. Ji neturi pakankamos raktų informacijos, todėl viena pati negali būti naudojama failo turiniui atkurti.

3.4.4. Metaduomenų ir raktų informacijos saugojimas

Failų metaduomenims, raktų apsaugos informacijai ir audito įrašams saugoti naudojama „PostgreSQL“ duomenų bazė. Ši technologija pasirinkta todėl, kad ji yra brandi, patikima, gerai dokumentuota ir tinkama struktūrizuotiems duomenims saugoti. Be to, „PostgreSQL“ patogiai palaiko reliacinį modelį, kuris šiame prototipe naudojamas naudotojų, failų, saugyklos objektų, raktų metaduomenų, ištrynimo operacijų ir audito įrašų tarpusavio ryšiams aprašyti.

Raktų ir metaduomenų sluoksnyje veikianti paslauga atsakinga už užšifruoto DEK, serverio rakto dalies, druskos, rakto būsenos ir susijusių metaduomenų valdymą. Toks sprendimas leidžia logiškai atskirti raktų valdymo informaciją nuo failų turinio.

3.4.5. Virtualizacijos ir infrastruktūros sprendimai

Prototipas realizuojamas virtualiųjų mašinų aplinkoje, nes tai leidžia aiškiai atskirti sistemos komponentus fiziniu ir loginiu lygmenimis. Pasirenkama trijų loginių sluoksnių architektūra: vienas sluoksnis skirtas taikomosios programos logikai, antras – failų saugyklai, trečias – raktų ir metaduomenų valdymui. Toks paskirstymas atitinka ankstesniuose poskyriuose apibrėžtą architektūrinę kryptį ir leidžia praktiškai parodyti duomenų, raktų ir valdymo informacijos atskyrimą.

Ryšiai tarp komponentų realizuojami naudojant saugius tinklo kanalus. Išorinis ryšys tarp kliento įrenginio ir taikomosios programos sluoksnio vyksta per HTTPS, o vidiniai ryšiai tarp taikomosios programos sluoksnio, failų saugyklos ir raktų bei metaduomenų sluoksnio taip pat apsaugomi transporto lygmens saugumo priemonėmis.

3.4.6. Naudotojo valdomos rakto dalies realizacija

Viena svarbiausių prototipo savybių yra tai, kad dalis rakto informacijos nėra nuolat saugoma serverio aplinkoje. Naudotojo valdoma rakto dalis realizuojama kaip atskiras apsaugotas failas, saugomas naudotojo pasirinktoje vietoje. Prototipo lygmeniu toks sprendimas gali būti įgyvendintas kaip failas, kurį naudotojas saugo savo kompiuteryje arba asmeninėje saugioje saugykloje ir pateikia sistemai tik tada, kai nori atkurti prieigą prie failo.

Šis sprendimas pasirinktas todėl, kad jis yra pakankamai paprastas praktiniam magistro darbo prototipui įgyvendinti, tačiau kartu leidžia išlaikyti pagrindinę darbo naujumo idėją. Sistemos raktų atkuriamumas tokiu atveju priklauso ne vien nuo serverio komponentų, bet ir nuo naudotojo valdomos papildomos rakto dalies.

3.4.7. Audito ir veiksmų registravimo sprendimai

Prototipe taip pat numatytas bazinis veiksmų registravimas. Audito informacija saugoma „PostgreSQL“ duomenų bazėje kaip atskiri įrašai, susiję su naudotoju, failu ir įvykio tipu. Tokiu būdu registruojami svarbiausi sistemos veiksmai, pavyzdžiui, failo įkėlimas, failo atsisiuntimo bandymas, rakto dalies pateikimas ar failo ištrynimo inicijavimas.

Šis sprendimas pasirinktas todėl, kad jis leidžia užtikrinti minimalią prototipo atsekamumo logiką be perteklinio sudėtingumo. Magistro darbo prototipui nėra būtina diegti viso SIEM ar centralizuoto įvykių stebėjimo sprendimo. Pakanka registruoti pagrindinius sistemos veiksmus taip, kad vėliau būtų galima parodyti, kokie procesai įvyko ir kaip tai susiję su failo gyvavimo ciklu bei saugaus ištrynimo scenarijumi.

3.5. Prototipo realizacijos planas

Prototipo realizacija šiame darbe planuojama etapais, siekiant nuosekliai pereiti nuo architektūrinio modelio prie veikiančio funkcinio sprendimo. Pirmajame etape turi būti parengta bazinė infrastruktūra: sukonfigūruoti taikomosios programos, duomenų saugojimo ir raktų bei metaduomenų sluoksniai, užtikrintas jų tarpusavio ryšys bei parengta pradinė duomenų bazės schema.

Antrajame etape turi būti įgyvendinta pagrindinė failo įkėlimo logika. Šiame etape realizuojamas failo priėmimas per žiniatinklio sąsają, duomenų šifravimo rakto generavimas, failo užšifravimas, DEK apsauga naudojant KEK, užšifruoto failo įrašymas į duomenų saugojimo sluoksnį ir metaduomenų išsaugojimas raktų bei metaduomenų sluoksnyje.

Trečiajame etape turi būti realizuota failo atkūrimo logika. Šiame etape sistema turi gebėti gauti užšifruotą failą, priimti naudotojo rakto dalį, atkurti KEK, iššifruoti DEK ir galiausiai iššifruoti patį failą. Taip pat šiame etape turi būti užtikrinama pagrindinė klaidų kontrolė ir vientisumo tikrinimas.

Ketvirtajame etape turi būti realizuotas saugaus ištrynimo scenarijus. Šiame etape sistema turi ne tik pašalinti aktyvų failo objektą iš saugyklos, bet pirmiausia panaikinti rakto atkuriamumą, atnaujinti rakto būseną metaduomenų sluoksnyje ir užregistruoti atitinkamą audito įrašą.

Penktajame etape turi būti atliktas prototipo funkcinis patikrinimas. Šiame etape tikrinami pagrindiniai naudotojo scenarijai: sėkmingas failo įkėlimas, sėkmingas failo atkūrimas su teisinga rakto dalimi, nesėkmingas atkūrimas be rakto dalies arba po ištrynimo bei korektiškas audito įrašų generavimas.

Toks etapinis realizacijos planas leidžia nuosekliai įgyvendinti siūlomą metodą ir praktiškai patikrinti, ar jo logika gali būti realizuota kaip funkciškai veikiantis prototipas.

3.6. Skyriaus apibendrinimas

Šiame skyriuje apibrėžtas siūlomo sprendimo prototipo tikslas, veikimo ribos, koncepcinė architektūra, pagrindiniai projektiniai modeliai, realizacijai pasirinkti technologiniai sprendimai ir etapinis realizacijos planas. Parodyta, kad prototipas grindžiamas atskirtu failų, raktų ir metaduomenų saugojimu bei naudotojo valdoma papildoma rakto dalimi, nuo kurios tiesiogiai priklauso failo atkuriamumas.

Suformuota prototipo struktūra leidžia praktiškai realizuoti 2 skyriuje apibrėžtą kriptografinių raktų valdymo metodą saugiam duomenų ištrynimui. Toks sprendimas sudaro pakankamą pagrindą tolesniam prototipo įgyvendinimo, testavimo ir eksperimentinio vertinimo aprašymui.

4. Prototipo testavimas ir eksperimentinis įvertinimas

Šio skyriaus tikslas – aprašyti sukurto prototipo realizaciją, kokybiškai įvertinti siūlomą kriptografinių raktų valdymo metodą kitų duomenų naikinimo būdų kontekste, apibrėžti testavimo strategiją ir eksperimentiškai įvertinti metodo korektiškumą, saugumo savybes bei praktinį pritaikomumą. Sukurtas prototipas realizuoja metodą, kuriame failo turinys šifruojamas naudojant atskirą duomenų šifravimo raktą DEK, o pats DEK papildomai apsaugomas raktų šifravimo raktu KEK. Skirtingai nuo sprendimų, paremtų išorinėmis raktų valdymo ar aparatinio saugumo modulio paslaugomis, šiame darbe KEK nėra laikomas kaip vientisas ilgalaikis raktas, bet išvedamas iš naudotojo rakto dalies S_{usr}, serverio rakto dalies S_{srv} ir druskos $salt_k$.

Pagrindinė šiame skyriuje tikrinama hipotezė yra tokia: panaikinus failui reikalingos raktų grandinės atkuriamumą, pradinio failo turinio atkurti neįmanoma net ir tuo atveju, jei užšifruotas failo objektas dar išlieka saugykloje. Ši hipotezė tiesiogiai atitinka siūlomo metodo esmę, pagal kurią saugus duomenų naikinimas realizuojamas kaip kriptografiškai valdomas duomenų turinio nebeiššifruojamumas ir praktinis nebeatkuriamumas.

Be pagrindinės hipotezės, šiame skyriuje taip pat vertinama, ar sistema korektiškai veikia įprastais scenarijais: ar failai sėkmingai įkeliami ir atkuriami, ar sistema aptinka neteisingą arba modifikuotą naudotojo rakto failą, ar vieno failo saugus ištrynimas nepaveikia kitų failų atkuriamumo ir ar papildomos prototipo funkcijos, tokios kaip kelių failų įkėlimas, aplankų naudojimas, masinės operacijos bei eksperimentinių duomenų vizualizavimas, veikia stabiliai.

4.1. Kokybinis siūlomo metodo vertinimas

Prieš pateikiant eksperimentinius rezultatus, tikslinga kokybiškai įvertinti siūlomą metodą kitų duomenų naikinimo būdų kontekste. Toks vertinimas leidžia pagrįsti, kodėl šiame darbe pasirinktas būtent kriptografinių raktų valdymu grindžiamas požiūris.

Loginis ištrynimas yra paprasčiausias ir praktiškai dažniausiai taikomas būdas, tačiau paskirstytoje ir virtualiojoje infrastruktūroje jis dažniausiai pašalina tik aktyvią nuorodą į duomenų objektą, o ne visas galimas jo kopijas, versijas ar tarpinius egzempliorius. Dėl to toks metodas neužtikrina pakankamo negrįžtamumo, kai fizinė laikmenos kontrolė yra ribota. Perrašymu pagrįsti metodai teoriškai leidžia pasiekti stipresnį rezultatą, tačiau virtualiojoje ar objektinėje saugojimo infrastruktūroje jų taikymas tampa problemiškas dėl saugojimo abstrakcijos, spartinančiosios atminties, replikacijos ir fizinio sluoksnio nepasiekiamumo naudotojui.

Centralizuotas kriptografinis naikinimas, kai visas atkuriamumas priklauso nuo vieno serverio rakto ar vieno raktų valdymo sluoksnio, yra geriau pritaikomas debesijos aplinkai negu loginis ištrynimas ar perrašymas, nes saugumo pagrindą perkelia nuo fizinio ištrynimo prie rakto gyvavimo ciklo valdymo. Tačiau tokie sprendimai išlieka jautrūs vieno raktų valdymo komponento kompromitavimui ir dažniausiai remiasi viena centralizuota paslaptimi.

Siūlomas metodas išsiskiria tuo, kad raktų šifravimo raktas nėra laikomas kaip vientisas objektas vienoje vietoje. Jis išvedamas iš naudotojo rakto dalies, serverio rakto dalies ir druskos. Tai reiškia, kad nei vien tik naudotojo pusėje esantis rakto failas, nei vien tik serverio duomenų bazė nesudaro pakankamo pagrindo failui atkurti. Toks sprendimas padidina atsparumą vieno komponento

kompromitavimui ir leidžia įgyvendinti atrankinį saugų ištrynimą net tais atvejais, kai keli failai siejami su tuo pačiu naudotojo rakto profiliu.

Šio metodo pranašumas ypač išryškėja virtualiojoje infrastruktūroje. Čia saugumo tikslas siejamas ne su fiziniu visų duomenų kopijų pašalinimu, bet su galimybės atkurti raktų grandinę panaikinimu. Toks požiūris yra geriau suderinamas su debesijos ir objektinės saugyklos logika, kur naudotojas dažnai negali tiesiogiai valdyti fizinio saugojimo sluoksnio. Kita vertus, siūlomas metodas yra sudėtingesnis realizacijos požiūriu, nes reikalauja tiksliai įgyvendinto raktų valdymo, naudotojo rakto failo vientisumo kontrolės, duomenų bei raktų sluoksnių atskyrimo ir nuoseklaus atkūrimo bei ištrynimo proceso realizavimo.

Todėl kokybinio vertinimo požiūriu siūlomas metodas gali būti laikomas stipresniu už loginį ištrynimą ir geriau pritaikytu paskirstytai infrastruktūrai negu perrašymu pagrįsti metodai. Jo pagrindinis privalumas yra atrankinis ir kriptografiškai valdomas duomenų turinio nebeiššifruojamumas bei praktinis nebeatkuriamumas, pasiekiamas nepaisant to, kad užšifruotas failo objektas kurį laiką gali išlikti saugykloje.

8 lentelė. Duomenų naikinimo metodų kokybinis palyginimas

Vertinimo kriterijus	**Loginis ištrynimas**	**Perrašymas**	**Kriptografinis naikinimas**	**Siūlomas metodas**
Tinkamumas virtualiajai infrastruktūrai	Žemas	Žemas	Aukštas	Aukštas
Nepriklausomumas nuo fizinės laikmenos kontrolės	Žemas	Žemas	Aukštas	Aukštas
Atrankinio ištrynimo galimybė	Žemas	Žemas	Vidutinis	Aukštas
Atsparumas vieno komponento kompromitavimui	Žemas	Žemas	Vidutinis	Aukštas
Tinkamumas objektinei saugyklai	Žemas	Žemas	Aukštas	Aukštas
Duomenų nebeiššifruojamumas išlaikant užšifruotą objektą	Žemas	Žemas	Aukštas	Aukštas
Realizacijos sudėtingumas	Žemas	Vidutinis	Vidutinis	Aukštas

Vertinimo reikšmių paaiškinimas: šioje lentelėje vertinimas žemas reiškia, kad metodas silpnai atitinka konkretų kriterijų, pavyzdžiui, yra priklausomas nuo fizinės laikmenos kontrolės arba neužtikrina realaus duomenų turinio nebeiššifruojamumo ir praktinio nebeatkuriamumo. Vertinimas vidutinis reiškia, kad metodas kriterijų tenkina iš dalies, tačiau turi tam tikrų ribojimų, pavyzdžiui, priklausomybę nuo centrinio raktų valdymo komponento. Vertinimas aukštas reiškia, kad metodas gerai atitinka kriterijų ir yra tinkamas virtualiajai, objektinei ar paskirstytai infrastruktūrai. Realizacijos sudėtingumo kriterijuje aukštas reiškia, kad metodą įgyvendinti yra sudėtingiau, nes reikia daugiau komponentų, raktų valdymo procedūrų ir audito mechanizmų.

Kaip matyti 8 lentelėje, loginis ištrynimas ir perrašymas yra silpniau pritaikomi virtualiajai bei objektinei infrastruktūrai, nes jų veiksmingumas priklauso nuo duomenų saugojimo sluoksnio ir fizinės laikmenos kontrolės. Loginis ištrynimas dažniausiai pašalina tik aktyvią prieigą prie objekto, tačiau neužtikrina, kad neliktų jo versijų, kopijų ar atsarginių kopijų. Perrašymas gali būti veiksmingesnis tradicinėje fizinėje infrastruktūroje, tačiau virtualiojoje ir debesijos aplinkoje naudotojas dažniausiai neturi galimybės tiesiogiai valdyti fizinių saugojimo blokų.

Kriptografinis naikinimas geriau atitinka paskirstytos infrastruktūros logiką, nes duomenų turinio nebeiššifruojamumas ir praktinis nebeatkuriamumas gali būti pasiekiamas sunaikinus arba padarius nebepasiekiamą raktą. Vis dėlto šio metodo atsparumas vieno komponento kompromitavimui yra tik vidutinis, nes raktų valdymas labiau priklauso nuo vieno centrinio komponento. Jeigu centrinė raktų valdymo sistema yra pažeidžiama arba netinkamai valdoma, gali kilti rizika visai raktų grandinei.

Siūlomas metodas įvertintas aukštai pagal atrankinio ištrynimo, duomenų turinio nebeiššifruojamumo, objektinės saugyklos ir virtualiosios infrastruktūros tinkamumo kriterijus, nes jis remiasi raktų grandinės atkuriamumo panaikinimu ir leidžia saugiai ištrinti konkretų failą nepaveikiant kitų failų. Failo turinys šifruojamas atskiru DEK, o pats DEK apsaugomas KEK, kuris nėra saugomas kaip vientisas ilgalaikis objektas. KEK atkuriamas tik tada, kai tam yra reikalingos rakto dalys, todėl vieno komponento kompromitavimas savaime dar nesuteikia galimybės atkurti duomenų.

Realizacijos sudėtingumo požiūriu siūlomas metodas įvertintas aukštai, nes jo įgyvendinimui reikia daugiau komponentų: raktų dalių valdymo, naudotojo rakto failo vientisumo tikrinimo, DEK ir KEK hierarchijos, metaduomenų valdymo, šifruotų objektų saugojimo ir audito mechanizmų. Tačiau šis didesnis sudėtingumas yra pagrįstas, nes leidžia pasiekti didesnį atsparumą, geresnį atrankinį ištrynimą ir aiškesnį duomenų turinio nebeiššifruojamumo bei praktinio nebeatkuriamumo pagrindimą.

Taigi kokybinis palyginimas rodo, kad siūlomas metodas geriau atitinka saugaus duomenų naikinimo virtualiojoje infrastruktūroje poreikius negu loginis ištrynimas ar perrašymu pagrįstas metodas. Lyginant su centralizuotu kriptografiniu naikinimu, siūlomas metodas suteikia didesnį atsparumą vieno komponento kompromitavimui ir geresnes atrankinio ištrynimo galimybes. Dėl šių priežasčių tolesniuose skyriuose siūlomas metodas vertinamas ne tik teoriškai, bet ir praktiškai, atliekant funkcinį, saugumo ir našumo eksperimentinį tyrimą.

4.2. Prototipo realizacijos aplinka

Eksperimentinis tyrimas atliktas naudojant lokalią virtualiąją aplinką, sukurtą asmeniniame kompiuteryje. Virtualizacijai naudota „VMware Workstation 16 Player“ platforma, kurioje sukonfigūruotos trys virtualiosios mašinos: taikomosios programos mazgas, duomenų saugyklos mazgas ir raktų bei metaduomenų mazgas. Toks paskirstymas pasirinktas siekiant praktiškai atkurti paskirstytos infrastruktūros logiką ir atskirti duomenų saugojimo, raktų valdymo bei taikomosios programos vykdymo funkcijas.

Taikomosios programos mazge įdiegta žiniatinklio taikomoji sistema, realizuota naudojant „Node.js“, „Express.js“ ir EJS šablonų sistemą. Šis komponentas atsakingas už naudotojų autentifikaciją, failų įkėlimą, kelių failų įkėlimą vienu metu, failų šifravimą, atkūrimą, aplankų

valdymą, masinį failų perkėlimą, pavienį ir masinį saugų ištrynimą, taip pat eksperimentinių duomenų rinkimą, peržiūrą ir eksportą.

Duomenų saugyklos mazge veikia „MinIO“ objektinė saugykla, kurioje laikomi tik užšifruoti failų objektai. Raktų ir metaduomenų mazge naudojama „PostgreSQL“ duomenų bazė, kurioje saugomi naudotojų duomenys, failų loginiai įrašai, raktų profiliai, užšifruoti DEK duomenys, serverio rakto dalys, druskos, rakto būsenos, aplankai, ištrynimo operacijų įrašai, audito žurnalai ir eksperimentinių matavimų įrašai.

Kriptografinės funkcijos realizuotos naudojant „Node.js“ kriptografines priemones. Failų ir DEK apsaugai naudotas AES-256-GCM algoritmas, KEK išvesti naudota HKDF funkcija, o atsitiktinėms reikšmėms generuoti – kriptografiškai saugus atsitiktinių baitų generatorius. Naudotojo rakto dalis realizuota rakto failo su .key plėtiniu pavidalu, kuris gali būti sugeneruojamas sistemos arba pateikiamas naudotojo. Papildomai realizuotas rakto failo vientisumo tikrinimas naudojant identifikatorių ir pranešimo autentiškumo kodą.

Eksperimentinė aplinka taip pat išplėsta matavimo ir vizualizavimo funkcijomis. Sistema automatiškai registruoja operacijų trukmes, svarbiausius eksperimentinius parametrus ir rezultatų būsenas, o surinkti duomenys gali būti peržiūrimi atskirame eksperimentų puslapyje, vaizduojami grafikais ir eksportuojami CSV bei XLSX formatais tolesnei analizei.

4.3. Testavimo strategija

Prototipo testavimas organizuojamas kaip eksperimentų rinkinys, apimantis funkcinius, kriptografinio vientisumo, kriptografinio naikinimo, atrankinio naikinimo, stabilumo ir veikimo trukmės bandymus. Eksperimentai orientuoti į tai, kad būtų įvertinta ne vien naudotojo sąsajos veikla, bet pirmiausia metodo logika ir jos praktinis įgyvendinimas.

Rezultatai vertinami dviem principais. Pirma, funkcinių ir saugumo savybių eksperimentai vertinami principu „įvykdyta“ arba „neįvykdyta“, papildomai fiksuojant klaidos pobūdį ir etapą, jei scenarijus nepavyksta. Antra, veikimo trukmės eksperimentams registruojami tikslūs laiko matavimai milisekundėmis su šešiais skaitmenimis po kablelio, o vėliau iš jų skaičiuojami vidurkiai ir sudaromi grafikai.

Svarbu pabrėžti, kad šio darbo veikimo trukmės analizė organizuota kaip dviejų veiksnių eksperimentinis tyrimas. Pirmasis nepriklausomas veiksnys yra failo dydis, o antrasis – naudotojo rakto medžiagos ilgis S_{usr}. Tokia tyrimo schema leidžia įvertinti ne tik kiekvieno veiksnio atskirą poveikį, bet ir jų galimą sąveiką. Tai reiškia, kad analizuojama ne tik tai, ar didesni failai apdorojami lėčiau, bet ir tai, ar skirtingo ilgio S_{usr} keičia šią priklausomybę.

Kadangi pagrindinis darbo tikslas yra pagrįsti metodo veikimą, pagrindinis dėmesys šiame skyriuje skiriamas šioms savybėms: failo atkūrimo korektiškumui prieš naikinimą, rakto failo vientisumo tikrinimui, AES-GCM vientisumo patikrai, atrankiniam vieno failo naikinimui, masinio naikinimo veikimui ir duomenų nebeatkuriamumui po rakto atkuriamumo panaikinimo.

Papildomos grafinės naudotojo sąsajos funkcijos, tokios kaip aplankų naudojimas ar masinis failų perkėlimas, šiame darbe vertinamos kaip pagalbinės prototipo funkcijos, didinančios realizacijos išbaigtumą, tačiau jos nelaikomos pagrindiniu metodo vertinimo objektu. Tuo tarpu eksperimentinių duomenų rinkimo, filtravimo ir duomenų eksporto funkcijos šiame skyriuje laikomos svarbiu metodo

tyrimo palaikymo sluoksniu, nes leidžia ne tik įvykdyti bandymus, bet ir sistemiškai analizuoti jų rezultatus.

4.4. Matuojami rodikliai ir eksperimentinių duomenų rinkimas

Siekiant ne tik kokybiškai, bet ir kiekybiškai įvertinti siūlomo metodo veikimą, prototipas papildytas eksperimentinių metrikų rinkimo funkcija. Kiekvienai pagrindinei operacijai – failo įkėlimui, failo atkūrimui ir saugiam ištrynimui – registruojama bendra operacijos trukmė ir atskirų etapų trukmės. Taip pat registruojami eksperimentiniai parametrai: failo tipas, failo dydis, naudotojo rakto medžiagos ilgis, rakto šaltinis ir operacijos rezultatas. Šie duomenys naudojami lentelėms, grafikams ir eksperimentinėms išvadoms sudaryti.

Failo įkėlimo metu registruojami šie rodikliai: bendras įkėlimo laikas, rakto tikrinimo arba sugeneravimo laikas, KEK išvedimo trukmė, failo šifravimo trukmė, DEK apsaugos trukmė, įrašymo į objektinę saugyklą laikas ir įrašymo į duomenų bazę laikas. Taip pat išsaugoma pradinio failo SHA-256 kontrolinė suma, kuri vėliau naudojama atkūrimo korektiškumui patvirtinti.

Failo atkūrimo metu registruojami naudotojo rakto failo tikrinimo laikas, KEK išvedimo trukmė, DEK atkūrimo trukmė, užšifruoto objekto atsisiuntimo iš saugyklos laikas, failo iššifravimo trukmė ir bendra atkūrimo trukmė. Be to, apskaičiuojama atkurtų duomenų SHA-256 kontrolinė suma ir nustatoma, ar ji sutampa su prieš įkėlimą išsaugota reikšme.

Saugaus ištrynimo metu registruojama rakto atkuriamumo panaikinimo trukmė, aktyvaus objekto pašalinimo iš saugyklos trukmė ir bendras ištrynimo veiksmo laikas. Tokiu būdu galima atskirai vertinti tiek pirminį kriptografinio naikinimo veiksmą, tiek papildomą aktyvaus objekto šalinimo etapą.

Surinkti duomenys saugomi atskiroje eksperimentinių matavimų lentelėje. Prototipas turi atskirą eksperimentų peržiūros puslapį, kuriame rezultatus galima filtruoti pagal operacijos tipą, rezultatą, konkretų failą ir pasirinktą metriką. Papildomai realizuoti iš anksto parengti grafikai, leidžiantys greitai atvaizduoti tipinius ryšius, pavyzdžiui, failo dydžio ir S_{usr} ilgio įtaką įkėlimo laikui, atkūrimo laikui arba KEK išvedimo trukmei. Eksperimentinius duomenis taip pat galima eksportuoti CSV ir XLSX formatais, todėl jie gali būti toliau apdorojami išorinėmis priemonėmis, pavyzdžiui, „Excel“.

4.5. Testiniai duomenys ir eksperimentinės sąlygos

Eksperimentams parinkti keli skirtingų tipų ir dydžių failai, siekiant patikrinti, kad siūlomo metodo veikimas nepriklauso nuo konkretaus turinio formato. Pagrindinei veikimo trukmės analizei pasirinkti neutralūs tekstiniai failai, kad papildomų skirtumų nekeltų sudėtinga failų struktūra ar specifinis turinio tipas. Funkciniams korektiškumo ir vientisumo bandymams papildomai naudoti tekstiniai, dokumentų ir vaizdo failai.

Pagrindinei veikimo trukmės analizei naudoti penki failų dydžiai: 10 KB, 100 KB, 1 MB, 10 MB ir 100 MB. Tokia skalė leidžia stebėti sistemos elgseną tiek su labai mažais, tiek su pakankamai dideliais failais, tačiau kartu neperkrauna eksperimentinės aplinkos. Kadangi eksperimentiniai paleidimai buvo žymimi pagal failo dydį ir naudotojo rakto medžiagos ilgį, praktiniame bandymų vykdyme naudoti tokie failų pavadinimai kaip *10kb_16b.txt, 10kb_24b.txt, 10kb_32b.txt, 10kb_64b.txt, 100kb_16b.txt* ir analogiški pavadinimai visiems kitiems dydžiams. Toks žymėjimas palengvina eksperimentinių paleidimų atsekamumą ir rezultatų grupavimą, nors metodologiškai

esminis veiksnys išlieka ne pats failo pavadinimas, bet užfiksuota failo dydžio ir S_{usr} ilgio kombinacija.

Papildomai prototipas buvo išplėstas taip, kad generuojamai naudotojo rakto medžiagai S_{usr} būtų galima pasirinkti keturis ilgius: 16, 24, 32 ir 64 baitų. Svarbu pabrėžti, kad šiame darbe kinta ne galutinio AES rakto ilgis, bet tik naudotojo rakto medžiagos ilgis, naudojamas kaip įvestis HKDF funkcijai. Galutinis KEK visais atvejais išvedamas kaip 32 baitų ilgio raktas. Toks eksperimentinis išplėtimas leidžia įvertinti, ar skirtingo ilgio S_{usr} turi apčiuopiamos įtakos metodo veikimo laikui ir suderinamumui.

Bendra eksperimentinė matrica sudaryta iš 20 kombinacijų: 5 failų dydžių ir 4 S_{usr} ilgio variantų. Kiekviena kombinacija kartota po 5 kartus. Taigi bendra įkėlimo ir atkūrimo operacijų apimtis sudarė po 100 bandymų kiekvienai pagrindinei operacijų grupei. Toks pasikartojimų skaičius pasirinktas tam, kad būtų galima apskaičiuoti vidurkius, sumažinti pavienių sistemos svyravimų įtaką ir aiškiau matyti tiek atskirų veiksnių poveikį, tiek jų sąveiką.

4.6. Eksperimentų aprašymai

4.6.1. Funkcinio korektiškumo eksperimentai

E1. Įkėlimo ir atkūrimo korektiškumas pagal SHA-256. Tikslas – patikrinti, kad iki saugaus ištrynimo failas gali būti atkurtas identiškas originalui. Vykdymas – naudotojas įkelia failą, sistema jį užšifruoja, o po to tas pats failas atsisiunčiamas pateikus tinkamą rakto failą su .key plėtiniu. Prieš įkėlimą ir po atsisiuntimo apskaičiuotos SHA-256 reikšmės palyginamos tarpusavyje. Laukiamas rezultatas – SHA-256 reikšmės sutampa, o failas atkuriamas sėkmingai.

E2. Skirtingų failų formatų korektiškumas. Tikslas – patikrinti, kad prototipas korektiškai veikia su skirtingais turinio tipais. Vykdymas – kartojamas E1 scenarijus naudojant TXT, PDF ir JPG arba PNG failus. Laukiamas rezultatas – visais atvejais failai atkuriami identiškai.

E3. Skirtingų failų dydžių korektiškumas. Tikslas – patikrinti, kad failo atkūrimo korektiškumas nepriklauso nuo failo dydžio. Vykdymas – kartojamas E1 scenarijus naudojant skirtingų dydžių failus. Laukiamas rezultatas – visais atvejais atkurtų failų SHA-256 reikšmės sutampa su originalais.

4.6.2. Rakto failo ir kriptografinio vientisumo eksperimentai

E4. Atkūrimo bandymas su netinkamu rakto failu. Tikslas – patikrinti, kad failo atkūrimas neįmanomas, jei pateikiamas netinkamas naudotojo rakto failas su .key plėtiniu. Vykdymas – įkeliamas failas su vienu rakto profiliu, vėliau bandoma jį atkurti naudojant kitam profiliui priklausantį rakto failą. Laukiamas rezultatas – failo atkūrimas nepavyksta, nes rakto identifikatorius nesutampa.

E5. Modifikuoto rakto failo vientisumo testas. Tikslas – patikrinti, kad sistema aptinka pakeistą rakto failą. Vykdymas – pakeičiama S_{usr}, identifikatoriaus arba autentiškumo kodo reikšmė rakto faile, po to bandoma atkurti failą. Laukiamas rezultatas – sistema atmeta rakto failą kaip pažeistą, o atkūrimas neįvyksta.

E6. Šifruoto teksto modifikacijos testas. Tikslas – patikrinti, kad AES-GCM aptinka failo šifruoto teksto pakeitimą. Vykdymas – modifikuojamas vienas baitas užšifruotame objekte ir po to bandoma

atlikti iššifravimą. Laukiamas rezultatas – iššifravimas nepavyksta, nes GCM autentifikacija neatitinka.

E7. Failo autentifikavimo žymos modifikacijos testas. Tikslas – patikrinti, kad neteisinga failo vientisumo žyma neleidžia atkurti duomenų. Vykdymas – pakeičiama *file_tag* reikšmė metaduomenyse ir bandoma atlikti atkūrimą. Laukiamas rezultatas – iššifravimas nepavyksta.

E8. Failo vienkartinės reikšmės arba inicijavimo vektoriaus modifikacijos testas. Tikslas – patikrinti, kad AES-GCM režime vienkartinė reikšmė yra kritiška korektiškam atkūrimui. Vykdymas – pakeičiama *file_nonce* reikšmė ir bandomas failo atkūrimas. Laukiamas rezultatas – iššifravimas nepavyksta.

E9. Užšifruoto DEK sluoksnio modifikacijos testai. Tikslas – patikrinti, kad DEK apsaugos sluoksnio vientisumas taip pat užtikrinamas. Vykdymas – atskirai pakeičiama edek, *edek_tag* arba *edek_nonce* reikšmė ir bandomas failo atkūrimas. Laukiamas rezultatas – DEK negali būti atkurtas, todėl failas nebeiššifruojamas.

4.6.3. Kriptografinio naikinimo ir atrankinio naikinimo eksperimentai

E10. Vieno failo saugus ištrynimas. Tikslas – patikrinti, kad po rakto atkuriamumo panaikinimo failas tampa nebeatkuriamas. Vykdymas – failas įkeliamas, patikrinama, kad jį galima atkurti, po to atliekamas saugus ištrynimas ir pakartotinai bandoma atlikti atkūrimą. Laukiamas rezultatas – po saugaus ištrynimo failas nebeatkuriamas.

E11. Negrįžtamumas išlaikant užšifruotą objektą. Tikslas – patikrinti pagrindinę hipotezę, kad failo turinio nebeiššifruojamumas ir praktinis nebeatkuriamumas pasiekiamas dar iki fizinio objekto pašalinimo. Vykdymas – po rakto atkuriamumo panaikinimo dirbtinai paliekamas užšifruotas objektas saugykloje arba analizuojama būsena iki jo pašalinimo, po to bandoma atkurti failą. Laukiamas rezultatas – failas nebeatkuriamas, nes KEK ir DEK nebegali būti atkurti.

E12. Atrankinio naikinimo testas naudojant bendrą rakto profilį. Tikslas – patikrinti, kad vieno failo saugus ištrynimas nepaveikia kitų failų, net jei jie susieti su tuo pačiu rakto profiliu. Vykdymas – keli failai įkeliami naudojant tą patį naudotojo rakto failą. Tada saugiai ištrinamas tik vienas iš jų ir tikrinama, ar likę failai vis dar atkuriami. Laukiamas rezultatas – ištrintas failas tampa nebeatkuriamas, tačiau kiti failai išlieka atkuriami.

E13. Masinio saugaus ištrynimo testas. Tikslas – patikrinti, kad kelių pažymėtų failų rakto atkuriamumas gali būti panaikintas vienu veiksmu. Vykdymas – pažymimi keli failai ir atliekamas masinis saugus ištrynimas. Laukiamas rezultatas – visi pažymėti failai tampa nebeatkuriami, o nepažymėti failai išlieka nepakitę.

4.6.4. Veikimo trukmės ir dviejų veiksnių eksperimentai

E14. Failo dydžio ir S_{usr} ilgio įtaka įkėlimo laikui. Tikslas – įvertinti, kaip failo dydis ir naudotojo rakto medžiagos ilgis veikia bendrą įkėlimo trukmę. Vykdymas – įkeliami skirtingų dydžių failai, naudojant 16, 24, 32 ir 64 baitų S_{usr} variantus, ir registruojama *total_ms* reikšmė. Laukiamas rezultatas – didėjant failo dydžiui bendra įkėlimo trukmė ilgėja, o S_{usr} ilgio įtaka išlieka palyginti nedidelė.

E15. Failo dydžio ir S_{usr} ilgio įtaka atkūrimo laikui. Tikslas – įvertinti, kaip abu veiksniai veikia failo atkūrimo trukmę. Vykdymas – atkuriami skirtingų dydžių failai su atitinkamais rakto profiliais ir registruojama *total_ms* reikšmė. Laukiamas rezultatas – didėjant failo dydžiui bendra atkūrimo trukmė ilgėja, o S_{usr} ilgio įtaka išlieka mažesnė negu failo dydžio.

E16. Failo dydžio ir S_{usr} ilgio įtaka KEK išvedimo trukmei. Tikslas – patikrinti, ar skirtingo ilgio naudotojo rakto medžiaga keičia KEK išvedimo trukmę ir ar ši trukmė priklauso nuo failo dydžio. Vykdymas – kartojami įkėlimo scenarijai naudojant visas 20 failo dydžio ir S_{usr} ilgio kombinacijų. Laukiamas rezultatas – S_{usr} ilgis gali turėti nedidelę įtaką *kek_derive_ms*, tačiau failo dydžio poveikis šiam etapui turėtų būti ribotas.

E17. Failo dydžio ir S_{usr} ilgio įtaka šifravimo bei iššifravimo trukmėms. Tikslas – nustatyti, kaip abu veiksniai veikia *file_encrypt_ms* ir *file_decrypt_ms* rodiklius. Vykdymas – kartojamos visos 20 kombinacijų, o rezultatai grupuojami pagal failo dydį ir S_{usr} ilgį. Laukiamas rezultatas – failo dydis yra pagrindinis šių rodiklių veiksnys, o S_{usr} ilgio įtaka išlieka minimali.

E18. Kartojamumo testas. Tikslas – patikrinti, ar prototipas veikia stabiliai kartojant tuos pačius scenarijus daug kartų. Vykdymas – kiekviena eksperimentinė failo dydžio ir S_{usr} ilgio kombinacija kartojama po 5 kartus. Laukiamas rezultatas – užfiksuoti laiko svyravimai išlieka nedideli, o klaidų dažnis artimas nuliui.

4.7. Eksperimentų rezultatų apdorojimas ir pateikimas

Eksperimentiniams rezultatams pateikti šiame darbe naudoti automatiškai prototipe surinkti duomenys. Kiekvienos pagrindinės operacijos – failo įkėlimo, failo atkūrimo ir saugaus ištrynimo – metu sistema registravo bendrą vykdymo trukmę ir atskirų etapų trukmes. Surinkti duomenys buvo saugomi duomenų bazėje, peržiūrimi prototipo eksperimentų puslapyje ir eksportuojami XLSX formatu tolesniam apdorojimui. Grafikai šiame skyriuje sudaryti remiantis būtent šiomis eksperimentinėmis reikšmėmis, todėl jie atspindi realų sukurto sprendimo veikimą, o ne teorines prielaidas.

Eksperimentinių rezultatų analizės aiškumui dalis surinktų matavimų šiame skyriuje papildomai pateikiama lentelėmis. Jose suvesti vidutiniai kiekvienos failo dydžio ir naudotojo rakto medžiagos ilgio kombinacijos rezultatai. Šios lentelės sudaro pagrindą toliau pateikiamiems grafikams ir leidžia sistemingai palyginti dviejų nepriklausomų veiksnių poveikį tiriamoms metrikoms.

Kadangi kiekviena failo dydžio ir S_{usr} ilgio kombinacija kartota po 5 kartus, tolesnėje analizėje naudojami šių pakartojimų vidurkiai. Toks sprendimas leidžia sumažinti pavienių vykdymo svyravimų įtaką ir aiškiau atskleisti bendras tendencijas. Pagrindinėje rezultatų dalyje pateikiami grafikai, kuriuose horizontalioji ašis rodo failo dydį, o skirtingos kreivės atspindi S_{usr} ilgio variantus. Tokia pateikimo forma leidžia vienu metu vizualiai įvertinti ir atskirą kiekvieno veiksnio poveikį, ir jų galimą sąveiką.

9 lentelė. Vidutinės failo įkėlimo trukmės priklausomai nuo failo dydžio ir S_{usr} ilgio

Failo dydis	**Rakto ilgis**			
	16 B	**24 B**	**32 B**	**64 B**
10 KB	7.457481 ms	6.665425 ms	6.574319 ms	5.99153 ms

100 KB	12.661935 ms	7.759428 ms	7.31786 ms	6.967709 ms
1 MB	18.249089 ms	19.494634 ms	18.075268 ms	23.034185 ms
10 MB	117.243603 ms	100.37829 ms	90.622302 ms	89.708028 ms
100 MB	1261.806878 ms	1150.070462 ms	1011.795339 ms	1041.07944 ms

Lentelėje pateikti vidutiniai upload *total_ms* rezultatai, apskaičiuoti iš penkių kiekvienos failo dydžio ir S_{usr} ilgio kombinacijos pakartojimų. Šie duomenys naudoti 24 paveikslui sudaryti.

10 lentelė. Vidutinės failo atkūrimo trukmės priklausomai nuo failo dydžio ir S_{usr} ilgio

Failo dydis	**Rakto ilgis**			
	16 B	**24 B**	**32 B**	**64 B**
10 KB	5.751947 ms	5.067166 ms	5.112868 ms	4.619077 ms
100 KB	5.734006 ms	5.797057 ms	6.052677 ms	5.136917 ms
1 MB	12.148378 ms	11.293186 ms	12.181819 ms	9.354901 ms
10 MB	58.809558 ms	58.973611 ms	56.344203 ms	52.577673 ms
100 MB	654.019284 ms	563.530005 ms	548.070009 ms	534.352646 ms

Lentelėje pateikti vidutiniai restore *total_ms* rezultatai, apskaičiuoti iš penkių kiekvienos failo dydžio ir S_{usr} ilgio kombinacijos pakartojimų. Šie duomenys naudoti 25 paveikslui sudaryti.

11 lentelė. Vidutinės KEK išvedimo trukmės priklausomai nuo failo dydžio ir S_{usr} ilgio

Failo dydis	**Rakto ilgis**			
	16 B	**24 B**	**32 B**	**64 B**
10 KB	0.125563 ms	0.069108 ms	0.093842 ms	0.067825 ms
100 KB	0.069377 ms	0.078378 ms	0.072541 ms	0.078257 ms
1 MB	0.064146 ms	0.106271 ms	0.067908 ms	0.081367 ms
10 MB	0.093334 ms	0.097377 ms	0.078523 ms	0.086603 ms
100 MB	0.111436 ms	0.093551 ms	0.109557 ms	0.110947 ms

Lentelėje pateikti vidutiniai *kek_derive_ms* rezultatai, apskaičiuoti iš penkių kiekvienos failo dydžio ir S_{usr} ilgio kombinacijos pakartojimų. Šie duomenys naudoti 26 paveikslui sudaryti.

12 lentelė. Vidutinės failo šifravimo trukmės priklausomai nuo failo dydžio ir S_{usr} ilgio

Failo dydis	**Rakto ilgis**			
	16 B	**24 B**	**32 B**	**64 B**
10 KB	0.113479 ms	0.086098 ms	0.041166 ms	0.042025 ms
100 KB	0.064235 ms	0.090107 ms	0.052836 ms	0.0902 ms
1 MB	0.541667 ms	1.841779 ms	0.666827 ms	0.659921 ms
10 MB	17.236988 ms	7.965149 ms	2.035882 ms	2.435004 ms
100 MB	199.631929 ms	105.34645 ms	105.58934 ms	105.237044 ms

Lentelėje pateikti vidutiniai *file_encrypt_ms* rezultatai, apskaičiuoti iš penkių kiekvienos failo dydžio ir S_{usr} ilgio kombinacijos pakartojimų. Šie duomenys naudoti 27 paveikslui sudaryti.

13 lentelė. Vidutinės failo iššifravimo trukmės priklausomai nuo failo dydžio ir S_{usr} ilgio

Failo dydis	Rakto ilgis			
	16 B	**24 B**	**32 B**	**64 B**
10 KB	0.036794 ms	0.036338 ms	0.031684 ms	0.034983 ms
100 KB	0.047366 ms	0.04825 ms	0.04427 ms	0.058892 ms
1 MB	0.215885 ms	0.194884 ms	0.189746 ms	0.182719 ms
10 MB	7.984749 ms	6.302031 ms	3.928667 ms	2.862089 ms
100 MB	103.486915 ms	104.760546 ms	105.124541 ms	103.551012 ms

Lentelėje pateikti vidutiniai *file_decrypt_ms* rezultatai, apskaičiuoti iš penkių kiekvienos failo dydžio ir S_{usr} ilgio kombinacijos pakartojimų. Šie duomenys naudoti 28 paveikslui sudaryti.

14 lentelė. Funkcinių ir saugumo eksperimentų rezultatų suvestinė

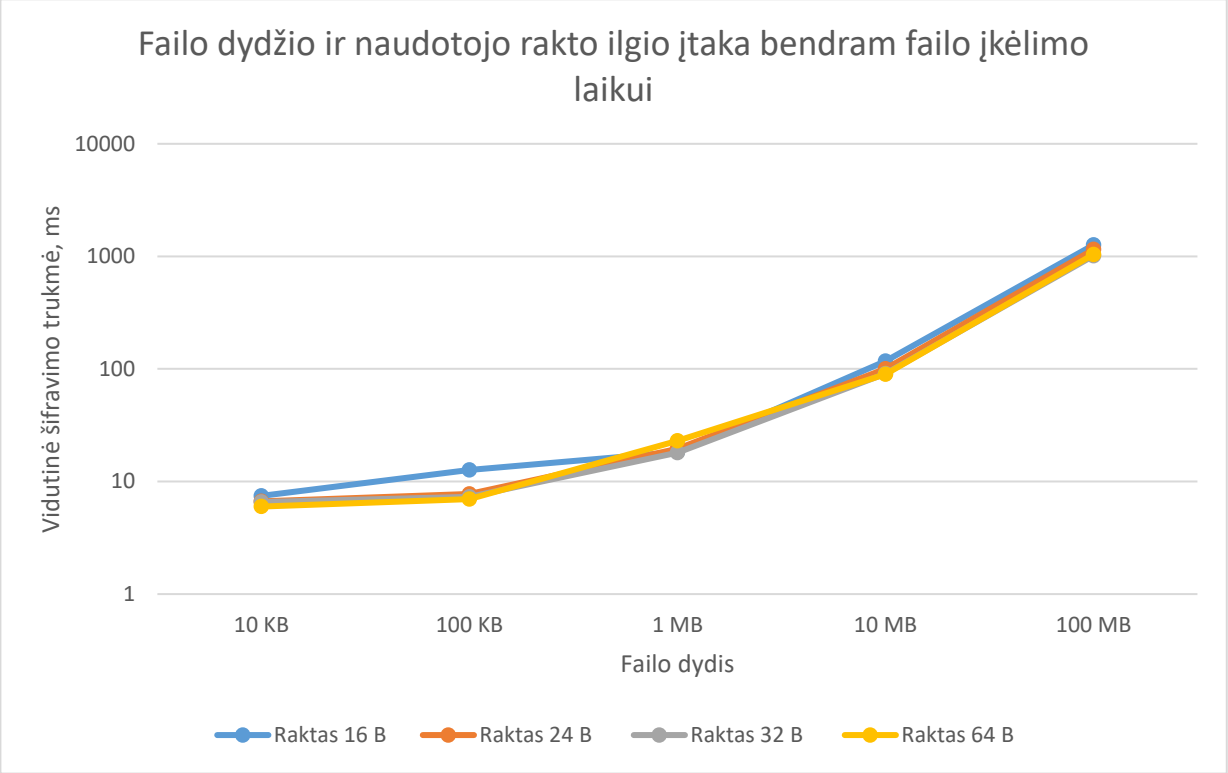
Eksperimentas	Trumpas aprašymas	Gautas rezultatas
E1	SHA-256 sutapimas po atkūrimo	Atkurtų failų SHA-256 kontrolinės sumos sutapo su originalių failų kontrolinėmis sumomis.
E2	Skirtingų formatų atkūrimas	TXT, PDF ir JPG / PNG failai buvo sėkmingai atkurti, jų turinys po atkūrimo sutapo su originalu.
E3	Skirtingų dydžių atkūrimas	Skirtingų dydžių failai buvo sėkmingai atkurti, o jų SHA-256 reikšmės sutapo su originaliomis.
E4	Netinkamas .key failas	Naudojant netinkamą .key failą, sistema neleido atkurti failo ir grąžino klaidą.
E5	Modifikuotas .key failas	Pakeitus S_{usr}, fingerprint arba mac reikšmę, sistema rakto failą atmetė kaip pažeistą.
E6	Šifruoto teksto modifikacija	Pakeitus vieną baitą šifruotame tekste, failo atkūrimas nepavyko, nes vientisumo patikra nebuvo sėkminga.
E7	file_tag modifikacija	Pakeitus file_tag reikšmę, failo iššifravimas nepavyko.
E8	file_nonce modifikacija	Pakeitus file_nonce reikšmę, failo iššifravimas nepavyko.
E9	EDEK sluoksnio modifikacija	Pakeitus edek, edek_tag arba edek_nonce, DEK atkūrimas nepavyko ir failas nebuvo atkurtas.
E10	Vieno failo saugus ištrynimas	Po saugaus ištrynimo pasirinktas failas tapo nebeatkuriamas.
E11	Negrįžtamumas palikus objektą	Net palikus užšifruotą objektą saugykloje, po rakto atkuriamumo panaikinimo failo atkurti nepavyko.
E12	Atrankinis ištrynimas	Ištrynus vieną failą, jis tapo nebeatkuriamas, tačiau kiti failai išliko atkuriami.
E13	Masinis saugus ištrynimas	Po masinio saugaus ištrynimo visi pažymėti failai tapo nebeatkuriami, o nepažymėti liko pasiekiami.

4.8. Eksperimentų rezultatai ir jų analizė

4.8.1. Failo dydžio ir naudotojo rakto ilgio įtaka bendram failo įkėlimo laikui

Siekiant įvertinti praktinį sprendimo veikimo laiką, pirmiausia analizuotas dviejų veiksnių – failo dydžio ir naudotojo rakto medžiagos ilgio – poveikis bendrai įkėlimo trukmei. Šiuo tikslu naudoti

skirtingos apimties failai ir keturi S_{usr} ilgio variantai, o kiekvieno įkėlimo metu automatiškai registruota *total_ms* reikšmė. Gauti rezultatai pateikti 24 paveiksle.



24 pav. Failo dydžio ir naudotojo rakto ilgio įtaka bendram failo įkėlimo laikui

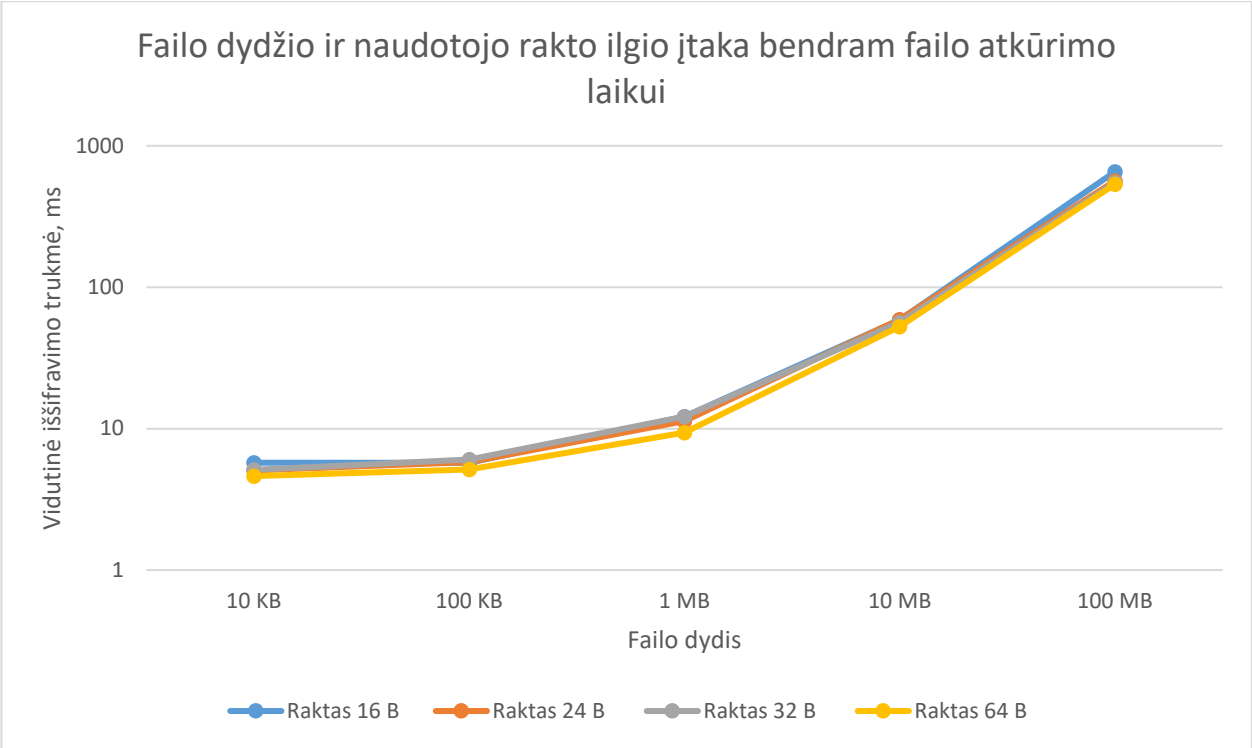
Kaip matyti 24 paveiksle, didėjant failo dydžiui bendras įkėlimo laikas nuosekliai ilgėja nepriklausomai nuo pasirinkto S_{usr} ilgio. Toks rezultatas yra tikėtinas, nes didesnio failo atveju ilgiau trunka tiek failo šifravimas, tiek jo įrašymas į objektinę saugyklą. Ši tendencija ypač ryški pereinant nuo 10 MB prie 100 MB failų, kur bendras laikas padidėja nuo kelių dešimčių ar šimtų milisekundžių iki daugiau nei vienos sekundės.

Taip pat galima pastebėti, kad skirtingų S_{usr} ilgių kreivės išlieka gana artimos viena kitai. Nors kai kuriuose taškuose matomi pavieniai nuokrypiai, jie nesudaro nuoseklios tendencijos, leidžiančios teigti, kad ilgesnis ar trumpesnis S_{usr} sistemingai spartina arba lėtina visą įkėlimo procesą. Tai reiškia, kad nors S_{usr} ilgis yra atskiras veiksnys, jo įtaka bendrai įkėlimo trukmei yra mažesnė už failo dydžio poveikį.

Šis rezultatas leidžia daryti išvadą, kad pagrindinis bendro įkėlimo laiko veiksnys yra failo dydis, o naudotojo rakto medžiagos ilgis neturi ryškaus praktinio poveikio viso įkėlimo scenarijaus trukmei. Pastebėti nedideli skirtumai tarp S_{usr} variantų labiau sietini su eksperimentinės aplinkos svyravimais negu su sisteminga rakto medžiagos ilgio įtaka.

4.8.2. Failo dydžio ir naudotojo rakto ilgio įtaka bendram failo atkūrimo laikui

Analogiškai buvo įvertinta failo dydžio ir S_{usr} ilgio įtaka bendrai failo atkūrimo trukmei. Atkūrimo metu vyksta naudotojo rakto failo tikrinimas, KEK išvedimas, DEK atkūrimas, užšifruoto objekto atsisiuntimas ir failo iššifravimas, todėl ši priklausomybė leidžia įvertinti viso atkūrimo proceso praktines sąnaudas. Rezultatai pateikti 25 paveiksle.



25 pav. Failo dydžio ir naudotojo rakto ilgio įtaka bendram failo atkūrimo laikui

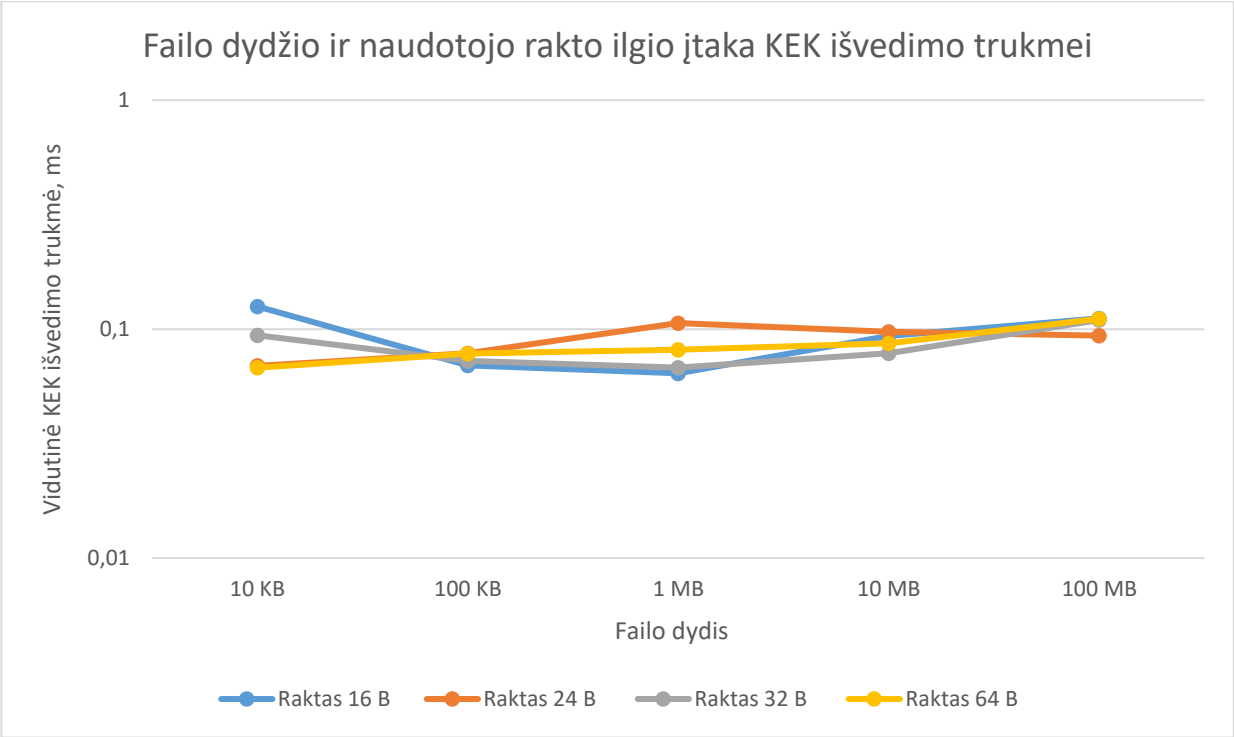
25 paveikslo duomenys rodo, kad failo atkūrimo laikas taip pat nuosekliai didėja didėjant failo dydžiui. Mažų failų atveju atkūrimo trukmė išlieka kelių milisekundžių ribose, o 100 MB failų atveju ji padidėja iki daugiau nei pusės sekundės. Kaip ir įkėlimo atveju, pagrindinis veiksnys čia yra ne rakto ilgis, o apdorojamų duomenų apimtis.

Kaip ir įkėlimo atveju, skirtingų S_{usr} ilgių kreivės išlieka gana artimos. Tai leidžia teigti, kad bendrą atkūrimo laiką daugiausia lemia duomenų gavimas ir failo iššifravimas, o ne naudotojo rakto medžiagos ilgio skirtumai. Nors kai kurie S_{usr} variantai tam tikruose taškuose atrodo šiek tiek spartesni, bendra tendencija išlieka tokia pati ir neleidžia išskirti vieno rakto ilgio kaip nuosekliai pranašesnio.

Šis rezultatas taip pat leidžia vertinti abiejų veiksnių sąveiką. Kadangi skirtingų S_{usr} ilgių linijos išlaiko panašią formą ir didėjant failo dydžiui nesiskiria radikaliai, galima teigti, kad rakto medžiagos ilgis neturi reikšmingos sąveikos su failo dydžio poveikiu bendram atkūrimo laikui. Kitaip tariant, didesnio failo apdorojimas išlieka lėtesnis visais S_{usr} ilgio atvejais, tačiau pats rakto medžiagos ilgis šios tendencijos iš esmės nekeičia.

4.8.3. Failo dydžio ir naudotojo rakto ilgio įtaka KEK išvedimo trukmei

Atskirai tirta, kaip failo dydis ir S_{usr} ilgis veikia KEK išvedimo trukmę. Šis eksperimentas svarbus tuo, kad siūlomame metode S_{usr} nėra tiesioginis AES raktas, bet naudojamas kaip įvesties medžiaga HKDF funkcijai. Todėl čia nagrinėjamas dviejų veiksnių poveikis vienam iš svarbiausių raktų valdymo etapų. Gauti rezultatai pateikti 26 paveiksle.



26 pav. Failo dydžio ir naudotojo rakto ilgio įtaka KEK išvedimo trukmei

Kaip matyti 26 paveiksle, KEK išvedimo trukmė yra labai maža visose tirtose kombinacijose. Reikšmės išlieka šimtųjų ar dešimtųjų milisekundės dalių ribose ir yra nepalyginamai mažesnės už bendrą įkėlimo ar atkūrimo laiką. Tai rodo, kad pats raktų išvedimo etapas nesudaro reikšmingos našumo naštos.

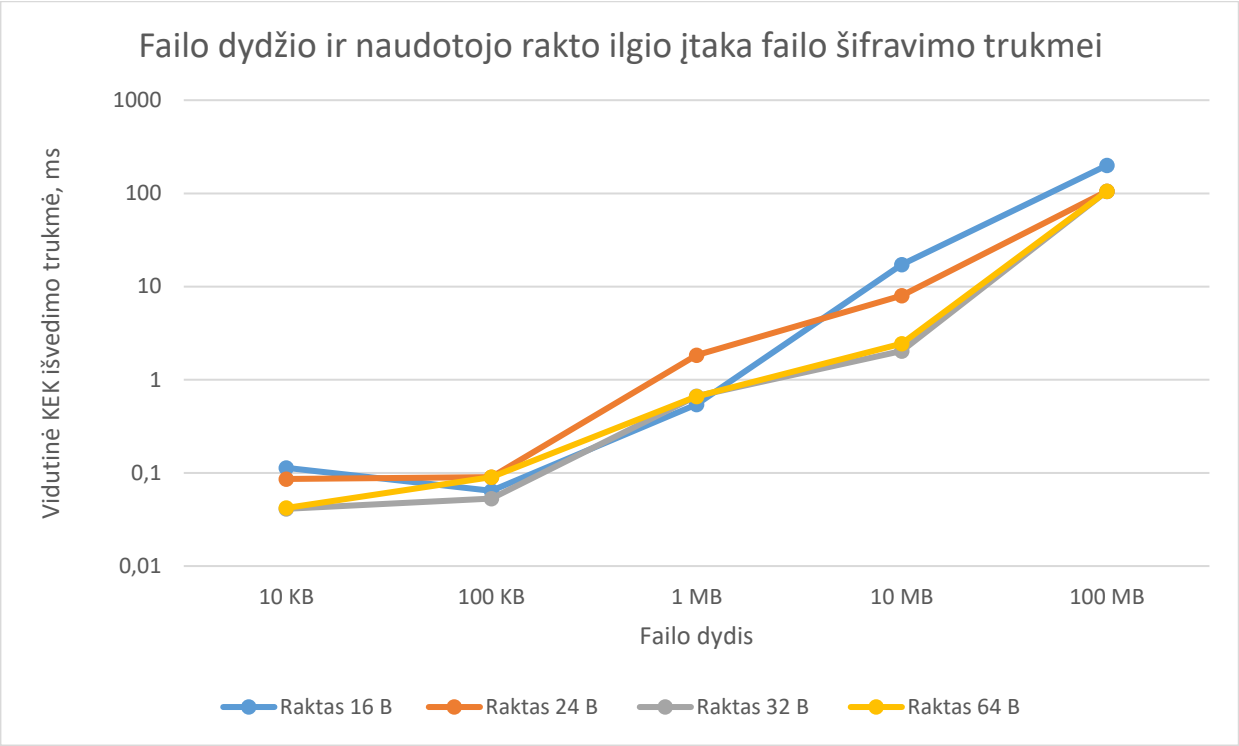
Taip pat matyti, kad nei failo dydžio, nei S_{usr} ilgio poveikis šiai metrikai nėra ryškus. Nors tarp atskirų kombinacijų galima stebėti nedidelius skirtumus, jie nesudaro nuoseklios tendencijos. Tai logiška, nes KEK išvedimas priklauso nuo raktų medžiagos apdorojimo, o ne nuo paties failo turinio apimties.

Ši išvada yra svarbi metodiniu požiūriu, nes ji rodo, kad papildomas raktų valdymo sluoksnis nesukuria reikšmingos našumo naštos. Kitaip tariant, siūlomo metodo praktinį veikimo laiką daugiausia lemia failo turinio apdorojimas, o ne pati KEK išvedimo logika. Be to, skirtingi S_{usr} ilgio variantai šioje realizacijoje neparodė apčiuopiamos papildomos praktinės naudos našumo požiūriu.

4.8.4. Failo dydžio ir naudotojo rakto ilgio įtaka failo šifravimo ir iššifravimo trukmėms

Siekiant detaliau suprasti, kurie etapai labiausiai lemia bendrą sistemos veikimo laiką, papildomai išanalizuotos pačios failo šifravimo ir iššifravimo trukmės. Tokia analizė leidžia atskirti bendrą operacijos laiką nuo vidinių kriptografinių veiksmų trukmių ir tiksliau įvertinti, kaip elgiasi metodo kriptografinis branduolys.

Pirmiausia įvertinta failo dydžio ir S_{usr} ilgio įtaka failo šifravimo trukmei, naudojant *file_encrypt_ms* metriką. Rezultatai pateikti 27 paveiksle.

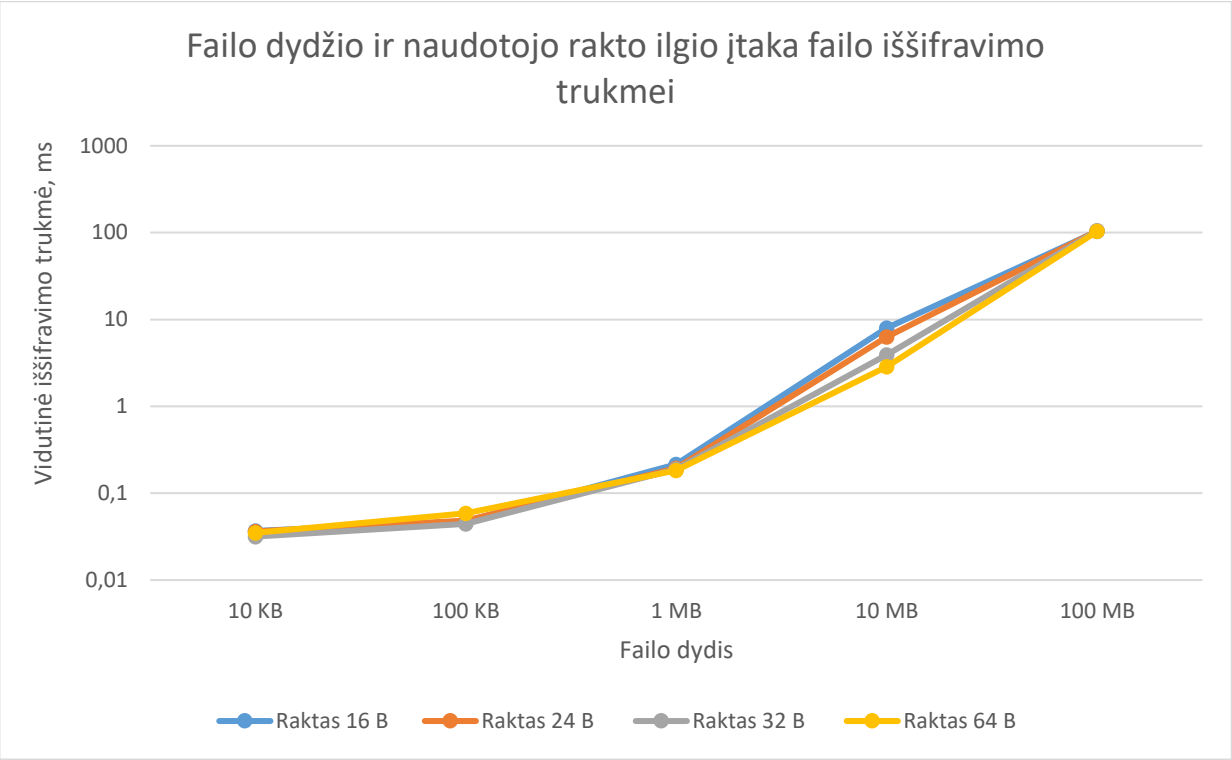


27 pav. Failo dydžio ir naudotojo rakto ilgio įtaka failo šifravimo trukmei

27 paveiksle matyti, kad failo šifravimo trukmė nuosekliai didėja kartu su apdorojamo failo dydžiu. Ši tendencija ypač aiški 10 MB ir 100 MB failų atvejais, kur šifravimo laiko skirtumai tampa ryškūs. Tai patvirtina, kad kriptografinis branduolys elgiasi prognozuojamai, o jo veikimo laikas tiesiogiai susijęs su apdorojamų duomenų apimtimi.

Skirtingų Susr ilgių kreivės daugeliu atvejų išlieka gana artimos, nors kai kuriuose taškuose matyti pavieniai nuokrypiai. Pavyzdžiui, 10 MB ir 100 MB failų atveju 16 B S_{usr} variantas kai kuriuose bandymuose pasirodė lėtesnis už kitus. Vis dėlto tokie skirtumai nesudaro nuoseklios tendencijos visuose failų dydžiuose, todėl labiau sietini su eksperimentinės aplinkos svyravimais negu su sisteminga rakto medžiagos ilgio įtaka pačiam šifravimo procesui.

Analogiškai įvertinta failo dydžio ir S_{usr} ilgio įtaka failo iššifravimo trukmei, naudojant *file_decrypt_ms* metriką. Rezultatai pateikti 28 paveiksle.



28 pav. Failo dydžio ir naudotojo rakto ilgio įtaka failo iššifravimo trukmei

28 paveikslo rezultatai rodo tokią pačią bendrą tendenciją ir failo iššifravimo etape. Didėjant failo dydžiui, iššifravimo trukmė nuosekliai ilgėja, ypač pereinant prie 10 MB ir 100 MB failų. Skirtingų S_{usr} ilgių kreivės vėlgi išlieka gana artimos. Tai rodo, kad tiek šifravimo, tiek iššifravimo trukmės daugiausia priklauso nuo failo dydžio, o naudotojo rakto medžiagos ilgis neturi reikšmingos įtakos šiems etapams.

Vertinant abu grafikus kartu galima teigti, kad kriptografinis branduolys veikia stabiliai ir prognozuojamai. Skirtingų S_{usr} ilgių kreivių artimumas leidžia daryti išvadą, kad rakto medžiagos ilgio poveikis failo turinio šifravimo ir iššifravimo etapams yra minimalus, o pagrindinis šių etapų trukmių veiksnys yra failo dydis.

4.8.5. Funkcinių ir saugumo eksperimentų rezultatų analizė

Be kiekybinių laiko matavimų, šiame darbe taip pat įvertinti funkciniai ir saugumo eksperimentai. Jų rezultatai parodė, kad prototipas korektiškai realizuoja siūlomą metodo logiką. Įkėlimo ir atkūrimo scenarijuose failai iki saugaus ištrynimo momento galėjo būti atkuriami identiškai originalui, o jų SHA-256 kontrolinės sumos sutapo. Tai patvirtina bazinio failo įkėlimo, šifravimo ir atkūrimo ciklo korektiškumą.

Eksperimentai su netinkamu ir modifikuotu naudotojo rakto failu parodė, kad sistema neleidžia atkurti failo naudojant klaidingą rakto failą su .key plėtiniu. Tai reiškia, kad rakto failo vientisumo kontrolė realizuota korektiškai, o naudojami identifikatoriaus ir autentiškumo kodo mechanizmai leidžia aptikti neatitinkančius ar pažeistus rakto artefaktus.

Taip pat nustatyta, kad AES-256-GCM vientisumo apsauga taikoma tiek failo šifruotam tekstui, tiek užšifruotam DEK. Pakeitus *file_tag, file_nonce, edek, edek_tag* arba *edek_nonce*, failo atkūrimas

nebeįvyksta. Tai rodo, kad sistemoje neleidžiama nepastebimai modifikuoti nei užšifruoto failo, nei jo raktų apsaugos sluoksnio.

Svarbiausi šio darbo požiūriu yra eksperimentai, susiję su kriptografiniu naikinimu. Jų rezultatai parodė, kad po rakto atkuriamumo panaikinimo failas tampa nebeatkuriamas. Dar svarbiau tai, kad šis rezultatas išlieka net tuo atveju, jei užšifruotas objektas dar kurį laiką egzistuoja saugykloje. Be to, nustatyta, kad vieno failo saugus ištrynimas nepaveikia kitų failų, net jei jie susieti su tuo pačiu naudotojo rakto profiliu. Tai patvirtina, kad prototipe realizuotas atrankinis kriptografinis naikinimas.

14 lentelėje pateikta funkcinių ir saugumo eksperimentų suvestinė patvirtina, kad sukurta sistema ne tik veikia korektiškai, bet ir užtikrina svarbiausias siūlomo metodo saugumo savybes: duomenų vientisumą, rakto failo vientisumo kontrolę, atrankinį ištrynimą ir negrįžtamumą po rakto atkuriamumo panaikinimo.

4.8.6. Bendras eksperimentinių rezultatų apibendrinimas

Apibendrinant kiekybinius ir kokybinius rezultatus galima teigti, kad pagrindiniai šiame darbe nagrinėti veiksniai – failo dydis ir naudotojo rakto medžiagos ilgis S_{usr} – turi nevienodą poveikį metodo veikimo laikui. Pirmasis veiksnys, tai yra failo dydis, turi aiškiai stipresnį poveikį bendroms įkėlimo, atkūrimo, šifravimo ir iššifravimo trukmėms. Antrasis veiksnys, tai yra S_{usr} ilgis, daro gerokai mažesnę įtaką ir daugiausia atsiskleidžia KEK išvedimo etape, tačiau ir ten jo poveikis išlieka ribotas.

Taip pat galima teigti, kad šių dviejų veiksnių sąveika nėra ryški. Tai rodo panašios formos kreivės skirtingiems S_{usr} ilgiams. Kitaip tariant, failo dydžio įtaka sistemos veikimo laikui išlieka pagrindinė visais naudotojo rakto medžiagos ilgio atvejais. Ši išvada leidžia teigti, kad siūlomas metodas išlieka praktiškai taikomas ir tuomet, kai naudojami skirtingo ilgio naudotojo raktų profiliai.

Svarbu pabrėžti, kad skirtingi S_{usr} ilgio variantai nepakeitė sistemos funkcionalumo, vientisumo apsaugos ar bendros saugumo logikos. Šioje realizacijoje failų apsaugos pagrindą visais atvejais išlaikė ta pati AES-256-GCM ir HKDF pagrindu realizuota schema. Todėl galima teigti, kad S_{usr} ilgio keitimas neparodė reikšmingos praktinės įtakos nei bendram našumui, nei metodo saugumo savybėms, vertinant pagal atliktų eksperimentų rezultatus.

Kartu funkciniai ir saugumo eksperimentai parodė, kad prototipas ne tik veikia korektiškai, bet ir įgyvendina pagrindinę darbo hipotezę: panaikinus raktų grandinės atkuriamumą, failo turinys tampa praktiškai neatkuriamas, net jei užšifruotas objektas dar egzistuoja saugykloje. Tokiu būdu eksperimentiniai rezultatai patvirtina, kad siūlomas metodas yra ne tik teoriškai pagrįstas, bet ir praktiškai realizuojamas.

4.9. Skyriaus apibendrinimas

Šiame skyriuje pateiktas siūlomo metodo kokybinis vertinimas, aprašyta prototipo realizacijos aplinka, eksperimentų strategija, matuojami rodikliai, testiniai duomenys ir eksperimentiniai scenarijai. Taip pat apibrėžta, kokiu būdu surinkti eksperimentiniai duomenys yra apdorojami ir pateikiami grafiškai.

Kokybinis vertinimas parodė, kad siūlomas metodas turi aiškių pranašumų prieš loginį ištrynimą ir yra geriau pritaikytas paskirstytai infrastruktūrai negu perrašymu pagrįsti metodai. Eksperimentinė

dalis orientuota į tai, kad būtų praktiškai patvirtinta pagrindinė darbo hipotezė: panaikinus raktų grandinės atkuriamumą, duomenys tampa nebeatkuriami net ir tuo atveju, jei užšifruotas objektas tam tikrą laiką išlieka saugykloje.

Atliktų eksperimentų struktūra leidžia nuosekliai įvertinti tiek metodo korektiškumą, tiek jo praktinį veikimo laiką. Surinkti duomenys sudaro pagrindą pagrįstai interpretuoti, kaip failo dydis, naudotojo rakto medžiagos ilgis ir atskiri kriptografiniai etapai veikia bendrą sistemos elgseną. Dėl to šis skyrius sudaro loginį pagrindą galutinėms darbo išvadoms.

Išvados

1. Išanalizavus saugaus duomenų ištrynimo problemą paskirstytoje, debesijos ir virtualiojoje infrastruktūroje nustatyta, kad tradicinis failo pašalinimas ne visada reiškia faktinį duomenų sunaikinimą. Duomenys gali išlikti atsarginėse kopijose, versijose, replikose ar kituose infrastruktūros sluoksniuose, o naudotojas dažnai neturi tiesioginės fizinės laikmenos kontrolės. Todėl tokioje aplinkoje saugus duomenų ištrynimas turi būti grindžiamas ne tik objekto pašalinimu, bet ir praktiniu duomenų nebeatkuriamumu.
2. Ištyrus pagrindinius duomenų ištrynimo metodus nustatyta, kad loginis ištrynimas yra paprastas ir lengvai pritaikomas, tačiau neužtikrina pakankamo negrįžtamumo, o perrašymu pagrįstas metodas paskirstytoje infrastruktūroje yra ribotai pritaikomas dėl fizinės laikmenos kontrolės nebuvimo. Kriptografinis duomenų naikinimas įvertintas kaip tinkamiausias metodas, nes panaikinus raktų atkuriamumą duomenų turinys tampa nebeiššifruojamas ir praktiškai nebeatkuriamas net tada, kai užšifruotas objektas dar išlieka saugykloje.
3. Išanalizavus kriptografinių raktų valdymo principus nustatyta, kad saugiam duomenų ištrynimui svarbiausi yra DEK ir KEK hierarchija, voko šifravimo principas, rakto gyvavimo ciklo valdymas ir kontroliuojamas rakto atkuriamumo panaikinimas. Siūlomame metode failo turinys šifruojamas atskiru duomenų šifravimo raktu DEK, o pats DEK apsaugomas raktų šifravimo raktu KEK, kuris nėra saugomas kaip vientisas ilgalaikis objektas, bet išvedamas iš naudotojo rakto dalies, serverio rakto dalies ir druskos reikšmės.
4. Palyginus mokslinėje literatūroje siūlomus ir praktikoje taikomus saugaus duomenų ištrynimo sprendimus nustatyta, kad raktų atskyrimu, smulkiai valdomu kriptografiniu naikinimu ir verifikuojamu ištrynimu pagrįsti sprendimai geriausiai atitinka paskirstytos infrastruktūros ribojimus. Taip pat nustatyta, kad komercinėse debesijos platformose saugus duomenų ištrynimas priklauso ne tik nuo objektų šalinimo, bet ir nuo versijų, atsarginių kopijų, saugojimo politikų bei raktų valdymo mechanizmų. Todėl praktinis saugaus ištrynimo patikimumas tiesiogiai priklauso nuo pasirinkto raktų valdymo modelio.
5. Suprojektavus ir praktiškai įvertinus kriptografinių raktų valdymo metodu pagrįstą prototipą nustatyta, kad pasiūlytas sprendimas yra praktiškai realizuojamas ir tinkamas saugiam duomenų ištrynimui virtualiojoje infrastruktūroje. Eksperimentai parodė, kad iki saugaus ištrynimo failai atkuriami korektiškai, jų SHA-256 kontrolinės sumos sutampa, sistema aptinka netinkamą arba modifikuotą rakto failą, o panaikinus raktų grandinės atkuriamumą failas tampa nebeatkuriamas. Taip pat nustatyta, kad vieno failo saugus ištrynimas nepaveikia kitų failų, todėl metodas leidžia įgyvendinti atrankinį kriptografinį duomenų naikinimą.

Informacijos šaltinių sąrašas

[1] MELL, P.; GRANCE, T. The NIST Definition of Cloud Computing. NIST [interaktyvus]. 2011, p. 2. [žiūrėta 2026-04-01]. Prieiga per: https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-145.pdf

[2] JANSEN, W.; GRANCE, T. Guidelines on Security and Privacy in Public Cloud Computing. NIST [interaktyvus]. 2011. [žiūrėta 2026-04-01]. Prieiga per: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-144.pdf

[3] ENISA. ENISA Threat Landscape 2024. ENISA [interaktyvus]. 2024. [žiūrėta 2026-04-01]. Prieiga per: https://www.enisa.europa.eu/publications/enisa-threat-landscape-2024

[4] CHANDRAMOULI, R. A Data Protection Approach for Cloud-Native Applications. NIST [interaktyvus]. 2024, p. 1. [žiūrėta 2026-04-01]. Prieiga per: https://nvlpubs.nist.gov/nistpubs/ir/2024/NIST.IR.8505.pdf

[5] RAMOKAPANE, K. M.; RASHID, A.; SUCH, J. M. Assured Deletion in the Cloud: Requirements, Challenges and Future Directions. ACM Digital Library [interaktyvus]. 2016 p. 97, 98, 101, 102. [žiūrėta 2026-04-01]. Prieiga per: https://dl.acm.org/doi/10.1145/2996429.2996434

[6] MICROSOFT. Soft delete for blobs. Microsoft Learn [interaktyvus]. n.d. [žiūrėta 2026-04-01]. Prieiga per: https://learn.microsoft.com/en-us/azure/storage/blobs/soft-delete-blob-overview

[7] MICROSOFT. Blob versioning. Microsoft Learn [interaktyvus]. n.d. [žiūrėta 2026-04-01]. Prieiga per: https://learn.microsoft.com/en-us/azure/storage/blobs/versioning-overview

[8] AMAZON WEB SERVICES. Working with delete markers. Amazon S3 User Guide [interaktyvus]. n.d. [žiūrėta 2026-04-01]. Prieiga per: https://docs.aws.amazon.com/AmazonS3/latest/userguide/DeleteMarker.html

[9] AMAZON WEB SERVICES. Locking objects with Object Lock. Amazon S3 User Guide [interaktyvus]. n.d. [žiūrėta 2026-04-01]. Prieiga per: https://docs.aws.amazon.com/AmazonS3/latest/userguide/object-lock.html

[10] NIST. Guidelines for Media Sanitization. NIST [interaktyvus]. 2025, p. 1, 5, 11, 32.. [žiūrėta 2026-04-01]. Prieiga per: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-88r2.pdf

[11] WEI, M.; GRUPP, L. M.; SPADA, F. E.; SWANSON, S. Reliably Erasing Data From Flash-Based Solid State Drives. USENIX [interaktyvus]. 2011, p. 1. [žiūrėta 2026-04-01]. Prieiga per: https://www.usenix.org/legacy/event/fast11/tech/full_papers/Wei.pdf

[12] RATLIFF, Z.; GOH, W.; WIELAND, A.; MICKENS, J.; WILLIAMS, R. R. Holepunch: Fast, Secure File Deletion with Crash Consistency. IEEE Xplore [interaktyvus]. 2024. [žiūrėta 2026-04-01]. Prieiga per: https://ieeexplore.ieee.org/document/10646634

[13] BARKER, E. Recommendation for Key Management: Part 1 – General. NIST [interaktyvus]. 2020, p. 109. [žiūrėta 2026-04-01]. Prieiga per: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf

[14] AMAZON WEB SERVICES. Data keys. AWS Key Management Service Documentation [interaktyvus]. n.d. [žiūrėta 2026-04-01]. Prieiga per: https://docs.aws.amazon.com/kms/latest/developerguide/concepts.html#data-keys

[15] AMAZON WEB SERVICES. AWS Encryption SDK concepts. AWS Documentation [interaktyvus]. n.d. [žiūrėta 2026-04-01]. Prieiga per: https://docs.aws.amazon.com/encryption-sdk/latest/developer-guide/concepts.html

[16] MICROSOFT. Key management in Azure Managed HSM. Microsoft Learn [interaktyvus]. n.d. [žiūrėta 2026-04-01]. Prieiga per: https://learn.microsoft.com/en-us/azure/key-vault/managed-hsm/key-management

[17] EUROPEAN DATA PROTECTION BOARD. Coordinated Enforcement Action 2025: Implementation of the Right to Erasure by Controllers. EDPB [interaktyvus]. 2025, p. 5, 18. [žiūrėta 2026-04-01]. Prieiga per: https://www.edpb.europa.eu/system/files/2026-02/edpb_cef-report_2025_right-to-erasure_en.pdf

[18] EUROPEAN DATA PROTECTION BOARD. Annex to the Coordinated Enforcement Action on the Right to Erasure. EDPB [interaktyvus]. 2025, p. 235–236. [žiūrėta 2026-04-01]. Prieiga per: https://www.edpb.europa.eu/system/files/2026-02/edpb_cef-report_2025_right-to-erasure_annex_en.pdf

[19] NIST. Recommendation for Password-Based Key Derivation: Part 1: Storage Applications. NIST [interaktyvus]. 2010, p. 1, 5. [žiūrėta 2026-04-01]. Prieiga per: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-132.pdf

[20] ISO. ISO/IEC 27001:2022 Information security management systems. ISO [interaktyvus]. 2022. [žiūrėta 2026-04-01]. Prieiga per: https://www.iso.org/standard/27001

[21] ISO. ISO/IEC 27002:2022 Information security controls. ISO [interaktyvus]. 2022. [žiūrėta 2026-04-01]. Prieiga per: https://www.iso.org/standard/27002

[22] TANG, Y.; LEE, P. P. C.; LUI, J. C. S.; PERLMAN, R. FADE: Secure Overlay Cloud Storage with File Assured Deletion. The Chinese University of Hong Kong [interaktyvus]. 2010. [žiūrėta 2026-04-01]. Prieiga per: https://www.cse.cuhk.edu.hk/~pclee/www/pubs/securecomm10.pdf

[23] TIAN, J.; WANG, Z. Fine-grained Assured Data Deletion Scheme Based on Attribute Association. ScienceDirect [interaktyvus]. 2020. [žiūrėta 2026-04-01]. Prieiga per: https://www.sciencedirect.com/science/article/abs/pii/S016740482030211X

[24] YUE, Z.; LIU, J.; ZHANG, C.; FANG, Y.; CHENG, X. ATDD: Fine-Grained Assured Time-Sensitive Data Deletion in Cloud Storage. arXiv.org [interaktyvus]. 2022. [žiūrėta 2026-04-01]. Prieiga per: https://arxiv.org/pdf/2205.02117

[25] LI, X.; NI, J. Accelerating Secure and Verifiable Data Deletion in Cloud Storage via SGX and Blockchain. arXiv.org [interaktyvus]. 2023. [žiūrėta 2026-04-01]. Prieiga per: https://arxiv.org/pdf/2307.04316

[26] XU, Z.; CHEN, X.; LAN, X.; TANG, R.; JIANG, S.; SHEN, C. Empowering Data Owners: An Efficient and Verifiable Scheme for Secure Data Deletion. ScienceDirect [interaktyvus]. 2024. [žiūrėta 2026-04-01]. Prieiga per: https://www.sciencedirect.com/science/article/abs/pii/S0167404824002839

[27] MICROSOFT. Overview of immutable storage for blob data. Microsoft Learn [interaktyvus]. n.d. [žiūrėta 2026-04-01]. Prieiga per: https://learn.microsoft.com/en-us/azure/storage/blobs/immutable-storage-overview

[28] AMAZON WEB SERVICES. Using server-side encryption with AWS KMS keys (SSE-KMS). Amazon S3 User Guide [interaktyvus]. n.d. [žiūrėta 2026-04-01]. Prieiga per: https://docs.aws.amazon.com/AmazonS3/latest/userguide/UsingKMSEncryption.html

[29] NIST. Recommendation for Cryptographic Key Generation. NIST [interaktyvus]. 2020, p. 21. [žiūrėta 2026-04-01]. Prieiga per: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-133r2.pdf

[30] DODIS, Y.; OSTROVSKY, R.; REYZIN, L.; SMITH, A. Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. Boston University [interaktyvus]. n.d. [žiūrėta 2026-04-01]. Prieiga per: https://www.cs.bu.edu/~reyzin/papers/fuzzy.pdf

[31] TRAN, T. T.; PHAN, D. H.; NGUYEN, N. T. Multi-Factor Fuzzy Extractor. arXiv.org [interaktyvus]. 2024, p. 1. [žiūrėta 2026-04-01]. Prieiga per: https://arxiv.org/pdf/2405.11456

[32] NANDAKUMAR, K.; NAGAR, A.; JAIN, A. K. Hardening Fingerprint Fuzzy Vault Using Password. Michigan State University Biometrics Research Group [interaktyvus]. 2007, p. 1. [žiūrėta 2026-04-01]. Prieiga per: https://biometrics.cse.msu.edu/Publications/SecureBiometrics/NandakumarNagarJain_FpFuzzyVaultHardening_ICB2007.pdf

[33] NAIR, V.; SONG, D. Multi-Factor Key Derivation Function (MFKDF) for Fast, Flexible, Secure, and Practical Key Management. USENIX [interaktyvus]. 2023. [žiūrėta 2026-04-01]. Prieiga per: https://www.usenix.org/system/files/usenixsecurity23-nair-mfkdf.pdf

[34] ŠEDĚNKA, J.; BALAGANI, K. S.; PHOHA, V.; GASTI, P. Privacy-Preserving Population-Enhanced Biometric Key Generation from Free-Text Keystroke Dynamics. arXiv.org [interaktyvus]. 2014. [žiūrėta 2026-04-01]. Prieiga per: https://arxiv.org/pdf/1407.4179

[35] SURESH, P.; PARVATHI, R. M. V.; KASHWAN, K. R. Two-factor-based RSA Key Generation from Fingerprint Biometrics and Password for Secure Communication. ResearchGate [interaktyvus]. 2022. [žiūrėta 2026-04-01]. Prieiga per: https://www.researchgate.net/publication/358709755_Two-factor-based_RSA_key_generation_from_fingerprint_biometrics_and_password_for_secure_communication

[36] BENTAJER, A.; EZBAKHE, A.; MERRAS, M.; TAIRI, H.; HEDABOU, M. Secure Cloud Key Management Based on Robust Secret Sharing. ResearchGate [interaktyvus]. 2021. [žiūrėta 2026-04-01]. Prieiga per: https://www.researchgate.net/profile/Ahmed-Bentajer/publication/351660744_Secure_Cloud_Key_Management_based_on_Robust_Secret_Sharing/links/60d49f05a6fdcc75a2505412/Secure-Cloud-Key-Management-based-on-Robust-Secret-Sharing.pdf

[37] SHAMSOSHOARA, A.; KORENDA, A.; AFGHAH, F.; ZEADALLY, S. A Survey on Physical Unclonable Function (PUF)-Based Security Solutions for Internet of Things. arXiv.org [interaktyvus]. 2019. [žiūrėta 2026-04-01]. Prieiga per: https://arxiv.org/pdf/1907.12525

[38] JARECKI, S.; KIAYIAS, A.; KRAWCZYK, H.; XU, J. Password-Protected Secret Sharing. IACR ePrint [interaktyvus]. 2010. [žiūrėta 2026-04-01]. Prieiga per: https://eprint.iacr.org/2010/561.pdf

[39] MIANDOAB, D. G.; NASER, M. E.; KHARAZI, M. R. H. K. Statistical Analysis of ReRAM-PUF Based Keyless Encryption Protocol Against Frequency Analysis Attack. arXiv.org [interaktyvus]. 2021. [žiūrėta 2026-04-01]. Prieiga per: https://arxiv.org/pdf/2109.11075

[40] MICROSOFT. BYOK specification for Azure Key Vault. Microsoft Learn [interaktyvus]. n.d. [žiūrėta 2026-04-01]. Prieiga per: https://learn.microsoft.com/en-us/azure/key-vault/keys/byok-specification