KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
MULTIMEDIJOS INŽINERIJOS KATEDRA

Karolis Kaulinis
Vygantas Česaitis

Duomenų pernešamose laikmenose apsaugos metodų tyrimas

Magistro darbas

Darbo vadovas
Doc. dr. A. Ostreika

Kaunas, 2012

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
MULTIMEDIJOS INŽINERIJOS KATEDRA

Karolis Kaulinis
Vygantas Česaitis

Duomenų pernešamose laikmenose apsaugos metodų tyrimas

Magistro darbas

Recenzentas

Darbo vadovas
Doc. dr. A. Ostreika
2012-05-28

Atliko
IMF-0/4 gr. Studentai
Vygantas Česaitis
Karolis Kaulinis
2012-05-28

Kaunas, 2012

Turinys

Santrauka

Atsižvelgiant į tendencijas, kurios šiuo metu tvyro pernešamųjų laikmenų segmente, šiame darbe buvo išnagrinėti jau egzistuojantys dabar populiariausių tokio tipo laikmenų, USB atmintinių, apsaugos metodai, kurie plačiai taikomi tiek verslo įmonėse, tiek privačių asmenų, kai norima apsaugoti duomenis, kurie po to būtų įkelti ir saugomi tokio tipo atmintinėje.

Suformuluotos techninės ir programinės sąlygos USB atmintinėje saugomų duomenų saugumui užtikrinti. Pateikti projektiniai sprendimai tokios sistemos aparatūrinei įrangai ir debesų kompiuterija pagrįstai informacinei sistemai. Suprojektuota sistema leidžia naudojant autorizuotą atmintinę pasiekti norimus duomenis, kurie talpinami serveryje.

Summary

Looking at the trends that are dominating today in portable medium segment in this paper we've analyzed existing data protection programs and solutions to secure data in the most popular at this moment USB Flash drives that are used widely by today's business and private users that want to protect their data in this kind of flash memory drives.

The technical and software conditions in order to secure data in the flash drives have been framed. Design solutions have been set to combine data protection solutions with cloud based technology. The control (test) sample imitates the flash drive client side functionality; created draft illustrates accessible resources in the system and possible functions which might be carried out by users or administrators.

Terminų ir santrumpų žodynėlis

USB (angl. *Universal Serial Bus*) – tai universali jungtis, kuri naudojama daugumoje šiuolaikinių kompiuterių ir buitinių įtaisų. Per šią jungtį galima prijungti įvairius išorinius įrenginius.

Šifravimas – tai toks procesas, kurio metu paprastas tekstas paverčiamas į nesuprantamą, vadinamą šifru (angl. *cipher*) arba kriptograma (angl. *cryptogram*).

UML (angl. *Unified Modeling Language*) – tai modeliavimui ir specifikacijoms kurti skirta kalba, kuria galima specifikuoti, konstruoti ir kitaip atvaizduoti objektiškai orientuotų programų dokumentus.

XOR (angl. eXclusive OR) – tai sumoms moduliu du operacija.

Flash – tai nekintanti kompiuterio atmintinė, kurioje dažniausiai yra integruota USB jungtis.

FAT (angl. *File Allocation Table*) – grupė failų sistemų, kurios naudojamos Windows operacinėse sistemose.

CD (angl. *Compact Disk*) – optinis diskas skirtas saugoti informacijai įrašytai skaitmenine forma.

DVD (angl. *Digital Versatile Disc*) – tai didesnės talpos optinis diskas skirtas skaitmeninei informacijai saugoti. CD yra DVD disko pirmtakas.

1. Įvadas

Šiais laikais, kai informacijos kiekiai didžiuliai, vyksta vis didesni informacijos apsikeitimai tarp žmonių. Šie apsikeitimai bei migracijos vykdomi įvairiais būdais, informacija gali būti perduodama rašytiniuose šaltiniuose, t.y. popieriuje, gali būti perduodama žodžiu, siunčiama internetu ar įrašoma į tam tikras laikmenas, kurias prijungus prie kompiuterio, ar kito tam skirto įrenginio, būtų galima tą informaciją perskaityti.

Didėjant informacijos mainams iš kompiuterio į kompiuterį, vis dažniau naudojama įvairiausios išorinės duomenų laikmenos (CD, DVD, USB atmintinės). Tačiau kuo paprastesnė ir visiems lengviau prieinama laikmena, tuo paprasčiau informacija ar duomenys gali būti prarasti, pavogti ar sunaikinti.

Šiame darbe yra tiriama jau esamos duomenų apsaugos priemonės ir metodai. Analizuojamas veikimas, privalumai ir trūkumai jau esančios programinės ir aparatūrinės įrangos skirtos duomenų apsaugai.

2. Duomenų pernešamose laikmenose apsaugos metodų analizė

2.1. Analizės tikslas

Tyrimo tikslas - išanalizuoti jau esamus išorinių nešiojamų USB atmintinių apsaugos metodus, nustatyti jų pliusus ir minusus, išsiaiškinti jų bendrus bruožus lyginant ją tarpusavyje. Taip pat šių metodų galimybes bei patikimumą.

Remiantis atlikta analize, bus pasiūlytas savas metodas bei sprendimas duomenų nešiojamuose atmintinėse apsaugai. Taip pat projektuojami būsimos sistemos komponentai.

2.2. Tyrimo sritis, objektas ir problema

Tyrimo objektas - metodai skirti apsaugoti duomenis USB atmintinėse nuo praradimo. Kaip žinome, informaciją ar pateiktus duomenis ant popieriaus lapo galima užšifruoti pagal tam tikras aprašytas taisykles, o norint sunaikinti - užtenka įdėti popieriaus lapą į naikintuvą. Informacija bei duomenys siunčiami internetu, taip pat gali būti šifruojami, koduojami pagal tam tikrus apsaugos algoritmus, todėl tretiems asmenims ji nėra pasiekiama. Problema iškyla kai norime apsaugoti informaciją, esančia išorinėse laikmenose, įrašytą skaitmeniniu pavidalu.

Ne taip seniai viena populiariausių išorinių laikmenų buvo kompaktinis diskas (CD), vėliau, vis didėjant informacijos kiekiams išpopuliarėjo DVD laikmenos.

Tačiau šio tipo laikmenos turėjo kelis neigiamus bruožus:

- Reikia saugoti nuo išorinio poveikio (tiesioginių saulės spindulių, subraižymo, sulaužymo);
- Duomenys į laikmeną gali būti įrašyti tik vieną kartą (vėliau atsirado ir daugkartinio įrašymo diskai, tačiau jie nėra patikimi);
- Reikalingas specialus įrenginys šioms laikmenoms įrašinėti (CD-RW, DVD-RW);
- Informacija ar duomenys įrašomi gana lėtai;
- Naudojant daugkartinio įrašymo laikmeną, ją ištrinti užima ne mažai laiko;
- Reikalinga speciali programinė įranga, norint įrašinėti informaciją ar duomenis į šias laikmenas;
- Gana lėtas informacijos ar duomenų talpinimas į laikmeną.

Šiuo metu populiariausios yra USB Flash tipo atmintinės. Iš pradžių USB Flash atmintinių populiarėjimą bei paplitimą stabdė palyginti didelė laikmenos kaina, tačiau kainoms nukritus šio tipo išorinės duomenų laikmenos tapo labai populiarios. Jos vertinamos dėl šių savo savybių:

- Yra ganėtinai didelės talpos (šiuo metu jau yra ir 128GB talpos atmintinių);
- Didelis rašymo ir skaitymo greitis;
- Užima labai mažai vietos;
- Lengvos;
- Nereikalinga jokia papildoma programinė įranga norint rašyti ar skaityti duomenis;
- Nereikalinga jokia papildoma, į kompiuterio standartinę komplektaciją, neįeinanti techninė ar programinė įranga;
- Nebijo subraižymų, nedidelių smūgių (yra net vandeniui atsparių);
- Veikia su bet kuria standartiniam vartotojui pritaikyta operacine sistema;
- Prijungi ir naudoji (*Plug and Play*).

Mūsų tyrimo sritis yra duomenų sauga, todėl analizuosime metodus, skirtus apsaugoti išorines informacijos laikmenas nuo jų turinio patekimo tretiems asmenims. Konkrečiau gilinsimės būtent į USB atmintinių duomenų apsaugą.

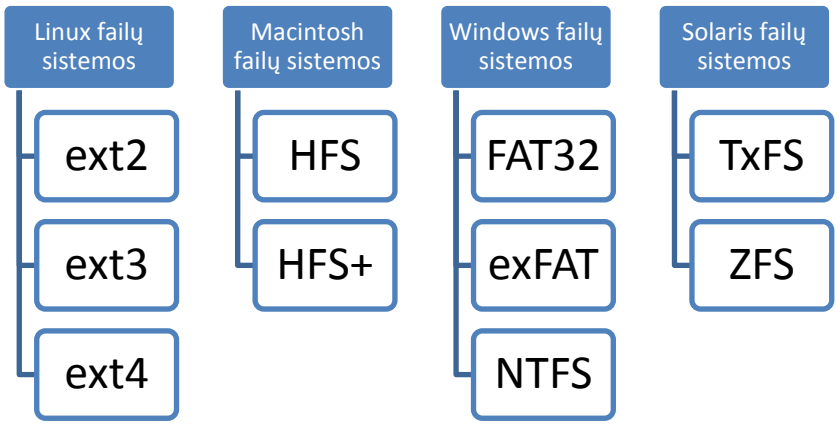
2.3. Aplinkos analizė

2.3.1. Duomenų įrašymo būdai

Norint įrašyti duomenis į CD ar DVD diską, USB atmintinę, išorinį kietąjį diską ar atminties kortelę reikia, kad jame būtų failų sistema. Failų sistema turėtų pasižymėti šiomis savybėmis:

- Suprantama skirtingose sistemose (turėtų būti galima kopijuoti failus *Windows*, *Mac* ar *Unix* sistemose);
- Failų sistemos formatas turėtų būti kuo paprastesnis. Tai ypač aktualu tokiose sistemose, kaip išmanieji telefonai, televizoriai, kameros ar fotoaparatai, kurios dar neturi galingos vidinės aparatūrinės įrangos Failų sistema turėtų būti suderinama su savo pirmtake, kad vis dar galėtume panaudoti seniau įrašytą informaciją naujose failų sistemose.

Žemiau pateiktos dabar naudojamos failų sistemos sugrupuotos pagal operacinės sistemos tipą. Toliau mes labiau gilinsimės į *Windows* terpei pritaikytoms failų sistemoms, nes ši operacinė sistema yra labiausiai paplitusi ir dauguma kitų įrenginių/operacinių sistemų gali nuskaityti failus, kurie įrašyti į laikmeną su viena iš *Windows* failų sistemų.



1 pav.: Sugrupuotos failų sistemos

FAT32

FAT (angl. *File Allocation Table*) – tai kompiuterio failų sistemos architektūra. FAT failų sistema yra gana paprasta techniniu požiūriu, tačiau tuo pačiu ir tvirta. Ši sistema siūlo pakankamai gerą kokybę net „mažo svorio" instaliacijoms, todėl yra plačiai naudojama ir palaikoma beveik visų esamų asmeninių kompiuterių

operacinių sistemų. Tai plačiai pritaikomas formatas, skirtas duomenų keitimuisi tarp kompiuterių ir kitų įrenginių, kurie buvo pagaminti nuo 1980 metų iki šių dienų.

Bėgant laikui FAT failų sistema tobulėjo, siekiant aprėpti kuo daugiau atminties. Kai vystėsi kietieji diskai, maksimalus klasterių skaičius žymiai padidėjo, todėl bitų kiekis, kurie buvo naudojami atpažinti kiekvienam klasteriui, didėjo. Taip keitėsi ir FAT sistemos pavadinimas, kuris buvo FAT, FAT12, FAT16 ir FAT32. Kiekvienas iš šių variantų yra tebenaudojami. FAT standartas taip pat buvo išplėstas kitais būdais, kad būtų išsaugotas atgalinis suderinamumas su egzistuojančia programine įranga. FAT32 sistema yra paskutinė FAT sistemos versija.

FAT32 ypatybės:

- FAT32 sistema palaiko daugiau nei 2 terabaitų kietuosius diskus ir kitas laikmenas;
- FAT32 išnaudoja vietą ekonomiškiau. FAT32 naudoja mažus klasterius (tai yra, 4-KB klasterius laikmenoms, kurios yra daugiau nei 8GB dydžio) , ko pasėkoje pasiekiamas nuo 10 iki 15 procentų geresnis laikmenos vietos išnaudojimas, palyginus su FAT16 ar FAT;
- FAT32 yra patikimesnė nei pirmtakės. FAT32 gali perkelti sisteminį aplanką ir naudoti atsarginę failų paskirstymo lentelės kopiją, vietoj originalios. Taip pat diskų įkrovos įrašas (angl. *boot disk record*) FAT32 laikmenose yra išplėstas, kad būtų įrašyta ir atsarginė kritinių duomenų struktūrų kopija. Todėl FAT32 sistemą turinčios laikmenos yra mažiau jautrios duomenų praradimo prasme nei turinčios FAT16;
- FAT32 yra lankstesnė, lyginant su pirmtakėmis. Pagrindinis sisteminis aplankas, esantis FAT32 failų sistemoje yra paprasta klasterio grandinė, todėl jis gali būti patalpintas bet kurioje laikmenos vietoje. Ankstesni sisteminio aplanko apribojimai daugiau nebeegzistuoja. Be to, failų paskirstymo lentelės dubliavimas (angl. *mirroring*) gali būti išjungtas, kas leidžia naudoti kitą, o ne pačią pirmą failų paskirstymo lentelę, kaip aktyvią. Šios ypatybės leidžia dinamiškai keisti FAT32 failų sistemų dydį. Tai leidžia vienoje laikmenoje turėti kelias failų sistemas, kurios kompiuterio matomos kaip atskiri kietieji diskai. [1, 2]

NTFS

NTFS – *Microsoft* sukurta, *Windows NT* tipo operacinėse sistemose (*Windows 2000, Windows XP* ir pan.) naudojama failų sistema.

NTFS buvo sukurta HPFS failų sistemos pagrindu ir buvo skirta pakeisti pasenusią FAT failų sistemą naujesnėse *Windows* versijose. Skirtingai nuo FAT, NTFS palaiko naudotojų teisių kontrolę, leisdama failus skaityti ar rašyti tik apibrėžtiems naudotojams ar jų grupėms, naujesnės NTFS versijos palaiko kodavimą ir pan.

NTFS pasižymi dideliu našumu, todėl ji naudojama net ir dabartinėse naujausiose *Microsoft Windows* versijose. Ji palaiko failo lygio apsaugą, kompresiją ir auditą. Taip pat ši failų sistema palaiko labai didelės apimties diskus ir laikmenas, bei galingus duomenų laikymo sprendimus tokius kaip RAID.

NTFS istorija

1980 metais, *Microsoft* ir IBM susivienijo, kad sukurtų pažangią naujos kartos operacinę sistemą, kuri turėtų grafinę vartotojo sąsają. Projekto rezultatu tapo OS/2. Kiek vėliau *Microsoft* ir IBM ėmė nesutarti dėl įvairių su šia sistema iškilusių problemų, todėl teko atsiskirti. OS/2 liko kaip IBM projektas, o *Microsoft* pradėjo dirbti prie *Windows NT* projekto. HPFS failų sistema, kuri buvo naudojama OS/2 operacinėje sistemoje turėjo naudingų privalumų. Kai *Microsoft* sukūrė savo naująją operacinę sistema, daugelis HPFS privalumų buvo panaudota NTFS failų sistemoje. Turbūt todėl abi failų sistemos HPFS ir NTFS turi tą patį identifikacijos tipo kodą (07). FAT failų sistema turi daugiau nei devynis atpažinimo kodus, kiekvienai iš versijų (FAT12, FAT16, FAT32 ir tt). Kad atpažinti failų sistemą, kuri HPFS ar NTFS tai failų sistema diske, reikalingi papildomi patikrinimai.

NTFS versijos

- v1.0 naudota *Windows NT 3.1*, 1993 metais.
- v1.1 naudota *Windows NT 3.5*, 1994 metais.
- v1.2 naudota *Windows NT 3.51* 1995 metais ir *Windows NT 4.0* 1996 metais, vėliau įgavo pavadinimą NTFS 4.0, nes operacinės sistemos versija buvo 4.0.

- v3.0 naudota *Windows 2000*, kuri taip pat turėjo pavadinimą „NTFS V5.0“, dėl to kad operacinės sistemos versija buvo 5.0.
- v3.1 naudota *Windows XP* 2001 metais, taip pat pagal operacinės sistemos versiją buvo vadinama „NTFS 5.1“, *Windows Server 2003* naudojo „NTFS 5.2“, *Windows Vista 2005* metais naudojo „NTFS V6.0“, taip pat ją naudojo ir *Windows Server 2008*.

V1.0 ir V1.2 nėra suderinamos, jei failų sistema yra įrašoma su NT3.5X, ji nebus skaitoma NT 3.1, tačiau šis nesuderinamumas ištaisomas įrašant atnaujinimą, kuris yra NT 3.5x instaliaciniame kompaktiniame diske. Šis atnaujinimas taip pat prideda FAT ilgų failų vardų palaikymą. V1.2 palaikė suspaustus failus, vadinamus srautais (angl. *streams*) paremtus ACL apsauga. V3.0 pridėjo disko kvotas, šifravimą, failų atskyrimą, taškų paskirstymą, numerio sekos atnaujinimo žurnalus (USN), $Extend aplanką ir jo failus bei perorganizuotus apsaugos deskriptorius. Keli ar keliolika failų, kurie naudoja ta patį saugumo lygį, gali naudoti ta patį deskriptorių. V3.1 praplėtė *Master File Table* (MFT) prieigas su MFT įrašo numeriu, kuris buvo naudingas atstatant MFT failus.

Windows Vista pristatė tranzakcinį NTFS, NTFS simbolių nuorodas ir „*Self Healing*“ funkciją.

Kaip veikia NTFS

Kai suformatuojamas kietasis diskas, jis padalinamas į skirsnius arba į pagrindines dalis, kurios išdėstomos visame kietojo disko dydyje. Kiekvienam skirsniui, operacinė sistema pasilieka takelį, reikalingą laikyti visiems operacinės sistemos failams. Kiekvienas failas laikomas viename ar daugiau kietojo disko klasterių ar disko vietų su paskirtu dydžiu. Naudojant NTFS, klasterių dydžiai gali būti nuo 512 baitų iki 64 kilobaitų. *Windows NT* turi rekomenduojamą pagal nutylėjimą klasterio dydį, kiekvienam laikmenos ar kietojo disko dydžiui. Pvz., 4 GB kietajam diskui, pagal nutylėjimą esantis klasterio dydis yra 4 kilobaitai. 4 KB klasterių sistemoje, klasteriai yra nedalinami, net mažiausias failas užima vieną klasterį, o failas kuris yra jau 4.1 KB, užima du klasterius arba 8 KB.

Klasterio dydžio parinkimas naudojamas disko išnaudojimo galimybių ir failų pasiekiamumo greičio pagerinimui. Naudojant NTFS, kuo didesnis yra diskas, tuo didesnis yra pagal nutylėjimą esantis klasterio dydis, taip yra todėl, kad galvojama jog

vartotojui priimtiniau yra pasiekti esamus failus greičiau, nei turėti šiek tiek daugiau vietos, nes būtent vietos sąskaita ir pasiekiamas greitesnis failų prieinamumas.

Kai kuriamas failas, naudojant NTFS, apie jį yra sukuriamas įrašas specialiame faile, kuris vadinamas pagrindine failų lentele (*Master File Table* (MFT)). Įrašas naudojamas išsibarsčiusių failo klasterių suradimui, jei jie išsibarstytų. NTFS bando surasti artimiausią vietą diske, kurioje būtų galima patalpinti visą failą (visus jo klasterius).

Kiekvienas failas su savo atributais turi ir aprašą apie juos, tai vadinama metaduomenimis.[1,2]

exFAT

Failų sistemą exFAT (angl. *Extended File Allocation Table*), dar žinomą kaip FAT64, 2009 *Microsoft* korporacija pristatė kaip pakaitalą senstančiai FAT32 failų sistemai ir yra specialiai pritaikyta nešiojamoms atmintinėms. Nuo to laiko ši failų sistema nebuvo plačiai paplitusi, bet „*SD Card Association*“ ją pasirinko kaip SDXC kortelių standartinę failų sistemą, taip prisidėdama prie šios sistemos populiarinimo.

exFAT failų sistema leidžia valdyti daug didesnius duomenų kiekius lyginant su konkurencinėmis sistemomis ir tuo pačiu metu leidžia pasiekti didelio greičio duomenų mainus tarp nešiojamųjų atmintinių ir kompiuterių.

Visi FAT32 failų sistemos trūkumai buvo pašalinti iš šios versijos, tad exFAT pasižymi šiomis failų sistemos savybėmis:

- Palaiko failų dydžius iki 64 ZiB (angl. *zebibyte*);
- Didelis sektorių dydžių pasirinkimas - nuo 512 iki 4096 baitų;
- Rekomenduojamas failų sistemos ar failo dydis – 512 TiB (angl. *tebibyte*);
- Tuščių bitų metodika;
- Failų pavadinimai *Unicode* UTF-16 koduote iki 255 ženklų;
- Iki 2,796,202 failų vienoje direktorijoje;

exFAT yra viena iš naujausių failų sistemų. Ji geba susitvarkyti su dideliais duomenų kiekiais ir atlikti failų mainus tarp nešiojamųjų laikmenų ir kompiuterių greičiau, nei konkurentės. Tai jai padaryti leidžia unikali duomenų struktūra, kurios dėka sumažėja I/O (angl. *input/output*) proceso laikas. Dėl šių savybių ateityje

exFAT failų sistema gali tapti viena dažniausiai naudojamų failų sistemų nešiojamuose įrenginiuose.

Nors exFAT failų sistema turi daug pliusų, tačiau vienas iš didesnių trūkumų, dėl ko ji ne taip sparčiai išplitusi, - tai, kad *Microsoft* vis dar neišleido oficialių šios failų sistemos specifikacijų ir, norint kurti sistemas su šia failų sistema, reikia nusipirkti ribojančią licenciją iš *Microsoft* korporacijos. [1,3,4]

2.3.2. **Failų sistemų palyginimas**

Žemiau pateiktas FAT32, exFAT ir NTFS failų sistemų palyginimas. Failų sistemos lyginamos atsižvelgiant į šiuos 4 faktorius, kad būtų išrinkta pati patikimiausia failų sistema duomenims apsaugoti ir keistis:

1. Failų sistemos duomenų limitas;
2. Failų sistemos metaduomenys;
3. Failų sistemos charakteristikos;
4. Duomenų paskirstymo strategijos.

1 lentelė: Failų sistemos duomenų limito palyginimas

Failų sistema	Maksimalus failo pavadinimo ilgis	Maksimalus failo dydis	Maksimalus particijos dydis
FAT32	255 ženklai	4 GB	2 TB
exFAT	255 ženklai	127 PB	64 ZB, rekomenduojama 512 TB
NTFS	255 ženklai	16 EB	16 EB

2 lentelė: Failų sistemos metaduomenų palyginimas

Failų sistema	Failo sukūrimo žymė	Kontrolinė suma
FAT32	Dalinai	-
exFAT	+	Dalinai
NTFS	+	-

3 lentelė: Failų sistemos charakteristikų palyginimas

Failų sistema	Jautri didžiosioms/mažosioms raidėms	Failų kaitos išrašas	Momentinė nuotrauka	Šifravimas
FAT32	-	-	-	-
exFAT	-	-	-	-
NTFS	+	+	Dalinai	+

4 lentelė: Duomenų paskirstymo strategijos palyginimas

Failų sistema	Pavieniai failai	Skaidrus suspaudimas	Skirtingas bloko dydis	Besitęsiantis
FAT32	-	-	-	-
exFAT	-	-	-	-
NTFS	+	+	-	+

Iš duomenų, kuriuos matome 1-oje lentelėje, galima daryti išvadą, jog turinti didžiausią duomenų limitą yra NTFS failų sistema. Nuo jos ne daug atsilieka exFAT, kuri, kad ir teoriškai leidžia turėti didesnį skirsnį (angl. *partition*) nei NTFS, bet rekomenduojamas dydis yra mažesnis.

Iš antrosios lentelės matome, kad *Microsoft* failų sistema exFAT aplenkia šioje srityje savo konkurentus ir leidžia talpinti daugiau metaduomenų, nei kitos mūsų apžvelgiamos failų sistemos.

Iš duomenų, pateiktų 3-joje lentelėje, galima spręsti, jog NTFS failų sistema turi daugiausiai papildomų galimybių lyginant su exFAT ir FAT32.

Iš ketvirtosios lentelės matome, jog NTFS failų sistema yra daug efektyvesnę duomenų paskirstymo strategiją, nei kitos mūsų aptariamos failų sistemos.

Apibendrinant visų mūsų apžvelgiamų failų sistemų lyginimo rezultatus galima atvaizduoti šioje 5 lentelėje:

5 lentelė: Palyginimo rezultatai

Faktorius	Efektyvi failų sistema	Artimas konkurentas
Failų sistemos duomenų limitas	NTFS	exFAT
Failų sistemos metaduomenys	exFAT	NTFS
Failų sistemos charakteristikos	NTFS	-
Duomenų paskirstymo strategijos	NTFS	-

2.3.3. **Apsauga nuo duomenų pakeitimo ar kopijavimo**

Kadangi USB atmintinėse saugomą informaciją yra gana paprasta kopijuoti ar kitaip pakeisti, tai ir priemonės, jai apsaugoti, yra sudėtingiau pritaikomos. Yra nemažai priemonių, kurios padeda ne tik, kad neleisti į USB atmintinę rašyti naujų duomenų ar pakeisti ten jau esančius, bet ir apskritai neleidžia tokių failų kopijuoti ar nuskaityti neturint atitinkamos programinės ar aparatinės įrangos. Vieni žinomiausių iš tokių metodų yra skaitmeninis teisių valdymas (angl. *DRM – Digital Rights Management*), CD-ROM skirsnius, tik skaityti (angl. *Read Only*) skirsnius, „Dummy“ failai, specialios licencijos ar DRM schemos paslėptos nematomoje atminties dalyje ir laiko limitai.

USB CD-ROM ir Read Only skirsniai

Jei vartotojui reikia pačios paprasčiausios apsaugos ir/ar jo turima USB atmintinė neturi specialaus apsaugos nuo duomenų rašymo jungtuko, kurio pagalba galima išjungti arba vėl įjungti galimybę rašyti duomenis į atmintinę, vartotojas gali iš dalies ar visos atmintinės laisvos vietos sukurti skirsnį „tik skaitymui“, kuriame duomenis bus galima skaityti ir nebus galima jų redaguoti ar kitaip keisti.

Specialias USB atmintines galima sukonfigūruoti taip, kad jose būtų galima sukurti USB CD-ROM skirsnius, kurie pilnai palaiko *Autorun* ir *Autoplay* funkcionalumą ten įrašytoms aplikacijoms, dokumentams ar html failams. Tokie skirsniai yra apsaugotos nuo ištrynimo ar pašalinimo iš USB atmintinės naudojant bet kurias standartines formatavimo ir/ar trynimo priemones.

„Dummy“ failai

Sukūrus CD-ROM skirsnį, USB atmintinėje galima taikyti tokius apsaugos metodus, kurie naudojami norint apsaugoti CD ar DVD diskus nuo kopijavimo. Vienas iš tokių metodų yra panaudoti „*Dummy*“ failus arba modifikuoti naudojamą failų sistemą ISO9660, kuri dažniausiai naudojama tokio tipo skirsniuose. Taigi norint apsaugoti duomenis nuo kopijavimo jiems priskiriami failų plėtiniai, kurie būna ilgesni nei yra numatyta šios failų sistemos architektūroje. Šie plėtiniai (*Joliet, El Torrito* ir *RockRidge*) gali būti naudojami ISO9660 failų sistemoje, bet juos modifikavus jie tampa nebesuderinami su failų sistema.

Taip pat galima pakeisti failų sistemą, kad ji būtų pritaikyta tik duomenų atvaizdavimui. Tai padaryti galima, kai į failų sistemą norimi duomenys patalpinami

tvarkingai, o po to sistema perkuriama taip, kad liktų tik galimybė šiuos duomenis atvaizduoti.

Failų sistemos sektoriai

Kiekviena *flash* tipo atmintinė yra formatuojama į pasirinkto dydžio (pvz.: 512, 1024, 2048 ar 4096 baitų) sektorius, kad būtų paliktas suderinamumas su standžiaisiais diskais. Taigi prijungus USB atmintinę prie bet kokio įrenginio, kuris gali jas atpažinti, sektorius būna pirminė duomenų struktūra, kurią nuskaito specializuota programinė įranga ar operacinė sistema. Sektorių sudaro vartotojo ir struktūrinė informacija, kuri susideda iš:

- Sektoriaus numerio;
- Klaidas aptinkančio kodo;
- Klaidas ištaisančio kodo.

Taigi, galima panaudoti tuos klaidas aptinkančius ir ištaisančius kodus ir juos „suklastoti“ duomenų įrašyto metu, kad apsaugotume duomenis nuo kopijavimo. Tokie duomenys paprastai aplikacijai ar vartotojui atrodo kaip neperskaitomi ar pažeisti, todėl tokios informacija tiesiog nekopijuojama. Tuo tarpu, apsaugos nuo kopijavimo programinė įranga bando nuskaityti tų suklastotų sektorių klaidų kodus ir taip leidžia vartotojui vėl atgaminti įrenginyje įrašytus duomenis.

Laiko limitas

Turint specialią atmintinę, kurioje yra laiko limito nustatymo funkcija, galima šios atmintinės savininkui nustatyti laiko limitą (datą ar tikslų laiką) iki kada galima naudotis atmintinėje esančiais duomenimis. Po nustatytos datos atmintinės turinys negrįžtamai ištrinamas ir duomenys diske prarandami.

2.3.4. Šifravimas

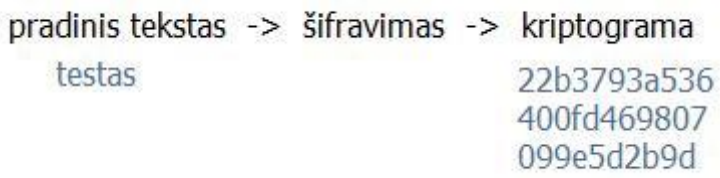
Kriptografija

Kriptografija - tai mokslas naudojantis matematiką užšifruoti ir iššifruoti informacijai. Kriptografija leidžia apsaugoti ypač slaptą informaciją ir taip įgalina siųsti ją nesaugiais tinklais kaip internetas, taigi jos negali perskaityti niekas kitas išskyrus gavėją.

Pradinė informacija, kurią galima skaityti nepanaudojus jokių specialių priemonių, vadinama paprastu tekstu. Metodas, naudojantis paprastą tekstą ir paverčiantis jį į gausybę nieko bendro tarp savęs neturinčių simbolių vadinamas

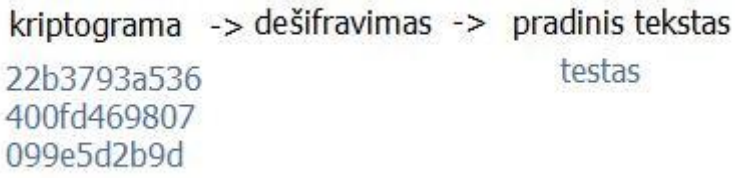
šifravimu, šifru arba algoritmu (ang. *cipher*, *algorithm*). O toks tekstas – užšifruotu tekstu arba kriptograma (angl. *ciphertext, ciphertext*). Pati duomenų šifravimo schema atrodytų taip:

1. Užšifravimas:



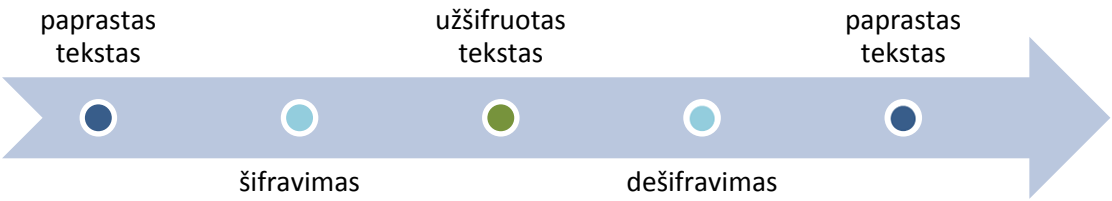
2 pav.: Užšifravimo schema.

2. Dešifravimas:



3 pav.: Dešifravimo schema.

3. O viską susumavus gaunama tokia veiksmų seka:



4 pav.: Šifravimo ir dešifravimo schema.

Kriptografijos saugumui tikrinti yra naudojama kriptoanalizė. Paprasta kriptoanalizė susideda iš loginių kombinacijų ir matematinių įrankių panaudojimo. Jos pagalba geri kriptoanalitikai ne tik sugeba dešifruoti sunkiausius algoritmus, bet ir tuo pačiu, atrasdami spragų viename ar kitame algoritme, tas spragas pašalinti ir taip prisidėti prie saugesnių metodų ar algoritmų sukūrimo. Kriptoanalitikų naudojamas atakas galima suskirstyti į tokias kategorijas:

1. *Kriptoanalitinė ataka, kai yra žinomas tik užšifruotas tekstas*. Įsilaužėlis turi tik kelių dokumentų kriptogramas K_1, K_2, ..., K_i, kurios užšifruotos naudojant tą patį šifravimo algoritmą A_R. Jo

tikslas yra gauti dokumentus D_1, D_2, ..., D_i arba apskaičiuoti raktą R, kuris buvo panaudotas šių dokumentų užšifravimui;

2. *Kriptoanalitinė ataka, kai žinomas dokumentas*. Įsilaužėlis turi keleto dokumentų kriptogramas K_1, K_2, ..., K_i ir jas atitinkančius dokumentus D_1, D_2, ..., D_i. Jo tikslas šiuo atveju yra surasti raktą R, kuris buvo panaudotas užšifravimui, arba naujų dokumentų, užšifruotų tuo pačiu raktu, dešifravimo algoritmą A_R.
3. *Kriptoanalitinė ataka, kai galima pasirinkti dokumentą*. Įsilaužėlis turi ne tik kriptogramas K_1, K_2, ..., K_i ir jas atitinkančius dokumentus D_1, D_2, ..., D_i, bet ir gali atsirinkti dokumentus, kuriuos gauna užšifruotus. Tokia kriptoanalizė bus daug efektyvesnė, nes įsilaužėlis gali atsirinkti tokius dokumento blokus užšifravimui, kurie jam duos daugiausiai informacijos apie raktą. Taigi čia įsilaužėlio tikslas yra surasti raktą R, kuris buvo panaudotas dokumentų užšifravimui, arba naujų dokumentų, užšifruotų tuo pačiu raktu, dešifravimo algoritmą A_R.
4. *Kriptoanalitinė ataka, esant adaptyviam dokumento parinkimui*. Tai toks atakos variantas, kai įsilaužėlis gali pasirinkti norimą dokumentą. Tad parenkamas ne tik dokumentas, kuris po to užšifruojamas, bet ir galima keisti savo pasirinkimą priklausomai nuo ankstesniojo šifravimo rezultatų. Taigi pirmiausia įsilaužėlis gali parinkti mažesnę ir palankesnę dokumento dalį, o po to parinkti ir kitą dokumento dalį, priklausomai nuo pirmojo parinkimo rezultatų.
5. *Kriptoanalitinė ataka, panaudojant pasirinktą užšifruotą tekstą*. Šiuo atveju įsilaužėlis gali pasirinkti skirtingas kriptogramas K_1, K_2, ..., K_i ir turi atitinkamus dokumentus D_1, D_2, ..., D_i. Jo tikslas – surasti šifro raktą.
6. *Kriptoanalitinė ataka visų raktų perrinkimo metodu*. Šiuo atveju įsilaužėlis turi kažkokį užšifruotą tekstą ir bando visus galimus rakto variantus tikrindamas, ar gautas dokumentas yra prasmingas. Toks metodas reikalauja didelių skaičiavimo resursų ir vadinamas jėgos ataka (angl. *Brute Force*).[5]

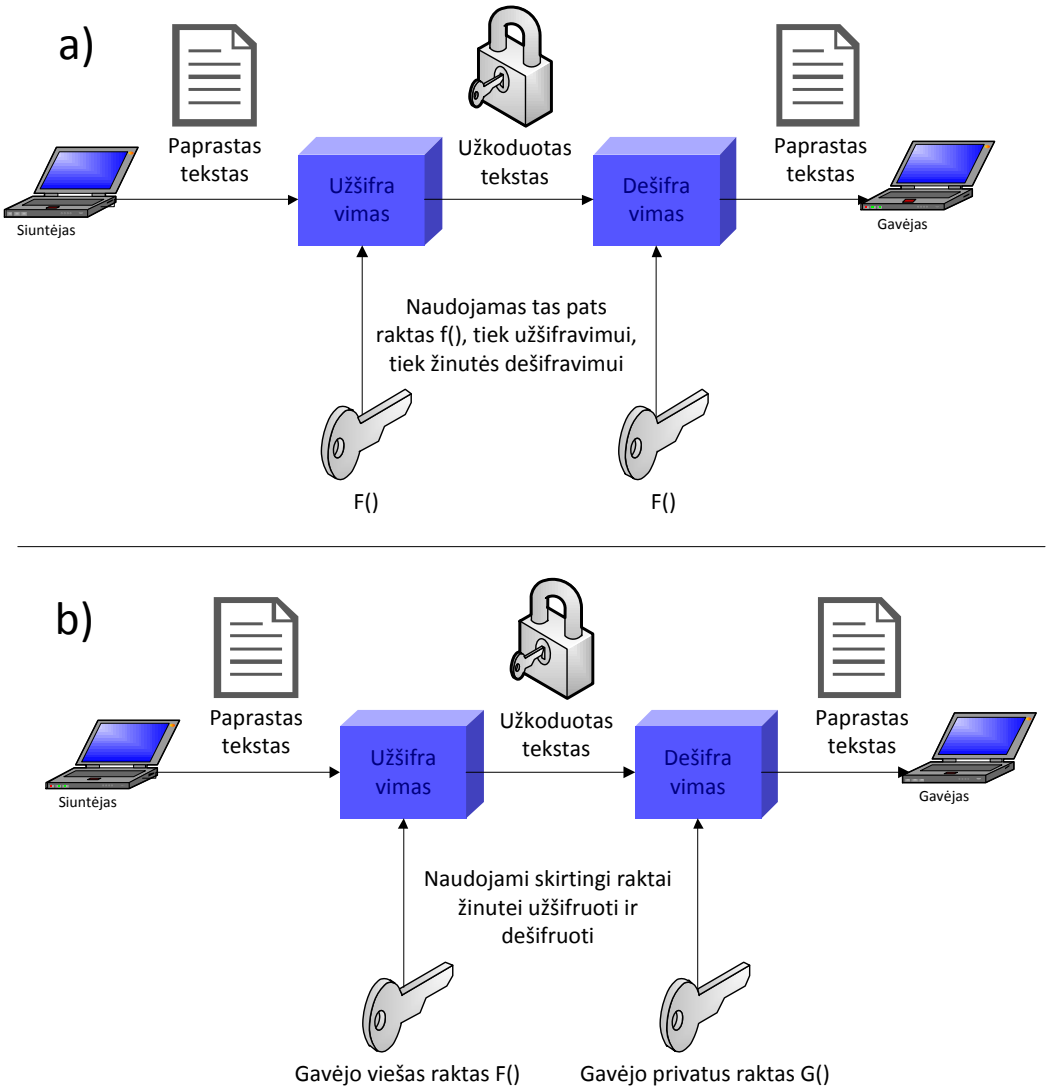
Simetrinio rakto kriptografija

Simetrinio rakto kriptografija – tai toks šifravimo būdas, kai slapto rakto algoritmai naudoja tą patį raktą informacijai užšifruoti ir dešifruoti. Lyginant su viešojo rakto kriptografija, kur naudojami du raktai (pirmas - viešasis raktas, naudojamas duomenims užšifruoti, o antras - privatus, naudojamas dešifruoti). Simetrinio rakto kriptografijos metodai yra daug paprastesni ir greitesni, o patys raktai naudojami šifravimui ir dešifravimui – daug trumpesni. Tačiau pagrindinis šios kriptografijos šakos trūkumas yra rakto perdavimas gavėjui, jis turi vykti saugiu kanalu.

Žemiau pateiktame 5 paveikslėlyje pavaizduota:

a) Simetrinio rakto kriptografijos pavyzdys;

b) Viešojo rakto kriptografijos pavyzdys.



5 pav.: Simetrinio ir asimetrinio (viešojo) rakto kriptografijos palyginimas

Algoritmai, kurie naudojami simetrinio rakto kriptografijoje, skirstomi į srauto ir blokinius:

- Srauto algoritmai duomenis šifruoja po vieną bitą. Labiausiai paplitę RC4, RC5 simetrinio rakto kriptografija paremti algoritmai;
- Blokiniai algoritmai šifruoja įvairaus dydžio blokais. Vieni žinomiausių yra DES, IDEA, AES.[6]

Viešojo rakto kriptografija

Viešo rakto algoritmai naudoja skirtingą raktą užšifravimui ir dešifravimui. Todėl dešifravimo raktas praktiškai negali būti išvestas iš užšifravimo rakto. Viešo rakto metodai yra svarbūs, nes jie gali būti naudojami perduodant užkodavimo raktus ir kitus duomenis saugiai net jei šalys neturi galimybės susitarti dėl saugumo rakto privačiai. Pagrindinė problema, naudojant viešojo rakto metodus, yra konfidencialumas, t.y. užtikrinimas, kad raktas priklauso tam asmeniui, kuris teigia jį turįs. Dažniausiai konfidencialumą užtikrinantis principas yra viešo rakto infrastruktūros (PKI) naudojimas, tai yra schema, kai viena ar daugiau trečiųjų šalių, vadinamų sertifikatų tiekėjais (angl. *key servers*), garantuoja raktų porų priklausomybę. Kitas principas, kurį naudoja PGP, kai remiamasi „pasitikėjimo“ principu.

Deja, visos viešo rakto sistemos nėra atsparios jėgos ataktos panaudojimo atveju. Tačiau šios atakos netenka prasmės ir būna nepraktiškos, jei skaičiavimų skaičius reikalingas nulaužti raktą yra per didelis potencialiems nulaužėjams. Daugeliu atvejų skaičiavimų skaičių galima padidinti tiesiog panaudojant ilgesnius raktus. Tačiau, kai kuriems metodams skaičiavimų skaičius yra gerokai mažesnis nei naudojant jėgos ataką. Tiek RSA (sutrumpinimas reiškia algoritmo kūrėjų pavardžių pirmąsias raides - Ron *Rivest*, Adi *Shamir* ir Leonard *Adleman*), tiek ElGamal (kūrėjas Taher Elgamal‘as) viešojo rakto šifrai turi galimus nulaužimo būdus, greitesnius nei pasitelkus jėgos atakos metodą. Šifrų įveikiamumas padidėjo išaugus kompiuterių galingumui bei padarius naujus atradimus matematikos srityje. Bet ir tai galima apeiti parenkant pakankamai ilgus raktus, tokius, kad informacijos dešifravimas užtruktų tiek ilgai, jog ją dešifravus ji jau nebebūtų aktuali įsilaužėliui.

Keliuose perspektyviuose asimetriniuose raktų algoritmuose rastos rimtos saugumo spragos. Pavyzdžiui - galimybė trečiajai šaliai perimti viešų raktų apsikeitimą ir pakeisti viešus raktus. Pasinaudojus šia spraga pranešimai gali būti

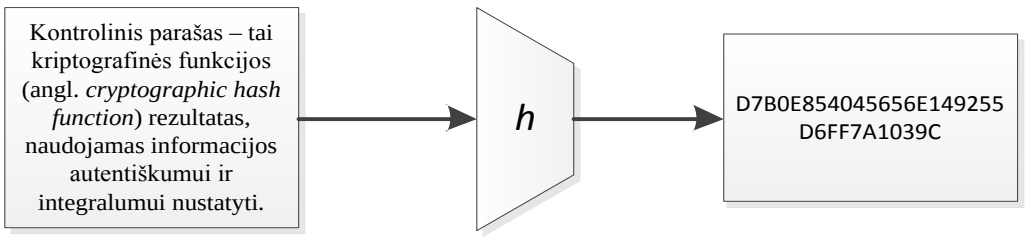
perimti ir užšifruoti, dešifruoti ir vėl užšifruoti naudojant korektišką raktą, kad nebūtų sukeltas įtarimas. Tai nelengva padaryti, tačiau vis dėlto įmanoma, ypač kai komunikacija vyksta nesaugia terpe (pavyzdžiui, internetas). Taip perimti užkoduotą pranešimą nesunkiai gali atlikti bet koks interneto paslaugų tiekėjo darbuotojas ar žmogus galintis prisijungti prie nesaugaus tinklo. Tokį pažeidžiamumą pašalinti leidžia sertifikatai, trečių šalių patikinimo patikrinimas.

Bet kokį viešą raktą gali žinoti didelis vartotojų skaičius, todėl, jei prireikia viešą raktą pakeisti ar užblokuoti, tai gali paveikti visus vartotojus (apie kuriuos galima ir nežinoti). Be to gali prireikti ilgo laiko tarpo norint tokį raktą pakeisti. Todėl rekomenduoja viešojo rakto kriptografija grįstų metodų nenaudoti sistemose, kurios privalo reaguoti į įvykius realiu laiku.[7,8]

Kontrolinis parašas

Kontrolinis parašas – tai kriptografinės funkcijos (angl. *cryptographic hash function*) rezultatas, naudojamas informacijos autentiškumui ir integralumui nustatyti. Aštuntajame dešimtmetyje maišos (angl. *hash*) funkcijos buvo pristatytos kaip įrankis, kuris turėjo pagerinti kriptografijos metodų saugumą padėdamas nustatyti pateiktos informacijos autentiškumą. Greitai pasidarė aišku, jog maišos algoritmai labai efektyviai padeda išspręsti saugumo problemas kylančias ir telekomunikacijų ar kompiuterių tinklų srityse.

Maišos funkcija pagal jai pateiktus duomenis (dokumentą, failą ir pan.) apskaičiuojama fiksuoto ilgio tos duomenų santrumpa (angl. *digest*). Šią santrumpą užšifravus privačiu raktu gaunamas kontrolinis parašas. Žemiau pateiktas paprastas pavyzdys iliustruojantis maišos algoritmo veikimą:



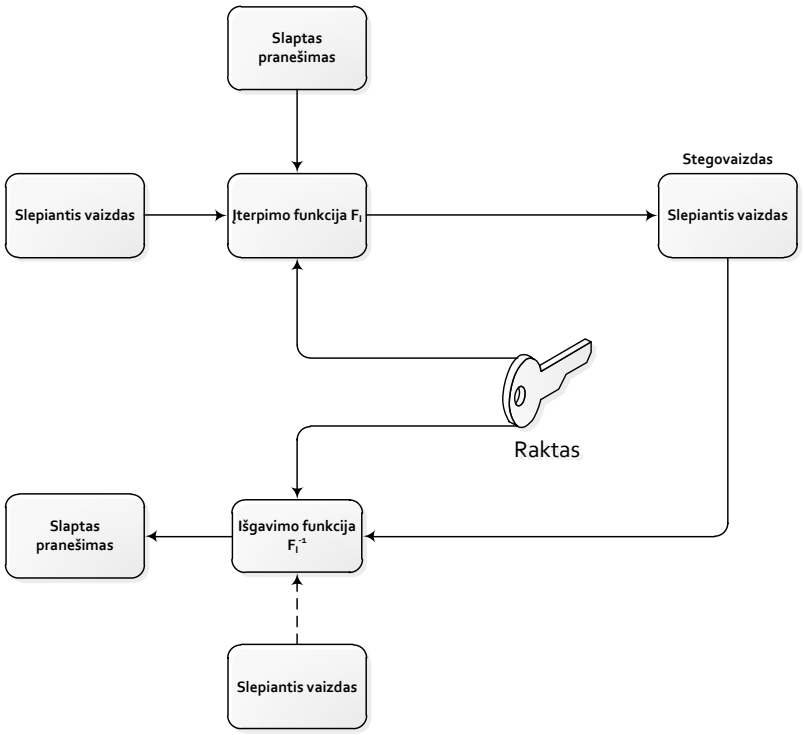
6 pav.: Maišos funkcijos pavyzdys

Visi kontrolinio parašo algoritmai iš prigimties yra neatsparūs jėgos atakoms. Tai yra todėl, kad šie algoritmai turi baigtinį galimų parašų skaičių ir anksčiau buvo manoma, jog nulaužti vieną ar kitą algoritmą užtruktų labai daug metų, tačiau vystantis kriptoanalitiniams būdams ir didėjant kompiuterių pajėgumams dabar populiariausių MD5 ar HAVAL-128 algoritmų sugeneruotoms reikšmėms kolizijas rasti galima per kelias valandas, o SHA-1 algoritmo sudėtingumą galimą sumažinti tiek, jog koliziją jau galima rasti priimtino laiko ribose.[9,10]

Steganografija

Steganografija – tai informacijos slėpimas. Terminas steganografija yra kilęs iš graikų kalbos žodžio ir verčiamas kaip paslėptas tekstas.

Steganografija yra artima kriptografijos giminaitė. Kriptografija sudėlioja pranešimą taip, kad jo būtų neįmanoma suprasti, tuo tarpu steganografija pranešimą paslepia taip, kad jo nebūtų galima pamatyti. Užšifruotas tekstas gali patraukti dėmesį, o „nematomas“ – ne. Steganografija gali būti taikoma įvairiais tikslais. Galima paminėti vandens ženklų naudojimą apsaugai nuo kopijavimo, internete skelbiamų nuotraukų žymėjimą, vertingos informacijos steganografavimą konfidencialumui išsaugoti.



7 pav.: Steganografijos principas

Šiuolaikinė steganografija apima informacijos slėpimą ne tik skaitmeniniuose paveikslėliuose, bet ir garso rinkmenose. Norint paslėpti pranešimą vaizde reikia dviejų rinkmenų:

a) slepiančios, kurioje bus talpinamas slaptas pranešimas;
b) slepiamo pranešimo.

Pranešimas gali būti paprastas tekstas, užšifruotas tekstas, garso rinkmena ar vaizdo medžiaga. Sujungus slepiantį vaizdą ir slepiamą pranešimą, gaunamas stegovaizdas. Tokiam pranešimui paslėpti ir po to atidengti naudojamas pasirinktas stegoraktas (slaptažodis), kuris įvedamas specialia steganografijos programine įranga norint iššifruoti gautą slepiantį vaizdą (7 pav.).

Slepiamas pranešimas dažniausiai iš pradžių užšifruojamas, o tik paskui įkomponuojamas į slepiantį vaizdą. Taigi tik tas, kuris žino, kaip buvo įkomponuotas pranešimas, gali jį aptikti ir, jei po to reikia, iššifruoti.

Skaitmeniniai vaizdai dažniausiai įrašomi į 8, 24 ar 32 bitų spalvų gylio rinkmenas. Dauguma steganografijos priemonių naudoja mažiausiai reikšmingo bito (angl. *Least Significant Bit* arba *LSB*) metodą. Pavyzdžiui, dešimtainis skaičius 187 aštuoniais bitais užrašomas kaip *10111011*. Dešiniausias bitas vadinamas mažiausiai reikšmingu, nes jį pakeitus mažiausiai keičiasi skaičius. Slepiamo pranešimo dvejetainiai duomenys po vieną bitą įterpiami į slepiančio vaizdo kiekvieno vaizdo elemento bito mažiausiai reikšmingą bitą. 8 bitų gylio vaizdų atveju kiekviename vaizdo elemente galima paslėpti tik po vieną bitą, 24 bitų gylio vaizde – po tris bitus. Pavyzdžiui, į 24 bitų gylio vaizdo elementą, kurio reikšmė yra *00010001 00001100 10110111*, įterpus slaptus bitus *110*, jis pakeičiamas į *00010001 00001101 10110110*. Akivaizdu, jog šių dviejų vaizdo elementų skirtumą sunku pastebėti.

Mažiausiai reikšmingo bito metodas ypač tinka pranešimams pilkų atspalvių paveikslėliuose slėpti. Vieno paveikslėlio baito net du nereikšmingiausius bitus pakeitus slepiamos informacijos bitais, žmogaus akis pokyčio paveikslėlyje nematys.

Vienas didžiausių LSB metodo trūkumas – tai tokiu metodu slepiantį vaizdą suspaudus, pakeitus rezoliuciją, formatą ar apkarpius, paslėpta informacija yra išdarkoma. Pavyzdžiui, jei *.png arba *.bmp tipo rinkmenoje paslėpsite pranešimą, paskui pakeisite rinkmenos formatą į *.jpeg, o vėliau bandysite jį vėl atkeisti, paslėpto pranešimo išgauti nebegalėsite.[11,12]

2.4. Vartotojų analizė

2.4.1. Vartotojų aibė, tipai ir savybės

Panašių sistemų vartotojai yra visi žmonės, kuriems reikalingas duomenų saugumas išorinėse nešiojamose duomenų laikmenose. Tačiau tokios sistemos yra dažniau orientuotos į paprastus kompiuterių vartotojus. Didesnėse įmonėse saugos specialistai žino, jog vien užšifravus dar nereiškia, kad duomenys bus saugūs, todėl papildomai tokiose įmonėse egzistuoja griežtos saugomo procedūros, o darbuotojai dalyvauja apmokymuose, kaip reikia elgtis duomenų praradimo ar vagystės atveju. Šie vartotojai gali būti skirstomi į:

- Paprastus vartotojus, siekiančius savo informacijos konfidencialumo;
- Įmonių vadovus bei darbuotojus;
- Valstybinių įstaigų vadovus ar darbuotojus;
- Menininkai.

2.4.2. Vartotojų tikslai ir problemos

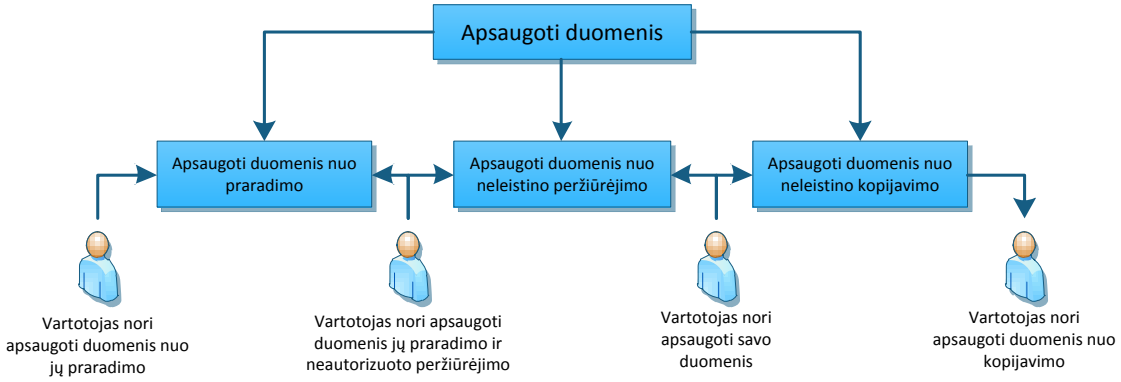
Paprasti vartotojai siekia, jog informacija be jų žinios neatitektų niekam kitam. USB atmintinės yra nedidelės, todėl jas gan lengva pamesti, tai labai dažnai ir atsitinka šio tipo vartotojų tarpe, todėl bet kas, radęs atmintinę, gali peržiūrėti joje esančią informaciją ar duomenis ir daryti jų kopijas. Tai ir yra pagrindinė šio tipo vartotojų problema.

Pagrindinis įmonių vadovų ir jų darbuotojų tikslas – apsaugoti savo informaciją ir duomenis nuo trečiųjų asmenų. Tai yra todėl, kad pasinaudojus atmintinės turiniu iš prarastosios laikmenos galima sužinoti slaptų ar konfidencialių faktų apie įmonės strategiją, finansus, įmonės politiką ir t.t. Ši informacija gali būti labai naudinga konkurentams. Problema yra ta, kad praradus atmintinę, informacija ir duomenys gali nutekėti, todėl reikalinga apsauga, kuri saugotų atmintinės turinį nuo neautorizuoto patekimo į jį, arba, kurią apeiti užtruktų tam tikras laiko tarpas, per kurį patys darbuotojai ar vadovai suprastų praradę atmintinę ir atliktų tam tikrus veiksmus, kad prarasta informacija ir duomenys nepakenktų jų vadovaujamos ar atstovaujamos įmonės tolimesnei veiklai.

Valstybinių įstaigų vadovai bei darbuotojai susiduria su ta pačia problema kaip ir įmonių vadovai ar jų darbuotojai, tik šiuo atveju grėsmė iškyla ne konkrečiai įmonei ar jų tinklui, bet valstybinei įstaigai ar net visai valstybei. Tikslai yra tie patys.

Menininkai susiduria su informacijos nutekėjimo problema, jei laikmena iš jų pavagiama arba jų pačių pametama, tada jų kūriniai (tarkime meninės fotografijos) gali patekti į kitas rankas ir būti panaudotos bei išplatintos prieš tai negavus jų autoriaus sutikimo. Šių vartotojų tikslas – turėti saugias atmintines, kurias praradus jų turinio negalėtų peržiūrėti kiti asmenys.

Norėdami nustatyti konkrečius poreikius vartotojus apklausėme. Apklausos rezultatai ir jų analizė pateikta 1 priede.



8 pav.: Vartotojų tikslų hierarchija

2.5. Panašių sistemų (Lietuvos ir tarptautiniu mastu) analizė

2.5.1. USB Secure® 1.6.6

„USB Secure“ – tai paprasta ir maža programėlė, kuri leidžia uždėti slaptažodį ant USB atmintinės ir kitas žmogus, nežinantis slaptažodžio, negalės prieiti prie atmintinėje saugomų failų. Programa veikia su daugeliu Windows OS versijų. Jos nereikia instaliuoti ir kitaip papildomai paleisti, bet programa neleidžia užkoduoti pasirinktų failų ir išvis nekoduoja failų – tiesiog uždeda slaptažodį ant disko ir taip padaro, ten patalpintus duomenis, neprieinamais, bet pasitelkus papildomą programinę įrangą ir turint daugiau IT žinių tokią apsaugą galima gana nesunkiai apeiti.

Pagrindiniai programos minusai:

- Programos kaina – viena licencija dabar kainuoja 29,95 JAV dolerius;
- Programa neužkoduoja USB atmintinėje esančių failų;
- Programa nepasiūlo jokių papildomų apsaugos ar administravimo funkcijų.

2.5.2. **MyUSBOnly 6.8**

„MyUSBOnly“ – programa, kuri instaliuojama į kiekvieną kompiuterį ir taip leidžia stebėti ir praneša vartotojui, kada prie kompiuterio prijungiamas neautorizuotas įrenginys. Taip pat suteikia „saugią“ (nuo kitų programų nepriklausomą) prieigą prie nešiojamosios laikmenos, kad duomenys galėtų būti perkelti pirmyn/atgal saugiai. Tačiau programos pagrindinis minusas – per didelė kaina (29.90 JAV doleriai) ir tai, kad programą būtina įdiegti į atskirą kompiuterį, kuriame norime, kad veiktų visas apsaugos mechanizmas, o už kiekvieną naują kompiuterį reikia pirkti naują licenciją. Tad šis sprendimas labiau tiktų didesnės kompanijos sistemos administratoriui sekti ar laikomasi saugumo politikos. Paprastam vartotojui šis sprendimas tiesiog neturėtų didesnės naudos.

2.5.3. **TrusCont Secure Flash Drive**

„TrusCont Secure Flash Drive“ (TSFD) – tai patobulintos USB atmintinės su papildomomis saugumo užtikrinimo savybėmis. Kompanija TrusCont prie šių atmintinių siūlo ir programinę įrangą TSFD Protection Toolkit, kurios pagalba galima greit ir efektyviai užtikrinti duomenų esančių atmintinėje saugumą. TSFD suteikia daugybė saugumą užtikrinančių funkcijų kaip:

- Failų apsaugą juos užšifruojant arba uždedant slaptažodžius ant atskirų failų;
- Visos atmintinės apsaugojimą slaptažodžiu;
- Failų laiko limitus;
- Skaitmeninių teisių apsaugą (DRM);
- Vartotojų, domenų ar kompiuterių parinkimas, kurie gali nuskaityti atmintinės turinį.

Sistema turi ir daugiau privalumų ir papildomų funkcijų, tačiau didžiausias šios sistemos trūkumas – tai kaina, nes ne tik, kad reikia pirkti atskiras atmintines, kurios kainuoja kelis kartus brangiau nei jų talpos ekvivalentai be šios apsaugos, bet ir reikia kas metus atnaujinti programinės įrangos licenciją kiekvienai USB atmintinei (po 30 JAV dolerių). To nepadarius visi failai patalpinti atmintinėje bus apsaugoti ir prieinami, tačiau nebebus galima pridėti naujų failų ir juos apsaugoti. Tad įvertinus šios sistemos kainą ir papildomas išlaidas, kurios susidarytų ją naudojant ilgiau nei metus, paprastam vartotojui ji nėra patraukli. Beje norint naudotis šia apsauga – reikia

būtinai suinstaliuoti TrusCont programinę įrangą į kompiuterį, kuriame ir bus naudojama saugioji atmintinė.

2.5.4. **TrueCrypt 7.1a**

Kaip vienas iš esamų sprendimų, yra programa „TrueCrypt“, ji naudoja įvairius duomenų užšifravimo algoritmus ir kitas priemones, kurios padeda apsaugoti atmintinę. Programa yra nemokama ir atvirojo kodo, todėl ją galima ne tik naudoti įmonėse, bet ir gauti programos išeities kodą. „TrueCrypt“ palaiko tokias funkcijas kaip:

- Gali užkoduoti pasirinktas ar visas atmintinės particijas, taip pat net gali užkoduoti skirsnį, kuriame suinstaliuota *Windows* operacinė sistema (tokiu atveju prašoma įvesti slaptažodį prieš pradedant krautis operacinei sistemai);
- Dėl savo saugumo parametrų, programa leidžia vartotojui tiesiog neigti, kad atmintinėje gali būti saugomi užkoduoti duomenys. Nežinant kas yra atmintinėje virtualiai neįmanoma nustatyti, kad kai kurie skirsniai gali būti paslėpti ar užkoduotos, nes paprastam vartotojui tie duomenys gali atrodyti kaip bet koks atsitiktinių duomenų kratinys (steganografijos principai).

2.6. Architektūros ir galimų įgyvendinimo priemonių variantų analizė

2.6.1. **AES (rijndael) šifravimo algoritmas**

AES (angl. *Advanced Encryption Standard*) naudojamas Rijndael šifravimas, kurį sukūrė du Belgijos kriptografai Joan Daemen ir Vincent Rijmen. Rjindael šifravimo algoritmo vardo kilmė yra iš jo kūrėjų pavardžių. Kriptografijoje AES yra simetrinio rakto šifravimo standartas, kurį naudojo JAV vyriausybė. Standartas susideda iš trijų blokinių šifrų AES-128, AES-192 ir AES-256, kurie yra paimti iš daugelio įeinančių į Rijndael. Kiekvienas šių šifrų turi 128 bitų bloko dydį, su 128, 192 ir 256 bitų rakto dydžiais. Dabar AES šifrai yra plačiai išnagrinėti ir naudojami visame pasaulyje.

Atitikmenys

Žemiau yra pateikti kai kurie žodžių paaiškinimai, kurių atitikmenys nurodyti (6) lentelėje.

6 lentelė

Žodis	Paaiškinimas
Šifravimas	Serija transformacijų, kurios *naudojant šifravimo raktą, paverčia paprastą tekstą į* šifruotą tekstą.
Šifravimo raktas	Slaptas, kriptografinis raktas, kuris naudojamas raktų generavimo veiksmuose, kad sukurti eilę raktų Gali būti įsivaizduojama kaip stačiakampė baitų lentelė, kurioje yra 4 eilutės ir ***Nk*** stulpelių.
Raktų generavimas	Tai veiksmai, kurie naudojami eilei raktų iš šifravimo rakto sukurti.
Raktų eilė	Yra reikšmės gautos iš šifravimo rakto, naudojant raktų generavimo veiksmus. Jos yra naudojamos šifravimui ir dešifravimui.
Šifruotas tekstas	Duomenys gaunami kaip išėjimas iš šifravimo, naudojami kaip įėjimas į dešifravimą.
Dešifravimas	Serija transformacijų, kurios naudojant šifravimo raktą, paverčia šifruotą tekstą į paprastą tekstą.
Būsena (State)	Tarpinis šifravimo rezultatas, kuris gali būti įsivaizduojamas kaip stačiakampė baitų lentelė, susidedanti iš 4 eilučių ir ***Nb*** stulpelių.

Algoritmo parametrai, simboliai ir funkcijos

Algoritmo parametrai, simboliai ir funkcijos aprašomos (7) lentelėje:

7 lentelė

Algoritmo parametrai, simboliai ir funkcijos	Paaiškinimas
AddRoundKey()	Transformacija šifravime arba dešifravime kai į būseną įvedama raktų eilė, naudojant XOR operaciją. Raktų eilės ilgis yra toks pat kaip būsenos (pvz.: jei ***Nb***=4, tai tada raktų eilė yra lygi 128 bitams / 16 baitų).
InvMixColumns()	Transformacija, naudojama dešifravime, ji yra invertuota *MixColumns()*
InvShiftRows()	Transformacija, naudojama dešifravime, ji yra invertuota *ShiftRows()*
InvSubBytes()	Transformacija, naudojama

	dešifravime, ji yra invertuota *SubBytes()*
K	Šifravimo raktas
MixColumns()	Transformacija, naudojama šifravime, kuri paima visus būsenoje esančius stulpelius ir sumaišo jų duomenis, nepriklausomai vienas nuo kito. Taip sukuriami nauji stulpeliai
Nb	Būseną sudarančių stulpelių skaičius. Šiame standarte ***Nb*** = 4
Nk	32-bitų žodžių skaičius, kuris nurodo kiek turėtų būti žodžių sudarančių šifravimo raktą. Šiame standarte ***Nk*** = 4,6 ar 8.
RotWord()	Funkcija, naudojama raktų generavime, kuri paima 4 bitų žodį ir atlieka ciklinį perstatymą.
ShiftRows()	Transformacija, naudojama šifravime, kuri paveikia būseną, ciklišKai sukeisdama jos paskutinius 3 stulpelius.
SubBytes()	Šifravime naudojama transformacija, kuri paveikia būseną naudodama ne linijinį bitų perstatymą (S-Box), kuris nepriklausomai operuoja kiekvieną būsenos bitą.
SubWord()	Funkcija naudojama raktų generavime, kuri paima 4 bitų žodį ir pritaiko S-Box kiekvienam iš 4 bitų, taip gaunamas išėjimo žodis (output word).
XOR	Exclusive-OR operacija.
$\oplus$	Exclusive-OR operacija.
$\otimes$	Dviejų polinomų daugyba.
$\bullet$	Baigtinio lauko daugyba.

Įėjimai ir išėjimai

Įėjimai ir išėjimai AES algoritme kiekvienas susideda iš 128 bitų sekų. Šios sekos kartais bus vadinamos blokais, bitų skaičius, iš kurių jos susideda bus vadinamas kaip jų ilgis. Šifro raktas AES algoritme yra 128, 192 ir 256 bitų seka. Kitokie įėjimo, išėjimo ir šifravimo rakto standartai šiame standarte yra neleistini. Bitai, esantys šių sekų viduje bus numeruojami pradedant 0 ir baigiant vienu mažiau nei sekos ilgis. Skaičius *i* yra naudojamas kaip indeksas ir bus šiose ribose $0 \le i < 128$, $0 \le i < 192$ or $0 \le i < 256$, priklausomai nuo bloko ir rakto ilgio.

Baitai

Pagrindinis vienetas, dominuojantis AES algoritme yra baitas, tai 8 bitų seka, laikoma kaip viena esybė. Įėjimo, išėjimo ir šifravimo rakto sekos yra apdorojamos kaip baitų eilės, kurios suformuojamos padalinant šias sekas po 8 bitus, kad suformuoti baitų eiles. Įėjimams, išėjimams ar šifravimo raktui, pažymėtam a, baitai pabaigoje bus perdaromi naudojant vieną iš dviejų formų a_n arba $a[n]$, kur n bus vienoje iš žemiau pateiktų ribų:

Rakto ilgis = 128 bitai, $0 \le n < 16$; Bloko ilgis = 128 bitai, $0 \le n < 16$

Rakto ilgis = 192 bitai, $0 \le n < 24$

Rakto ilgis = 256 bitai, $0 \le n < 32$

Visos bitų reikšmės AES algoritme bus matomos kaip savo individualių bitų (0 ir 1) konkatenacija tarp $\{b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0\}$. Šie bitai yra interpretuojami kaip baigtinio lauko elementai, naudojant polinominę reprezentaciją:

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

Pavyzdžiui, $\{01100011\}$ identifikuoja specifinį baigtinio lauko elementą $x^6 + x^5 + x + 1$.

Taip pat patogu paversti baitų reikšmes naudojant dešimtainę sistemą, kur kiekvieni 4 bitai reiškia vieną simbolį, kaip parodyta lentelėse:

Bit Pattern	Character
0000	0
0001	1
0010	2
0011	3

Bit Pattern	Character
0100	4
0101	5
0110	6
0111	7

Bit Pattern	Character
1000	8
1001	9
1010	a
1011	b

Bit Pattern	Character
1100	c
1101	d
1110	e
1111	f

9 pav.

Baitų eilės

Baitų eilės pateikiamos tokia forma:

$a_0 a_1 a_2 ... a_{15}$

Baitai ir bitų išdėstymas baituose yra paskirstomi į128 bitų įėjimų seką:

$input_0\ input_1\ input_2\ \ldots\ input_{126}\ input_{127}$

Iš to gaunama:

$$a_0 = \{input_0, input_1, \ldots, input_7\};$$
$$a_1 = \{input_8, input_9, \ldots, input_{15}\};$$
$$\vdots$$
$$a_{15} = \{input_{120}, input_{121}, \ldots, input_{127}\}$$

Šablonas gali būti išplėstas į ilgesnes sekas (pvz.: 192 ar 256 bitų raktams), todėl bendru atveju:

$$a_n = \{input_{8n}, input_{8n+1}, \ldots, input_{8n+7}\}.$$

Būsena (*State*)

AES algoritmo operacijos yra atliekamos dvimatėje baitų lentelėje, vadina Būsena (angl. *state*). Būsena susideda iš 4 baitų eilučių, kur kiekvienoje yra ***Nb*** baitai, kur ***Nb*** yra bloko ilgis padalintas iš 32. Būsenos lentelėje, kurioje pagrindinis simbolis yra s, kiekvienas individualus baitas turi po du indeksus su eilutės numeriu r, tarpe $0 \le r < 4$ ir su stulpelio numeriu c, tarpe $0 \le c < Nb$. Tai leidžia kiekvienam būsenos baitui būti atpažintam kaip $s_{r,c}$ arba $s[r,c]$. Pradžioje šifravimo, įėjimas – bitų eilė $in_0, in_1, \ldots in_{15}$, nukopijuojama į būsenos lentelę, kaip parodyta (10) paveiksle. Būsenoje vykdoma šifravimo operacija, po kurios galutinė reikšmė nukopijuojama į išėjimą, baitų eilę $out_0, out_1, \ldots out_{15}$.

10 pav.

Pradžioje šifravimo įėjimo eilė *in* nukopijuojama į būseną pagal schemą:

$$s[r, c] = in[r + 4c] \qquad \text{for } 0 \le r < 4 \text{ and } 0 \le c < Nb$$

Šifravimo pabaigoje, kai iš būsenos reikšmės nukopijuojamos į išėjimo eilę, tai daroma pagal:

$$out[r + 4c] = s[r, c] \qquad \text{for } 0 \le r < 4 \text{ and } 0 \le c < Nb$$

Matematiniai veiksmai

Visi baitai AES algoritme yra interpretuojami kaip baigtinio lauko elementai. Baigtinio lauko elementai gali būti sudedami, padauginami, tačiau šios operacijos skiriasi nuo tų, kurios naudojamos su skaičiais. Toliau pateikiami pagrindiniai matematiniai koncertai.

Pridėjimas

Baigtinio lauko elemento sudėtis pasiekiama sudedant laipsnių koeficientus polinomuose dviem elementams. Sudedama pasinaudojus XOR funkcija, kuri dar žymima $\oplus$ ženklu.

Sudedant du baitus $\{a_7a_6a_5a_4a_3a_2a_1a_0\}$ ir $\{b_7b_6b_5b_4b_3b_2b_1b_0\}$, suma būtų $\{c_7c_6c_5c_4c_3c_2c_1c_0\}$, kur kiekvienas $c_i = a_i \oplus b_i$.

Pavyzdžiui, šios išraiškos yra viena kitai ekvivalenčios:

$(x^6 + x^4 + x^2 + x + 1) + (x^7 + x + 1) = x^7 + x^6 + x^4 + x^2$ Polinominė išraiška

$\{01010111\} \oplus \{10000011\} = \{11010100\}$ Dvejetainė išraiška

$\{57\} \oplus \{83\} = \{d4\}$ Dešimtainė išraiška

Daugyba

$\{57\} \bullet \{83\} = \{c1\}$ nes

$$(x^6 + x^4 + x^2 + x + 1)(x^7 + x + 1) = x^{13} + x^{11} + x^9 + x^8 + x^7 + x^7 + x^5 + x^3 + x^2 + x + x^6 + x^4 + x^2 + x + 1 = x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1$$

ir

$$x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1 \bmod (x^8 + x^4 + x^3 + x + 1) = x^7 + x^6 + 1$$

Modulio funkcija užtikrina, kad rezultatas bus dvejetainis daugianaris, su laipsniu, mažesniu nei 8 ir todėl gali būti naudojami baitai. Ne taip kaip sudėtyje, čia nėra paprastos operacijos baitų lygyje, kuri suderinama su dauginimu.

Daugyba iš x

Padauginus $b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0$ iš polinominio x, rezultatas būtų $b_7x^8 + b_6x^7 + b_5x^6 + b_4x^5 + b_3x^4 + b_2x^3 + b_1x^2 + b_0x$.

Rezultatas $x \bullet b(x)$ gaunamas redukuojant, panaudojus prieš tai gautą rezultatą su funkcija mod. Jeigu $b_7 = 0$, rezultatas jau yra redukuotoje formoje. Jeigu $b_7 = 1$, redukavimas pasiekiamas atimant (panaudojant XOR) polinominei $m(x)$. Iš to seka, kad daugyba iš x gali būti įgyvendinama baitų lygyje, du kartus panaudojant funkciją XOR. Šiai operacijai baitais, naudojama funkcija `xtime`. Daugyba iš x su didesniais laipsniais, gali būti atliekama pakartotinai naudojant funkciją `xtime`.

Pvz.: $\{57\} \bullet \{13\} = \{fe\}$ nes:

$\{57\} \bullet \{02\} = xtime(\{57\}) = \{ae\}$

$\{57\} \bullet \{04\} = xtime(\{ae\}) = \{47\}$

$\{57\} \bullet \{08\} = xtime(\{47\}) = \{8e\}$

$\{57\} \bullet \{10\} = xtime(\{8e\}) = \{07\}$

todėl,

$$\begin{aligned}\{57\} \bullet \{13\} &= \{57\} \bullet (\{01\} \oplus \{02\} \oplus \{10\}) \\ &= \{57\} \oplus \{ae\} \oplus \{07\} \\ &= \{fe\}.\end{aligned}$$

AES privalumai

Kaip ir kiekvienas blokinis, algoritmas AES šifruoja 128-bitų dydžio blokus – tai net du kartus didesnius nei DES. DES buvo sukurtas, kad optimaliai išnaudotų aparatinę įrangą, o AES yra universalesnis ir yra efektyvus įvairiausiose realizacijose. Kiekvienas algoritmo žingsnis susideda iš operacijų, kurios gali būti atliekamos vienu metu, lygiagrečiai, o tai leidžia pasiekti didelį algoritmo greitį panaudojant kelis procesoriaus branduolius ar pačius procesorius skaičiavimams ir transformacijoms atlikti. Paprasta algoritmo struktūra leidžia išvengti klaidų.

Sunkumai įgyvendinant

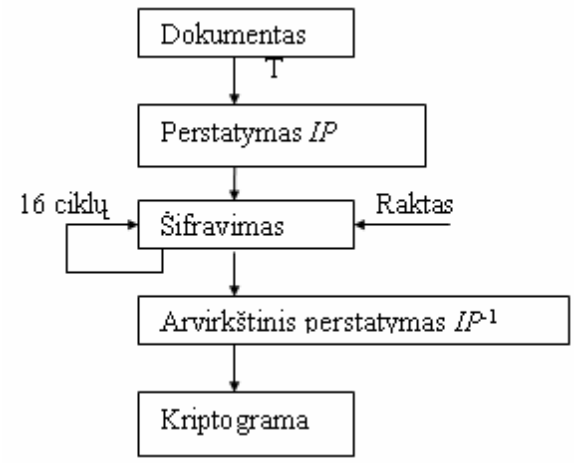
Realizavus AES algoritmą jis turi palaikyti bent vieną iš trijų toliau pateiktų raktų ilgių. Ir tai būtų: 128, 192 ar 256 bitų (t.y.: ***Nk*** = 4, 6 ar 8). Realizacijos pasirinktinai gali palaikyti du arba net visus tris raktų ilgius, kurie gali padidinti programos lankstumą ir suderinamumą. Taip pat algoritmo specifikacijoje nėra apibrėžta kas yra

silpnas ar pusiau silpnas raktas, todėl nėra jokių apribojimų renkantis norimą raktą.[13]

2.6.2. DES šifravimo standartas

DES (angl. *Data Encryption Standard*) – simetrinis blokinis algoritmas, kurio šifruojamų blokų ilgis yra 64 bitai naudojant 64 bitų ilgio raktą. DES duomenų šifravimo standartu JAV paskelbė 1977 metais JAV Nacionalinis standartų biuras. 1972 – 1975 metais IBM ir NSA specialistų sukurtas DES algoritmas turėjo būti skirtas svarbiai informacijai apsaugoti valstybinėse ir komercinėse JAV organizacijose.

DES algoritme dešifravimas vyksta atvirkščia tvarka nei užšifravimas, o algoritmo saugumas priklauso nuo pasirinkto rakto. Algoritmas turi 64 silpnuosius raktus, kuriais galima pasinaudoti norint pažeisti DES apsaugos mechanizmą. DES algoritmas susideda iš 16 etapų: vienam dokumento blokui yra taikoma ta pati metodų kombinacija 16 kartų (11 pav.).



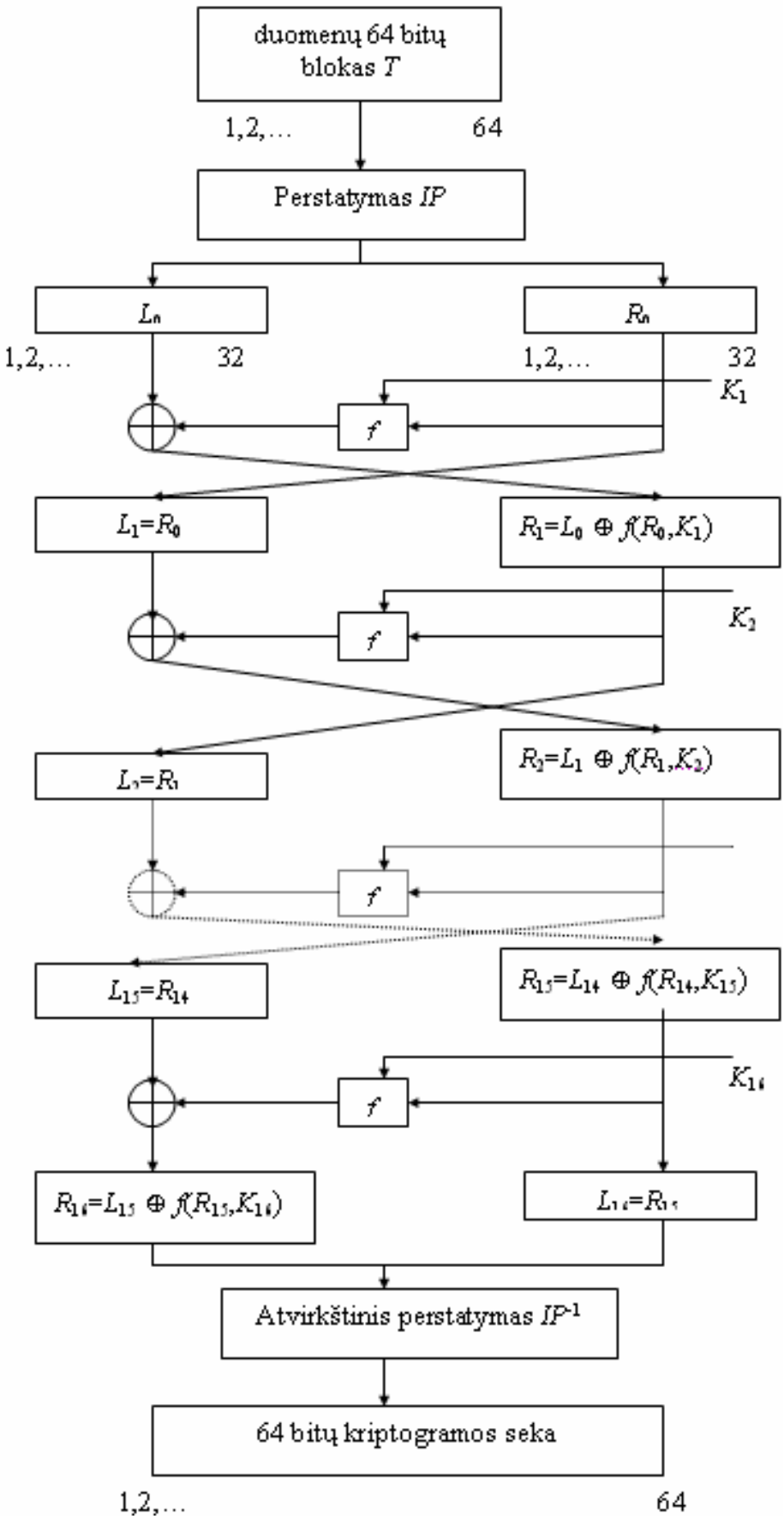
11 pav.: DES algoritmo veikimo schema

Užšifravimas DES algoritme susideda iš 64 bitų dokumento bloko *T* perstatymo (*Perstatymas IP*), šešiolikos šifravimo etapų ir tada gauto bloko bitų *atvirkštinio perstatymo* IP^{-1}.

Kitame puslapyje pateikta DES algoritmo struktūra. Joje naudojami tokie žymėjimai:

- L – kairioji (ang. *left*) bitų seka, R – dešinioji (angl. *right*) bitų seka;
- LR – tai tokia seka, kurioje sekos R bitai eina po sekos L bitų. Jos ilgis yra lygus sekų L ir R ilgių sumai;

- $\oplus$ – bitų sudėtis moduliu 2.



12 pav.: DES algoritmo struktūra

Perskirstymo ir atvirkštinė perskirstymo matricos (8 ir 9 lentelės), naudojamos DES algoritme, yra standartinės ir negali būti keičiamos. Matricų kodai yra parinkti taip, kad būtų kuo sunkiau dešifruoti parenkant raktą.

9 lentelė: Perstatymo matrica IP

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

8 lentelė: Atvirkštinio perstatymo matrica IP^{-1}

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

DES šifravimas:

Sakykime, jog turime kokio nors dokumento 64 bitų bloką *T*. Iš pradžių šis blokas pertvarkomas naudojant pradinio perstatymo matricą *IP* (8 lentelė). Įėjimo bloko *T* 64 bitai perstatomi tokia tvarka kaip pavaizduota perstatymo matricoje, t.y., bloko *T* 58-asis bitas tampa pirmuoju bitu, 50-asis – antruoju bitu ir t.t. Po šių veiksmų gaunamas blokas T_0. Šį perstatymą galime aprašyti išraiška: $T_0 = IP(T)$. Tada gautoji bitų seka T_0 padalinama į dvi 32 bitų sekas: L_0 – kairiąją seką ir R_0 – dešiniąją seką. Padalinus T_0 į dvi sekas atliekamas duomenų užšifravimas, kuris būna sudarytas iš 16 pasikartojančių ciklų.

$T_i = L_iR_i$, čia $L_i=t_1, t_2 \dots t_{32}$ (pirmi 32 bitai); $R_i=t_{33}, t_{34} \dots t_{64}$ (paskutiniai 32 bitai).

Tada atliekama 16 šifravimo etapų:

$L_i = R_{i-1}$, kai $i = 1,2,....,16$.

$R_i = L_{i-1} + f(R_{i-1}, k_i)$, kai $i = 1,2,....,16$.

Čia funkcija f - tai šifravimo funkcija. Ji naudoja du argumentus: seką R_{i-1}, kuri būna gaunama ankstesniame cikle, ir 48 bitų subraktą k_i, kuris yra suformuojamas iš pradinio 64 bitų rakto *K*. Iš viso iš pradinio rato *K* sukonstruojama 16 dalinių raktų $k_1, k_2 \dots, k_{16}$, kurie naudojami kiekviename šifravimo cikle.

Paskutiniajame cikle gaunamos sekos R_{16} ir L_{16}, kurios sujungiamos atgal į 64 bitų seką. Baigus šifravimą, naudojant atvirkštinio perstatymo matricą IP_{-1}, atstatomos bitų pozicijos.

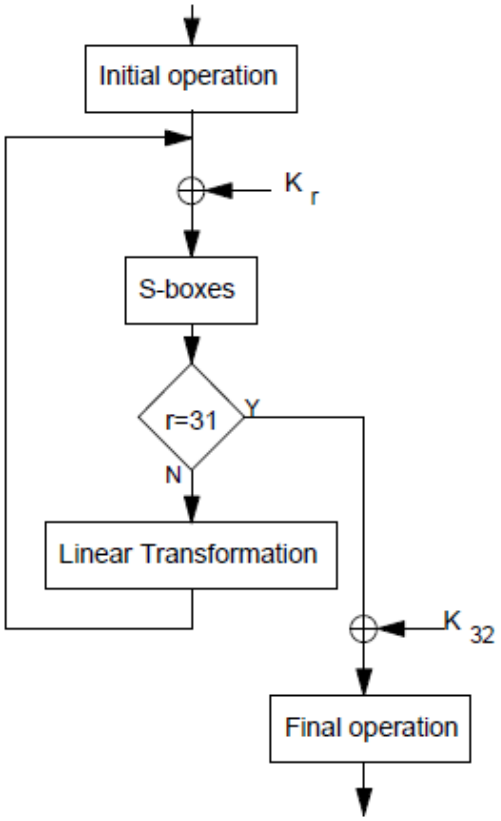
DES algoritmo trūkumai

Kaip ir visi šifravimo algoritmai, taip ir DES nėra atsparus jėgos atakoms, todėl anksčiau ar vėliau pavyksta rasti šifravimo raktą. Ypač todėl, kad DES realus rakto ilgis yra 56 bitai ir šiais laikais jis tikrai per trumpas. Taip pat DES turi 64 raktus, kurie yra silpni ir neužtikrina optimalaus saugumo:

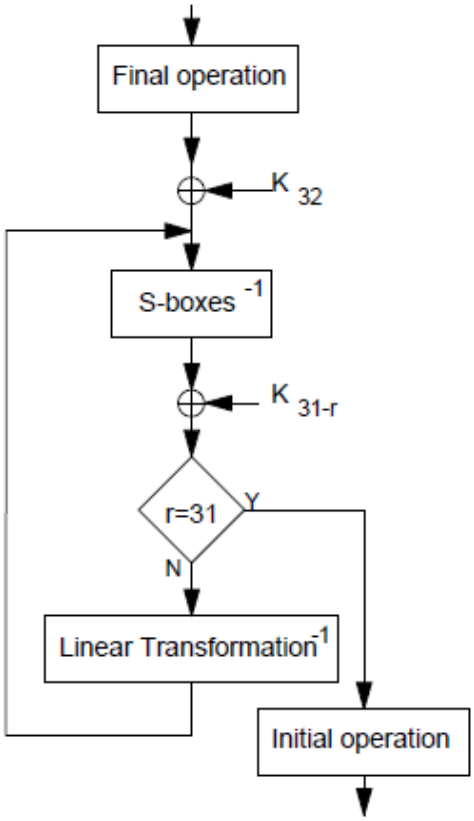
- *4 iš galimų 256 raktų yra silpni*. Kadangi algoritmo subraktai (k_i) gaunami iš pradinio rakto, skaidant jį į dvi dalis, o toliau, kiekvienos dalies bitus ciklišKai perstumiant per 1 arba 2 bitus, tai jei visi rakto bitai yra 0 arba 1, arba vienos dalies bitai – 0, o kitos – 1, tai kiekvienam etapui bus naudojamas tas pats raktas.
- *Šešios raktų poros yra pusiau-silpnieji raktai*. Informacija užšifruota vieno poros rakto pagalba, bus dešifruota su kitu poros raktu, tad vietoj 16 skirtingų subraktų algoritmas generuoja tik du skirtingus subraktus.
- Likę 48 raktai, generuoja tik 4 skirtingus subraktus, iš kurių kiekvienas naudojamas 4 kartus, tad šie *raktai skaitomi nepatikimais*.[14]

2.6.3. Serpent

Serpent – tai dar vienas blokinis simetrinis šifravimo algoritmas sukurtas R. Anderson‘as, E. Biham‘as ir L. Knudsen‘as. Šis algoritmas buvo finalistas AES šifravimo algoritmų konkurse ir užėmė antrąją vietą po Rijndael.
Serpent naudojamas šifravimo bloko dydis - 128 bitai. Palaikomas rako ilgis - 128, 192 arba 256 bitai. Šifravimas vyksta 32 iteracijų blokais sudarytais iš 32 žodžių. Kiekviena iteracija paraleliai operuoja vieną iš aštuonių 4x4 bitų pakeitimo lentelę 32 kartus. Žemiau pateiktos schemos vaizduoja šifravimo ir dešifravimo procesą naudojamą šiame algoritme:



13 pav.: Užšifravimo operacija



14 pav.: Dešifravimo operacija

Serpent algoritmo veikimą galima aprašyti šiomis lygtimis:

$$\hat{B}_0 := IP(P)$$
$$\hat{B}_{i+1} := R_i(\hat{B}_i)$$
$$C := FP(\hat{B}_{32}),$$ kur

$$R_i(X) = L(\hat{\mathcal{S}}_i(X \oplus \hat{K}_i)) \qquad i = 0, \ldots, 30$$
$$R_i(X) = \hat{\mathcal{S}}_i(X \oplus \hat{K}_i) \oplus \hat{K}_{32} \quad i = 31$$

Čia L – linijinė transformacija.

Serpent buvo vertinamas kaip naudojantis labiau konservatyvų požiūrį į apsaugą, lyginant su kitais AES konkurso finalistais. Buvo priimta, kad 16 iteracijų yra pakankamas šifravimo kiekis nuo žinomų atakų, tačiau Serpent jų naudojo 32, nes buvo tikimasi, kad ateityje jų reikės daugiau. Dėl to algoritmas yra tokio pat greitumo kaip DES, bet daug saugesnis nei 3DES.

Serpent yra viešas ir atviras algoritmas, kuris nebuvo patentuotas. Todėl nėra jokių apribojimų norint jį naudoti savo ar komercinėms reikmėms.

Serpent algoritmo trūkumai ir pliusai

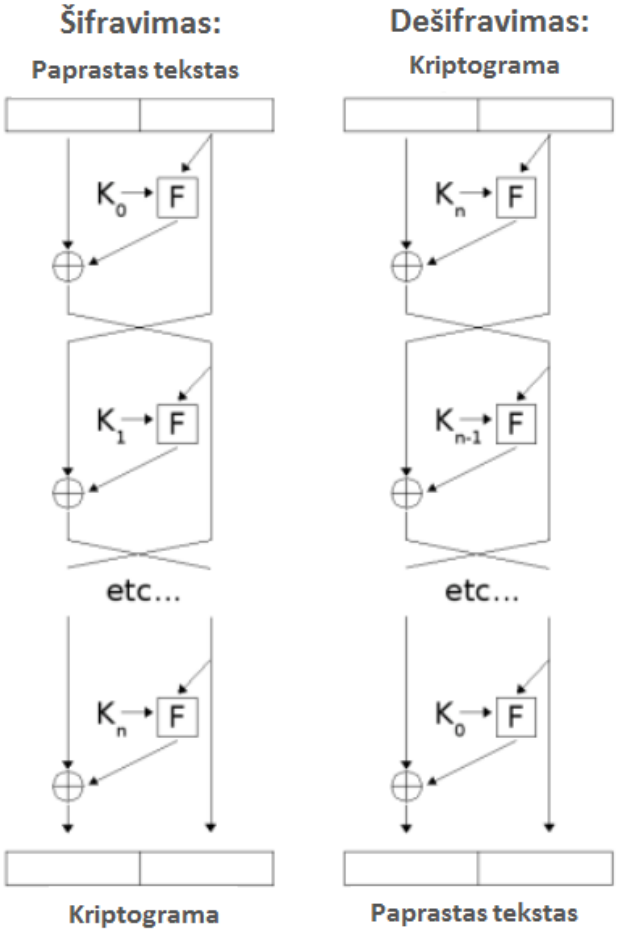
Serpent nebuvo pasirinktas kaip AES konkurso nugalėtoju, nes Rijndael buvo greitesnis šifruojant mažesnius duomenų blokus ir maišas. Taip pat jis buvo

greitesnis ir su didžiąja dalimi kitos informacijos. Serpent kartais būdavo net 2 ar 6 kartus lėtesnis nei Rijndael. Tačiau Serpent vis dar lieka vienu iš saugiausių algoritmų, jei jums svarbiau yra saugumas, nei algoritmo veikimo greitis.[15,16]

2.6.4. **CAST-128**

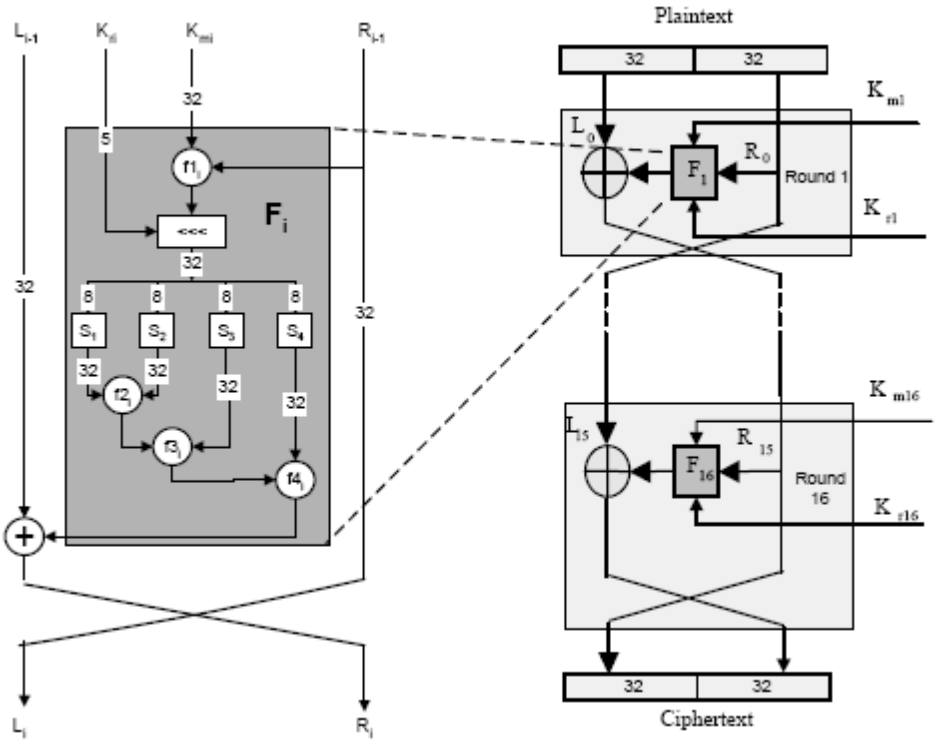
CAST-128 – tai blokinis simetrinis algoritmas, kuris buvo sukurtas C. Adams'o ir S. Tavares'o 1996 metais. Jis standartiškai naudojamas GPG (angl. *GNU Privacy Guard*) ir PGP (angl. *Pretty Good Privacy*) apsaugos sistemose. Taip pat jį naudoja nacionalinė Kanados kriptografijos agentūra. Vėliau, naudojant CAST-128 pagrindus buvo sukurtas CAST-256 algoritmas.

CAST-128 naudojama standartinė Feistelio struktūra (pav. 15) susidedanti iš 12 ar 16 iteracijų simetrinių 64-bitų dydžio blokų ir nuo 40 iki 128 bitų rakto dydžiu. Raktą galima didinti kas 8 bitus. Visos 16 iteracijų naudojamos tik tada, kai rakto dydis yra didesnis nei 80 bitų. Vėlesnė CAST versija CAST-256 turėjo bloko dydį iš 256 bitų.



15 pav.: CAST-128 šifravimo ir dešifravimo schema

Funkcijoje F (pav. 16) naudojamas raktas K kiekvienai iš trijų interacijų išskaidomas į K_m ir K_r. Kiekvienoje iteracijoje naudojamos keturios pakeitimo lentelės – S_1, S_2, S_3 ir S_4. Pakeitus bitus, su jais atliekamos trys, atimties, sudėties ir XOR, operacijos.



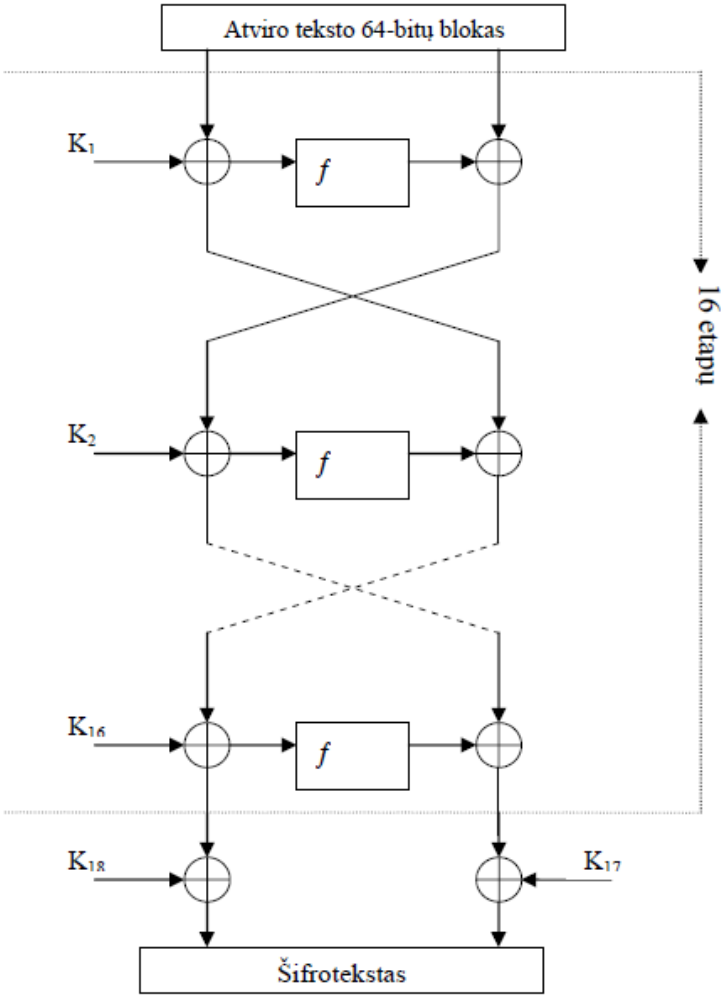
16 pav.: CAST-128 šifravimo schema

CAST-128 pažeidžiamumas

Naudojant CAST-128 reikėtų pasisaugoti trumpų raktų naudojimo. Jei rakto ilgis būna 56 bitai ar net trumpesnis, tai panaudojus jėgos ataką pakankamai greitai galima gauti nulaužti bet kokį šifravimo algoritmą. Todėl šifruojant informaciją CASP-128 algoritmu patartina naudoti ne trumpesnius nei 80 bitų raktus.[17,18]

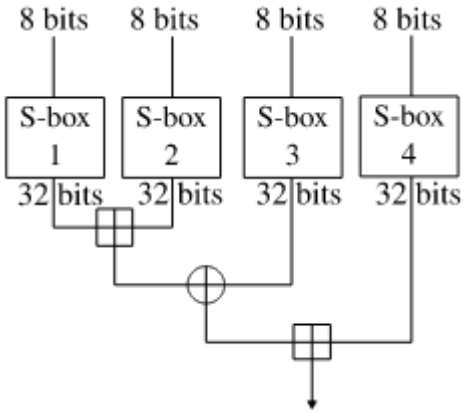
2.6.5. **Blowfish**

Blowfish - tai blokinis simetrinis šifravimo algoritmas, sukurtas 1994 metais Bruce'o Schneier'io. Jis buvo sukurtas kaip neužpatentuota DES algoritmo alternatyva. Vėliau šio algoritmo pagrindu buvo sukurtas Twofish algoritmas. Blowfish algoritmas šifruoja atvirą tekstą 64 bitų blokais. Algoritmo rakto ilgis gali kisti nuo 32 iki 448 bitų ilgio. Diagrama apačioje rodo Blowfish šifravimo algoritmo veikimą.



17 pav.: Blowfish algoritmo schema

Pirmiausiai atliekama pirmojo 32-bitų bloko ir įvesto rakto masyvo *K[1] XOR* operacija. Tuomet atliekama *f* pakeitimo funkcija. Gavus rezultatą, XOR funkcija atliekama su antruoju 32-bitų bloku.



18 pav.: Funkcijos *f* struktūrinė schema

Kiekviena iš dviejų linijų turi po 32-bitus – tai 64-bitų blokas išskaidytas į dvi dalis po 32-bitus. Algoritmas turi du masyvus: 18 įrašų P masyvą ir keturias 256 įrašų pakeitimo lenteles (angl. *S-Boxes*), kurios naudojamos pakeitimo funkcijai *f*. Vykdant pakeitimo funkciją, pakeitimo lentelės įėjime priima 8 bitus, o išėjime grąžina 32 bitus. *f* funkcijos veikimą apibūdina diagrama kairėje.

Dešifravimas vyksta taip pat kaip ir šifravimas, skirtumas tik tas, kad dešifruojant *K* masyvo įrašai panaudojami atvirkščia tvarka, t.y. nuo 18 iki 1. Funkciją *f* galima aprašyti taip:

$$f(x) = ((S_{1,a} + S_{2,b}) \bmod 2^{32}) \oplus S_{3,c}) + S_{4,d} \bmod 2^{32}.$$

Čia $\oplus$ – sudėtis *mod* 2.

Blowfish algoritmo privalumai ir trūkumai

Iš šio algoritmų trūkumų galima paminėti šiuos:

- Blowfish algoritme rakto praplėtimo ir užšifravimo operacijos nėra lygiagrečios. Jos atliekamos atskirai, todėl kenčia algoritmo našumas;
- Algoritmas nėra tinkamas, kai reikia dažnai keisti raktus. Vienas iš didžiausių Blowfish algoritmo privalumų - didelis informacijos šifravimo greitis, kuris pasireiškia tik tada, kai vienu raktu šifruojama daug informacijos. Todėl, jeigu raktas yra keičiamas po kiekvieno bloko užšifravimo, algoritmo efektyvumas ir greitis labai sumažėja;
- Mažas šifruojamo bloko ilgis.

Algoritmo privalumai:

- Paprastas;
- Didelis šifravimo greitis, jei naudojamas vienas raktas;
- Atsparus atakoms, jei naudojamas 16 etapų šifravimas.[19, 20]

2.6.6. MD5 algoritmo analizė

Apibrėžimai

Žodis yra 32 bitų grupė.

Baitas yra 8 bitų grupė.

'b' reprezentuoja įvesties ilgį bitais.

'|' = OR. '&' = AND. '^' = XOR. '~' = NOT.

X <<< s, tai bitų pasislinkimas ratu X per s bitų pozicijų.

Įėjimo duomenys

Duomenys, kuriuos apdoroja MD5 algoritmas gali būti bet koks skaičius bitų. Failas nebūtinai turi užimti minimaliai 8 bitus – gali būti ir 0 bitų ilgio.

Pirmas žingsnis: duomenų išplėtimas

Pradžioje įvesties duomenys išplečiami taip, kad iki 512 bitų ribos trūktų 64 bitų. Tai reiškia, kad bitų skaičius turėtų būti 448. Pradžioje duomenys išplečiami juos paverčiant dvejetainiu kodu, o po to pridedant tiek nulių, kiek reikia, kad būtų užbaigtas išplėtimas ir duomenų ilgis būtų 512 be 64 bitų.

Antras žingsnis: ilgio papildymas

Žinutės ‚b' ilgis yra papildomas pirmame žingsnyje. Tai yra įvesties bitų skaičius prieš išplėtimą. Jei ilgis yra didesnis už 64 bitus, tai tik žemutiniai bitų sluoksniai yra panaudojami. Dabar duomenų ilgis bus lygus 512. Duomenis dabar galima suskaldyti į lygius bitų žodžius po 16 ar 32 bitus. Šie yra sužymimi kaip M[0 .. N-1], kai N yra 16 kartotinis.

Trečias žingsnis: inicijuoti žinutės apdorojimo buferį

Yra naudojami keturi vieno žodžio buferiai. Jie inicijuojami pradžioje naudojant žemesnio lygio bitus. Žemiau pateiktas pavyzdys hex formoje. Šios reikšmės užkraunamos kaip numatytas iniciacijos vektorius. Pastaba: C ir D tai tik B ir A atvirkščiai.

```
A: 01 23 45 67
B: 89 ab cd ef
C: fe dc ba 98
D: 76 54 32 10
```

Ketvirtas žingsnis: apdoroti įvesties duomenis

Kad būtų galima pradėti šią fazę, turi būti nustatytos keturios funkcijos. Kiekviena iš šių funkcijų paima tris 32 bitų žodžius ir išeityje gaunamas vienas 32 bitų žodis.

F(A,B,C) = (A & B) | (~A & C)

G(A,B,C) = (A & C) | (B & ~C)

H(A,B,C) = A ^ B ^ C

I(A,B,C) = B ^ (A | ~C)

F veikia kaip už pozicinis sąlygos operatorius, taip jog jei (if) A, tada (then) B, priešingu atveju (else) C.

G irgi labai panašus, vos ne identiškas. If C, then A, else B.

H yra tiesiog lygybės funkcija susidedanti iš 3 įvesčių.

Visose keturiose funkcijose kiekvienas bitas yra nešališkas ir nepriklausomas. Taip jo negali veikti joks kitas bitas iš šalies apart jo dublikato iš kitų įvesčių.

Toliau yra sudaroma lentelė iš 64 elementų. Jų reikšmės yra: 4294967296 kart abs(sin(i)), kur i yra elementų skaičius radianais. Lentelė sunumeruojama T[1 .. 64]. Taigi, T[1] = 0xD76AA478. Šios reikėmės gali būti apskaičiuotos kiekvieną ciklą, bet kadangi jos niekada nesikeičia, tai efektyviausia jas tiesiog įrašyti jas į kodą.

Dabar mums reikia apdoroti kiekvieną 16 žodžių bloką.

```
For i = 0 .. N/16-1
```

Taip pat reikia užkrauti visas žinutes į laikiną buferį.

```
For j = 0 .. 15
X[j] = M[i*(j+16)]
```

Toliau mums reikia sukurti duomenų iš A,B,C,D kopiją, nes mums šių dar reikės vėliau.

```
AA = A, BB = B, CC = C, DD = D
```

Dabar galima atlikti 4 lygių skaičiavimus keisdami kiekvieną kartą po funkciją. Taigi pirmą kartą naudojame funkciją F, antrą – funkciją G ir t.t. Bet kadangi jos tokios skirtingos tai geriausia jas įrašyti į kodą statiškai, nei naudoti ciklus.

Galiausiai mes panaudojame paprastą matematinę sudėtį ir pridedame pradines žodžių reikšmes prie galutinių.

A = A + AA

B = B + BB

C = C + CC

D = D + DD

Penktas žingsnis: išeiga

Prieš pradedant šį žingsnį mums lieka 4 žodžių išeiga. A yra žemesnės eilės žodis, o D - aukščiausios.

Dabar juos galime tiesiog atspausdinti.

T reikšmės lentelė

Žemiau pateikta lentelė talpina numerius, kurie yra naudojami ketvirtame žingsnyje kaip T reikšmės. Šių reikšmių formulė būtų: 4294967296 kart abs(sin(i)), kur i yra elementų skaičius radianais.

```
D76AA478 E8C7B756 242070DB C1BDCEEE F57C0FAF 4787C62A A8304613 FD469501
698098D8 8B44F7AF FFFF5BB1 895CD7BE 6B901122 FD987193 A679438E 49B40821
F61E2562 C040B340 265E5A51 E9B6C7AA D62F105D 02441453 D8A1E681 E7D3FBC8
21E1CDE6 C33707D6 F4D50D87 455A14ED A9E3E905 FCEFA3F8 676F02D9 8D2A4C8A
FFFA3942 8771F681 6D9D6122 FDE5380C A4BEEA44 4BDECFA9 F6BB4B60 BEBFBC70
289B7EC6 EAA127FA D4EF3085 04881D05 D9D4D039 E6DB99E5 1FA27CF8 C4AC5665
F4292244 432AFF97 AB9423A7 FC93A039 655B59C3 8F0CCC92 FFEFF47D 85845DD1
6FA87E4F FE2CE6E0 A3014314 4E0811A1 F7537E82 BD3AF235 2AD7D2BB EB86D391
```

Techninis pažeidžiamumas

Kol kas yra žinomas tik vienas paprastas MD5 algoritmo pažeidžiamumas. Jo pagrindas – galimybė pakeisti vos kelis atsitiktinių duomenų rinkinių bitus. Du duomenų rinkiniai, kuriais galima išgauti tą patį MD5 hash‘ą vadinami „kolizija“. Tokios kolizijos gali būti panaudotos keliais būdais..

Yra nemažai būtų aprašančių kaip rasti tokias kolizijas. Vienas iš naujausių vadinamas „tuneliavimu“. Šiuo būdu galima atrasti kolizijas greičiau nei per minutę. Pavyzdžiui: Pentium 4, 3.2GHz gali surasti kolizijas vidutiniškai per 17 sekundžių.

MD5 algoritmo apibendrinimas

Nors buvo surengtos 2 atakos prieš MD5, bet gali dar drąsiai teigti, jog MD5 algoritmas vis dar patikimas. Kol kas nebuvo užfiksuotos jokios atakos prieš MD5 algoritmą, kai norima sukurti failus su identiškais MD5 hash‘ais. Vienintelė vieta, kur būtų galima pasinaudoti MD5 algoritmo skyle – tai slaptažodžiai. Nes šiuo atveju jau galima panaudoti „brute force“ ataką.[21, 22]

2.6.7. CRC32 algoritmo analizė

Apie bitus ir polinomus

Polinomas su koeficientais 0 ir 1 gali būti pateikiamas kaip bitų eilė ir atvirkščiai. Tarkim polinomas $a_0 + a_1x + a_2x^2 + ... + a_nx^n$ priklauso $Z_2[x]$. Tai reiškia, kad a_i priklauso {0,1}. Koeficientai a_0, a_1, a_2, ..., a_n gali būti pateikiami binariniu formatu. Pavyzdžiui, $1 + x^2 + x^4 + x^7$ yra $1 + 0x + x^2 + 0x^3 + x^4 + 0x^5 + 0x^6 + x^7$, o jo

koeficientai yra 1,0,1,0,1,0,0,1 ir gali būti pateikti kaip binarinis skaičius 10101001 (= A9h). Pastebėtina, kad didžiausias laipsnis (x^7) yra pateikiamas paskutiniu skaičiumi. Todėl kiekviena bitų eilė gali būti skaitoma kaip $Z_2[x]$ polinomas ir atvirkščiai.

Polinominė sudėtis realizuojama XOR funkcija

a mod b yra a/b dalybos liekana [*r=a mod b* reiškia, kad *a=kb+r*, kur r yra mažiau nei b]. Dabar kad sudėti *h* ir *g* mod *m*, mes imame dalybos *(h+k) / m* liekaną. Pavyzdžiui, sudėtis *35 ir 42 mod 50* yra *(35+42) mod 50=77 mod 50=27*. Grįžkime prie Z_2, ir pažiūrėkime kas atsitinka kai mes panaudojame sudėtį mod 2. 0+0 = 0, 0+1 = 1, 1+0 = 1, bet 1+1 = 0 nes (1+1) mod 2 = 2 mod 2 = 0. Tai yra toks pat veikimas kaip eXclusive-OR binariniuose skaičiuose. Todėl mes galime sudėti $Z_2(x)$ polinomus paprasčiausiai naudodami XOR operacijas. Štai pavyzdys, kur mes sudedame $1 + x^2 + x^5$, ir $x + x^2 + x^7$, kad gauti $1 + x + x^5 + x^7$. Pastebėkime, kad $x^2 + x^2 = 0$. Tai yra XOR funkcija.

$1 + 0x + 1x^2 + 0x^3 + 0x^4 + 1x^5 + 0x^6 + 0x^7 = 10100100$, sudedame

$0 + 1x + 1x^2 + 0x^3 + 0x^4 + 0x^5 + 0x^6 + 1x^7 = 01100001$, gauname

$1 + 1x + 0x^2 + 0x^3 + 0x^4 + 1x^5 + 0x^6 + 1x^7 = 11000101$

Daugyba iš x realizuojama su SHR:

O kaip dėl daugybos iš x? Tai reiškia, kad 1 + x taps $x + x^2$, arba kad $1 + x + 0x^2$ tampa $0 + x + 1x^2$. Jei mes naudosime 8 bitus, tai reiškia 11000000 kas tampa 01100000. Iš tiesų mes tik pasislinkome į dešinę per vieną poziciją, kas gali būti lengvai atlikta naudojant SHR komandą. Taigi dabar turime, kad: kiekviena bitų eilė gali būti traktuojama kaip polinomas, XOR gali būti naudojamas sudėčiai, SHR gali būti naudojamas daugybai.

Kas yra CRC32

CRC32 yra 32 bitų ciklinis dublikatų tikrinimas, pagrinde naudojamas kaip klaidų aptikimo metodas, kai siunčiami duomenys. Jeigu pažeisti CRC bitai skiriasi nuo originalių (perduotų) CRC bitų, vadinasi įvyko klaida perdavime. Jeigu jie yra identiški, tada traktuojama kad klaidų neįvyko (yra tikimybe 1 iš 4 milijardų, kad dvi skirtingos bitų eilės turės ta patį CRC32). Idėja yra tokia, kad duomenų bitai yra

laikomi duomenų polinomais, o CRC bitai yra liekana kai padaliname duomenų polinomą iš fiksuoto ir žinomo polinomo, vadinamo CRC polinomu.

CRC32 polinomas yra $c(x) = 1 + x + x^2 + x^4 + x^5 + x^7 + x^8 + x^{10} + x^{11} + x^{12} + x^{16} + x^{22} + x^{23} + x^{26} + x^{32}$

Mums reikia gauti b(x), tada

$d(x) = a(x)c(x) + b(x)$.

Čia d(x) yra mūsų duomenys, b(x) mūsų CRC32, o a(x) - nesvarbu. Kitaip sakant, $b(x) = d(x)$ *mod* $c(x)$.

Kaip tai daroma:

Skaičiavimas atliekamas baitas po baito. Pirmame baite, turime 7 laipsnio polinomą. Pavyzdžiui, baitas 4Ah = 01001010 atitinka $x + x^4 + x^6$. Jį padauginame iš x^{32} , kad gautume polinomą, kurį galėtume padalinti iš c(x) (kuris turi laipsnį 32).

Daliklis [c(x)] padauginamas iš x laipsnio, taip gaunamas polinomas su tokiu pačiu laipsniu kaip ir dalmuo ir jis atimamas iš dalmens. Liekana yra naujas dalmuo, o koeficientas nesvarbus.

$s(x) = x^{32}$ *mod* $c(x)$ naudojamas, kad palengvinti skaičiavimus. $s(x) = 1 + x + x^2 + x^4 + x^5 + x^7 + x^8 + x^{10} + x^{11} + x^{12} + x^{16} + x^{22} + x^{23} + x^{26}$.

Kai aptinkamas pirmasis 1 bitas, liekanos reikšmė nustatoma į s(x).

Tai atliekama todėl, kad norima, kad būtų gautas x^{32} polinomas ir liekana x^{32} *mod* $c(x)$ tikrai būtų s(x).

Kiekvienam naujam bitui, liekana dauginama iš x. Kintamajam naudojamas SHR 1. Atsiranda naujas bitas. Tai reiškia, kad paskutinis polinomas buvo vienu laipsniu didesnis. Naujos liekanos kintamasis padauginamas iš x, ir liekana dabar pasikeičia į x^{33} *mod* c(x). Bitų operacija yra shr 1. Kai sutinkamas 0, nedaroma niekas. Kai sutinkamas 1, XOR pagalba s(x) pridedamas prie liekanos.

Taip yra todėl, kad yra d(x)=a(x)c(x) + b(x). Kaip matome, turime: d(x) + x^{32}. Tai reiškia, kad d(x) + x^{32}=a(x)c(x) + b(x) + x^{32}. Liekana yra x^{32} + b(x). Tai yra nepriimtina, jei tai yra polinomas su tokiu pačiu laipsniu kaip daliklis, todėl tokiu atveju gali būti išskaidomas.[x^{32} + b(x)] *mod* c(x) = [x^{32} *mod* c(x)] + [b(x) *mod* c(x)] = s(x) + b(x). Taigi reikia naudoti XOR pridedant s(x). Jeigu ateina bitas su reikšme (po shr 1) 1, vykdomas XOR sudėjimas s(x).

Atliekant daugybą iš x (SHR 1) gali būti, kad liekanos laipsnis bus 32. Tada išmetamas vienas bitas iš 32 bitų ilgio kintamojo. Reikia jį padalinti iš c(x), naudojant *mod*. Taigi x^{32} + b(x) (kintamasis yra b(x) ir jis neturi x^{32}). Kaip jau minėta anksčiau, reikalinga panaudoti XOR sudėtį s(x).[23]

2.6.8. **MD4 algoritmo analizė**

Įžanga

MD4 algoritmas kaip įėjimą paima tam tikro ilgio žinutę ir išėjime išveda 128 bitų žinutės kodą. Šis kodas yra unikalus kiekvienam įėjimui, jis dar vadinamas žinutės pirštų anspaudu. MD4 naudojamas skaitmeninio parašo programoms, kur didelis failas turi būti paverstas saugiu, prieš užkodavimą privačiu slaptu raktu tokioje sistemoje kaip RSA.

MD4 algoritmas sukurtas taip kad veiktų greitai 32 bitų kompiuteriuose. Be to, MD4 nereikalauja didelių substitucijos lentelių, algoritmas gali būti suprogramuotas gana kompaktiškai.

Apie MD4

Turima žinutė užima b bitų, norima surasti žinutės kodą. Tariama, kad b yra ne neigiamas sveikas skaičius, b taip pat gali būti ir 0, reikalinga, kad jis būtų dalus iš 8. Užrašyti žinutės bitai įsivaizduojami taip:

m_0 m_1 ... m_{b-1}

Yra 5 žingsniai, pagal kuriuos suskaičiuojamas žinutės kodas.

Pirmasis žingsnis. Išplėstų bitų prijungimas.

Žinutė skaitosi išplėsta, kai jos ilgis bitais yra artimas 448 mod 512. Išplėtimas reikalingas, nes žinutė būna 64 bitų, o ją reikia padalinti iš 512 bitų. Išplėtimas naudojamas visada, net jeigu ir iškarto žinutės ilgis būna artimas 448 mod 512. Išplėtimas atliekamas taip: prie žinutės prirašomas bitas su reikšme „1“, tada prijungiama tiek bitų su reikšme „0“, kiek reikalinga, kad išplėsta žinutė būtų artima

448 mod 512. Apibendrinus, bent vienas arba daugiausiai 512 bitų yra prijungiama prie žinutės.

Antrasis žingsnis. Ilgio prijungimas.

64 bitų b reprezentacija (žinutės ilgis, prieš tai kai išplėtimo bitai buvo pridėti) prijungiama prie praeito žingsnio rezultato. Jeigu atsitinka taip, kad b yra didesnis nei 2^64, tada naudojami tik pradiniai 64 bitai. (Jie yra prijungiami kaip du 32 bitų žodžiai)

Šiame taške žinutė turi ilgį (po to kai išplečiama su bitais ir b), kuris tiksliai dalinasi iš 512 bitų. Taip pat ji turi ilgį, kuris dalinasi iš 16 žodžių (32 bitų). Tegul M[0 ... N-1] reiškia žinutės žodžius, kur N yra dalyba iš 16

Trečiasis žingsnis. Sukuriamas MD buferis.

Keturių žodžių buferis (A, B, C, D) naudojamas žinučių kodams palyginti. Čia kiekvienas A, B, C, D yra 32 bitų registras. Šie registrai yra sukuriami šiomis šešioliktainėmis reikšmėmis.

```
word A: 01 23 45 67
word B: 89 ab cd ef
word C: fe dc ba 98
word D: 76 54 32 10
```

Ketvirtasis žingsnis. Paversti žinutę į 16 žodžių blokus.

Pirmiausiai reikia išsiaiškinti apie tris funkcijas, kurios kiekviena paima 3 32 bitų žodžius kaip įėjimą, o išėjime gaunasi vienas 32 bitų žodis. Šios trys funkcijos užrašomos taip:

F(X,Y,Z) = XY v not(X) Z

G(X,Y,Z) = XY v XZ v YZ

H(X,Y,Z) = X xor Y xor Z

Kiekvienoje bito pozicijoje F elgiasi kaip sąlyga: Jei X, tai Y, arba Z (if X then Y else Z). Funkcija F gali būti realizuojama ir naudojant „+“ vietoj „v“, jeigu XY ir not(X) Z niekada neturės bito reikšmės vienas toje pačioje bito pozicijoje. Kiekvienoje bito pozicijoje G elgiasi kaip pagrindinė funkcija: jei bent dvi X, Y, Z veikia, tada G turi bito reikšmę „1“ toje pozicijoje, jei ne, G toje pozicijoje turi reikšmę „0“. Funkcija H yra bitų požiūriu „OR“ ar „parity“ funkcija. Ji turi panašius parametrus kaip F ir G funkcijos.

Penktasis žingsnis. Išėjimas.

Žinutės kodas išėjime gaunamas iš A, B, C, D. Tai yra, pradedama nuo žemutiniojo (low-order) A bito ir baigiame aukštutiniuoju (high-order) D bitu.

MD4 algoritmo apibendrinimas

MD4 algoritmą nesunku realizuoti ir surasti žinutės kodą, kuris yra unikalus ir yra dar vadinamas žinutės pirštų anspaudu. Yra spėjama, kad sudėtingumas padaryti žinutes su vienodu kodu susideda iš 2^64 operacijų, ir sudėtingumas, kad ateis žinutė su jau turimu kodu susideda iš 2^128 operacijų. MD4 buvo atidžiai struktūrizuotas, kad neturėtų silpnybių. Tačiau šis algoritmas yra gana naujas ir jam reikalingi tolimesni saugumo ir trūkumų tyrimai.[24,25]

2.6.9. SHA1 algoritmo analizė

Įžanga

SHA1 (arba „Secure Hash Algorithm“) – tai duomenų kodavimo algoritmas, kurio pagalba galima sugeneruoti tų duomenų reprezentaciją, kurią galima pavadinti „duomenų santrauka“. SHA1 algoritmą reikia naudoti kartu su „Digital Signature Algorithm (DSA)“ , taip pat kai reikia naudoti saugų maišos algoritmą.

SHA1 naudojamas kai norima sukurti kažkokios žinutės ar duomenų failo sutrumpintą atitikmenį („hash“). Kai bet kokio ilgio ($< 2^{64}$) žinutė paduodama į SHA1 algoritmą, jis sugeneruoja 160 bitų tos žinutės santrauką.

SHA1 laikomas saugiu, nes neįmanoma kompiuteriu sugeneruoti norimos santraukos ar rasti dvi skirtingas žinutes, kurios duotų tą pačią santrauką. Bet koks žinutės pakeitimas duotų visiškai kitokią santrauką, todėl pasinaudojus SHA1 algoritmą nepavyktų apeiti numatytų saugos priemonių. SHA1 – tai atnaujinta SHA (FIPS 180) versija papildyta papildomomis saugumą užtikrinančiomis priemonėmis. SHA1 algoritmas paremtas panašiais principais, kurie naudojami MD4 maišos algoritme.

SHA1 specifikacija:

• Žinutė ar duomenų failas turėtų būti traktuojamas kaip bitų eilę.

• Žinutės ilgis yra bitų skaičius žinutėje (tuščios žinutės ilgis būtų = 0).

• Jei būtų skaičius žinutėje yra 8 kartotinis, tai tokią žinutę galima pateikti „hex‘u“, kad būtų galima sutaupyti kažkiek vietos.

• Žinutės ilgis padidinamas, nes reikia jį padaryti 512 kartotiniu.

• Kurdama žinutės santrauką SHA1 algoritmas iš eilės apdoroja informacijos blokus po 512 bitų.

SHA1 veikimo principas

Maišos funkcija SHA-1 algoritme paima žinutę, kurios ilgis mažesnis nei 2^{64} bitai ir apdoroja į 160 bitų santrumpą. Tada įvesties žinutė yra išplečiama ir apgodojama kaip 512 bitų blokais „Damgard/Merkle“ besikartojančioje struktūroje. Kiekvienas ciklas aktyvuoja kompresijos funkciją, kuri paima 160 bitų pradinę reikšmę ir 512 bitų ilgio žinutės bloką. Tada išveda kitą 160 bitų pradinę reikšmę. Pirminė reikšmė (vadinama IV) sudaryta iš fiksuotų konstantų, o galutinė reikšmė – tai jau sudarytas žinutės „hash‘as“.

Žemiau pateiktame paaiškinime parodysime kaip veikia SHA1 algoritmo suspaudimo funkcija.

Kiekvienas 512 bitų išplėstos žinutės blokas suskaldomas į 16 32 bitų žodžius (m_0,m_1,...,m_{15}). Iš pradžių žinutės žodžiai yra išplečiami taip:

$$m_i = (m_{i-3} \oplus m_{i-8} \oplus m_{i-14} \oplus m_{i-16}) \ll 1.$$

Išskleistos žinutės žodžiai apdorojami keturiais etapais, kur kiekvieną sudaro po 20 žingsnių. Vieno žingsnio funkcija apsirašo taip:

For $i = 1, 2, ..., 80$,

$$a_i = (a_{i-1} \ll 5) + f_i(b_{i-1}, c_{i-1}, d_{i-1}) + e_{i-1} + m_{i-1} + k_i$$
$$b_i = a_{i-1}$$
$$c_i = b_{i-1} \ll 30$$
$$d_i = c_{i-1}$$
$$e_i = d_{i-1}$$

Pradinė reikšmė *IV* = (a_0, b_0, c_0, d_0, e_0) apsirašo kaip:

(0*x*67452301*;* 0*xefcdab*89*;* 0*x*98*badcfe;* 0*x*10325476*;* 0*xc*3*d*2*e*1*f*0)

Kiekvienas etapas naudoja skirtingą „Boolean“ funkciją f_i *ir konstantą* k_i, kurias galima pamatyti 19 pavyzdyje [26, 27]:

round	step	Boolean function f_i	constant k_i
1	$1-20$	IF: $(x \wedge y) \vee (\neg x \wedge z)$	0x5a827999
2	$21-40$	XOR: $x \oplus y \oplus z$	0x6ed6eba1
3	$41-60$	MAJ: $(x \wedge y) \vee (x \wedge z) \vee (y \wedge z)$	0x8fabbcdc
4	$61-80$	XOR: $x \oplus y \oplus z$	0xca62c1d6

19 pav.: Boolean funkcijos ir konstantos naudojamos SHA1

2.7. Siekiamos sistemos apibrėžimas

Norint apsaugoti duomenis pernešamose laikmenose, tokiose kaip USB atmintinė, šiuo metu nėra vieno optimalaus sprendimo, kurio pagalba būtų galima pasiekti visų vartotojų keliamų reikalavimų. Vienos, jau egzistuojančios, sistemos suteikia didesnę galimybę kontroliuoti norima saugumo lygi nei kitos, tačiau arba tokioms sistemoms reikia specialios ir brangios programinės įrangos, arba sistema būna orientuota į didelių kompanijų poreikius. Taip pat atsiranda problema, jog pasinaudojus viena apsaugos sistema duomenys tampa ne tik apsaugoti, bet ir neprieinami pačiam vartotojui kitose vietose be specialios programinės įrangos. Todėl kruopščiai išanalizavus daugumą esamų sprendimų, metodų mūsų sprendimas būtų suprojektuoti naujos kartos USB atmintinių apsaugos sistemą, kuri remtųsi debesų technologijomis ir leistų vartotojui ne tik apsaugoti atmintinės turinį, bet ir turėti jos atsarginę kopiją „debesyse“, taip pat nebūti priklausomam nuo jokios papildomos programinės įrangos bereikalingo diegimo. Taip būtų išspręsta daugelis trūkumų, kurie buvo aptikti šioje analizėje jau aptartų sistemų.

2.8. Analizės išvados

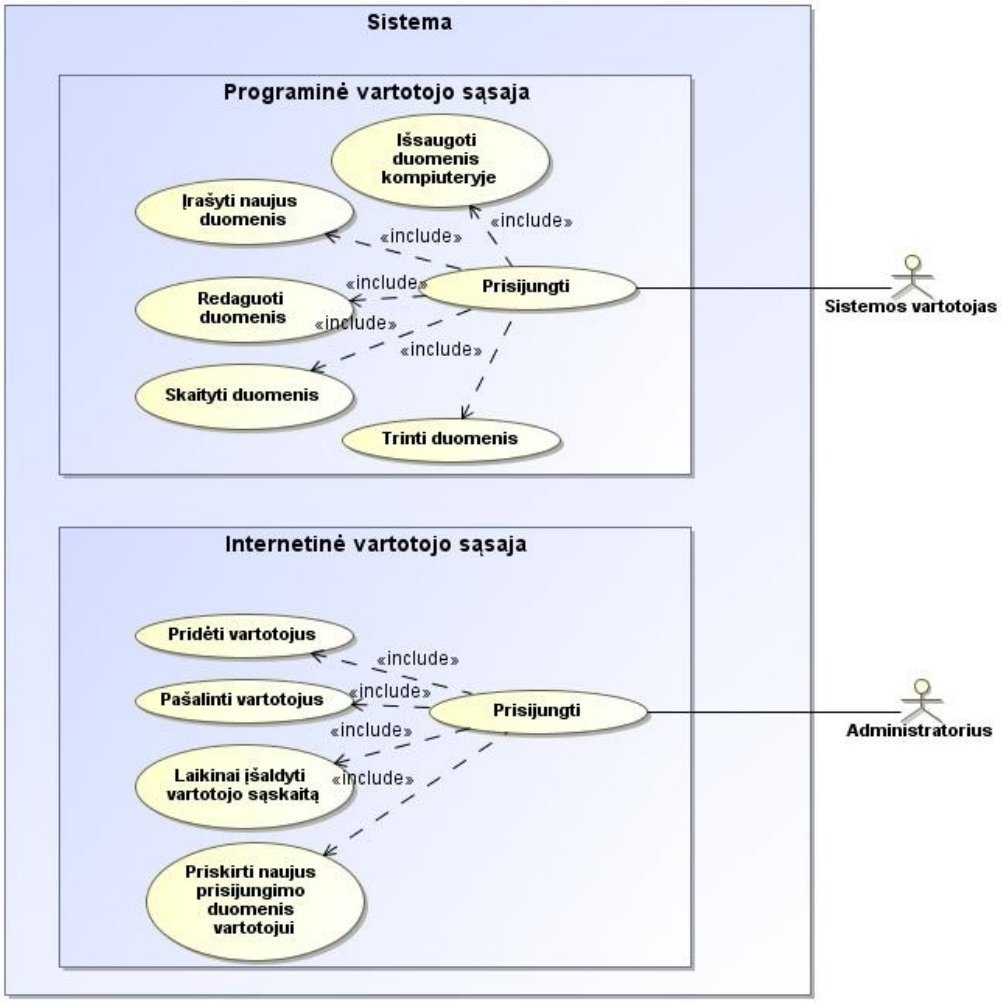
- Atlikus jau egzistuojančių sprendimų analizę paaiškėjo, kad beveik nėra sprendimų, kurie nereikalautu papildomos programinės įrangos diegimo į kompiuterius, prie kurių būtų jungiama pernešamoji laikmena;
- Išanalizavus duomenų šifravimo metodus nustatyta, jog šiuo metu nėra visiškai saugių algoritmų, todėl siūlomas metodas neturėtų visiškai pasikliauti vien šifravimo algoritmais duomenų USB atmintinėje saugumui užtikrinti;
- Programinė įranga, kuri dabar yra rinkoje, paprastam vartotojui gali būti per brangi ar per sudėtinga naudotis.

3. Duomenų pernešamose laikmenose apsaugos metodų reikalavimų specifikacija ir analizė

3.1. Reikalavimų specifikacija

3.1.1. Funkciniai reikalavimai

Sistemos funkcinius reikalavimus atspindi panaudojimo atvejų diagrama, ji nusako sistemos funkcionalumą. Sistemos vartotoju gali būti bet kuris asmuo.



20 pav.: Kuriamos sistemos panaudojimo atvejų diagrama

10 lentelė: Panaudojimo atvejo „Prisijungti“ aprašas

Panaudojimo atvejis	Prisijungti
Aprašymas	Vartotojas įveda prisijungimo duomenis, kad patvirtintų tapatybę sistemoje
Aktoriai	Sistemos vartotojas
Prieš-sąlyga	Programa startuota
Rezultatai	• Prisijungta. Atsiranda vartotojo sąsajos langas, kuriame yra apsaugotų failų sąrašas. • Neprisijungta. Prašoma pakartotinai įvesti vartotojo duomenis arba išjungti sistemą

11 lentelė: Panaudojimo atvejo „Skaityti duomenis“ aprašas

Panaudojimo atvejis	Skaityti duomenis
Aprašymas	Vartotojas pasirenka iš rodomo sąrašo norimą failą ir jį atidaro.
Aktoriai	Vartotojas
Prieš-sąlyga	Prisijungti
Rezultatai	Vartotojas gali naudoti apsaugotus duomenis

12 lentelė: Panaudojimo atvejo „Redaguoti duomenis“ aprašas

Panaudojimo atvejis	Redaguoti duomenis
Aprašymas	Vartotojas atidaro saugomą failą ir jį gali redaguoti
Aktoriai	Vartotojas
Prieš-sąlyga	Skaityti duomenis
Rezultatai	Vartotojas gali redaguoti saugomus duomenis

13 lentelė: Panaudojimo atvejo „Trinti duomenis“ aprašas

Panaudojimo atvejis	Trinti duomenis
Aprašymas	Vartotojas pašalina failą iš saugomų failų sąrašo
Aktoriai	Vartotojas
Prieš-sąlyga	Prisijungti
Rezultatai	Vartotojas ištrina nereikalingus failus

14 lentelė: Panaudojimo atvejo „Įrašyti duomenis“ aprašas

Panaudojimo atvejis	Įrašyti duomenis
Aprašymas	Vartotojas į saugomų failų sąrašą gali įtraukti naują failą iš pasirinkto katalogo
Aktoriai	Vartotojas
Prieš-sąlyga	Prisijungti
Rezultatai	Į saugomų failų sąrašą įtraukiamas naujas failas iš pasirinkto katalogo. Failas nusiunčiamas į „debesį“.

15 lentelė: Panaudojimo atvejo „Išsaugoti duomenis kompiuteryje“ aprašas

Panaudojimo atvejis	Išsaugoti duomenis kompiuteryje
Aprašymas	Vartotojas gali pašalinti failą iš saugomų failų sąrašo, tuo pačiu jis pašalinamas ir iš debesio
Aktoriai	Vartotojas
Prieš-sąlyga	Prisijungti
Rezultatai	Vartotojas ištrina nereikalingus failus

16 lentelė: Panaudojimo atvejo „Prisijungti“ aprašas

Panaudojimo atvejis	Prisijungti
Aprašymas	Administratorius prisijungia prie administravimo vartotojo sąsajos per interneto naršyklę
Aktoriai	Administratorius
Prieš-sąlyga	Pasiekti sistemą internetu
Rezultatai	Administratorius prisijungė prie sistemos administravimo per interneto naršyklę ir gali atlikti sekančius veiksmus

17 lentelė:. Panaudojimo atvejo „Pridėti vartotojus“ aprašas

Panaudojimo atvejis	Pridėti vartotojus
Aprašymas	Administratorius gali pridėti naujus vartotojus sistemoje, t.y. sukurti jiems prieigą
Aktoriai	Administratorius
Prieš-sąlyga	Prisijungti
Rezultatai	Administratorius įvedė vieną ar kelis naujus vartotojus

18 lentelė: Panaudojimo atvejo „Pašalinti vartotojus“ aprašas

Panaudojimo atvejis	Pašalinti vartotojus
Aprašymas	Administratorius gali pašalinti sistemoje registruotus vartotojus
Aktoriai	Administratorius
Prieš-sąlyga	Prisijungti
Rezultatai	Administratorius pašalino vieną ar kelis vartotojus, kurie buvo registruoti sistemoje

19 lentelė: Panaudojimo atvejo „Laikinai įšaldyti vartotojo sąskaitą“

Panaudojimo atvejis	Laikinai įšaldyti vartotojo sąskaitą
Aprašymas	Administratorius, gavęs pranešimą iš vartotojo, kad šis prarado išorinę laikmeną ar autorizacijos duomenis, gali laikinai sustabdyti vartotojo sąskaitą sistemoje, kol vartotojas patvirtins kad nerado arba surado tą kas buvo prarasta.
Aktoriai	Administratorius
Prieš-sąlyga	Prisijungti
Rezultatai	Vartotojo sąskaita buvo laikinai sustabdyta

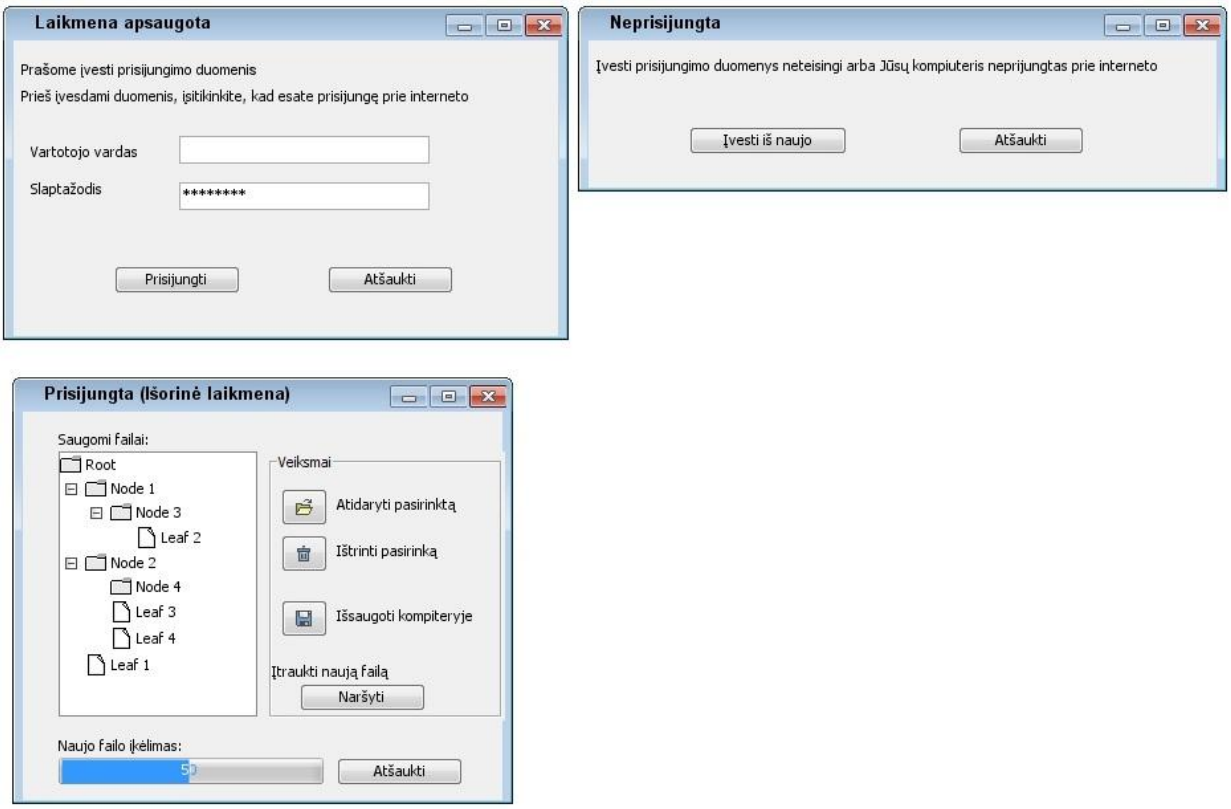
20 lentelė: Panaudojimo atvejo „Priskirti naujus prisijungimo duomenis vartotojui“ aprašas

Panaudojimo atvejis	Priskirti naujus prisijungimo duomenis vartotojui
Aprašymas	Administratorius gavęs patvirtinimą, kad vartotojas tikrai prarado laikmeną ir vartotojui pageidaujant, priskiria jo sąskaitai naujus prisijungimo duomenis.
Aktoriai	Administratorius
Prieš-sąlyga	Prisijungti
Rezultatai	Administratorius priskyrė naujus prisijungimo duomenis vartotojui

Planuojamos sistemos vartotojo sąsajos prototipas yra lengvai suprantama ir naudojama bei intuityvi kiekvienam vartotojui programinė įranga.

Sistemos administratoriaus vartotojo sąsaja bus sukurta interneto naršyklės aplinkoje, vadovaujantis standartiniu šablonu, todėl čia jos prototipas pavaizduotas nebus.

Sistemos vartotojo sąsajos pagrindiniai langai pavaizduoti 21 paveiksle.



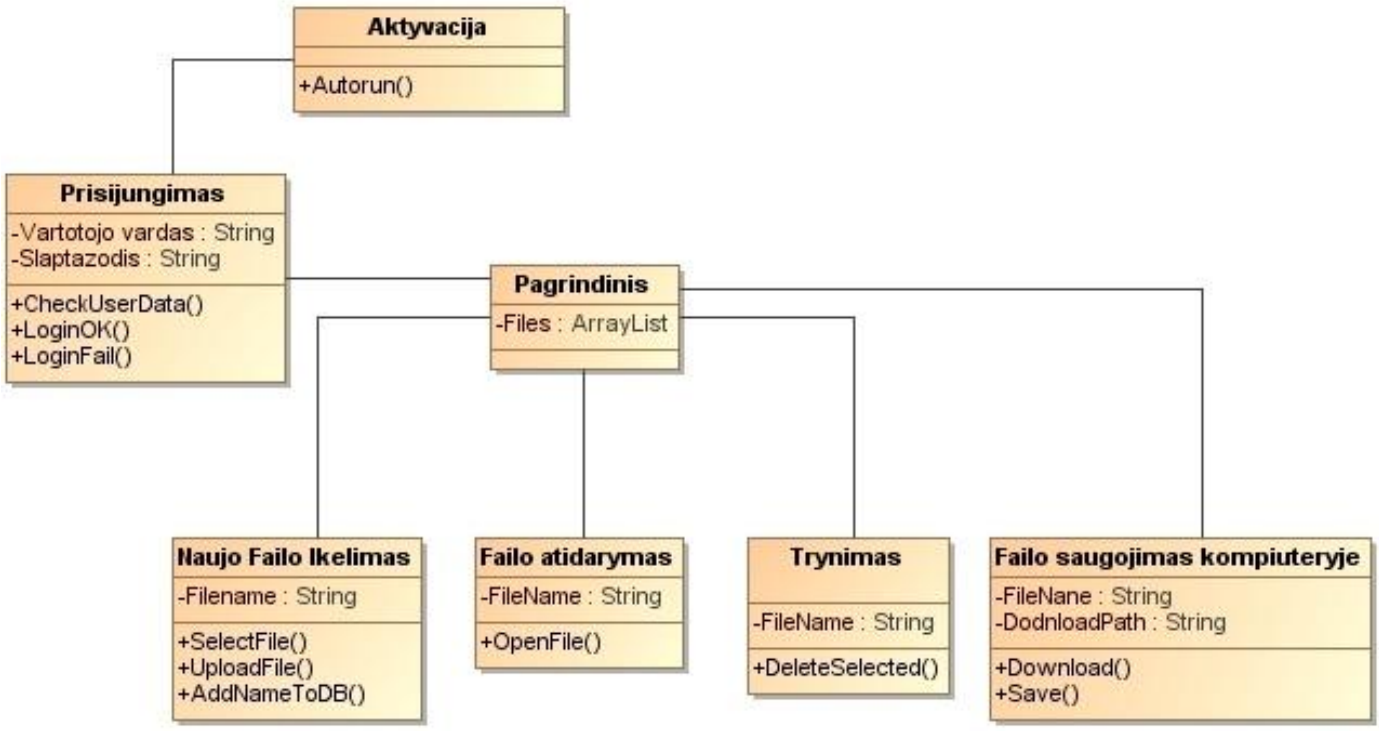
21 pav.: Vartotojo sąsajos prototipas

3.1.2. Nefunkciniai reikalavimai

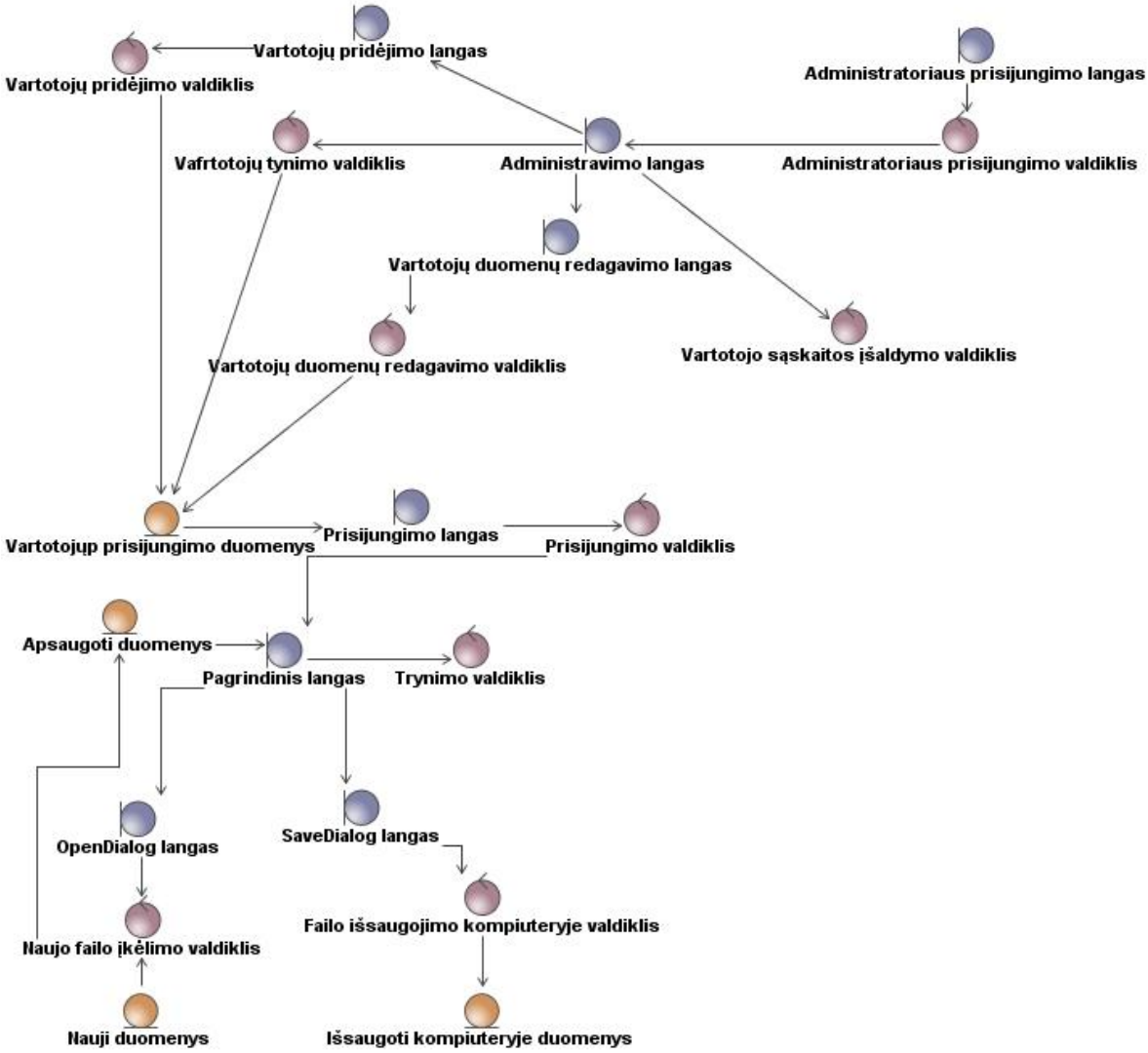
- Vartotojo dalies sistema turi veikti su *Windows XP*, *Windows Vista*, bei *Windows 7* operacinėmis sistemomis;
- Administratoriaus dalies sistema turi veikti su bet kuria interneto naršykle;
- Sistema turi būti lengvai suprantama ir intuityvi bet kuriam vartotojui;
- Sistemos klaidų skaičius negali viršyti 5%;
- Sistema turi palaikyti visų atminties dydžių išorines laikmenas.

3.2. Dalykinės srities modelis

Dalykinės srities modelis atvaizduotas UML klasių diagrama. Šio modelio tikslas atvaizduoti galimą sistemos architektūrą. Vaizduojamas tik pagrindinės klasės, be ryšių bei atributų su metodais. Klasių diagrama atvaizduojama tik vartotojo sistemos daliai, nes administratoriaus sistemos dalis kuriama internetinėje erdvėje. Vėliau klasių pavadinimai bei atributų ir metodų pavadinimai gali pasikeisti.



22 pav.: Dalykinės srities klasių diagrama



23 pav.: Dalykinės srities robustness diagrama

3.3. Reikalavimų apibendrinimas

Atlikos reikalavimų analizę, buvo suformuoti programinės įrangos panaudojimo atvejai bei apibrėžta dalykinė sritis. Šiame etape buvo apibrėžtas ir vartotojo sąsajos prototipas. Remiantis visais šiais duomenimis, tolimesniuose etapuose bus realizuojamas sistemos projektas.

4. Duomenų pernešamose laikmenose apsaugos metodų tyrimo projektas (metodas, modelis)

4.1. Sistemos/sprendimo/metodo/modelio pagrindimas ir esmės išdėstymas

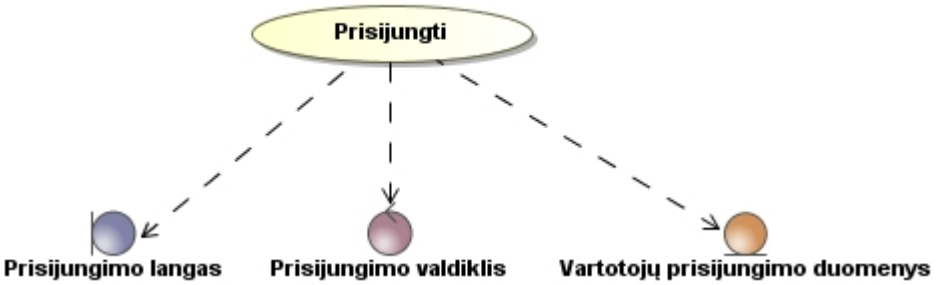
Kuriamos sistemos tikslas apsaugoti vartotojo turimus duomenis. Sistemos pagalba, vartotojas gali apsaugoti duomenis nuo neteisėto jų naudojimo.

Atsižvelgiant į esamus sprendimus ir į esamus apsaugos metodų apėjimo būdus, nuspręsta vartotojo saugomus failus laikyti ne kaip nors užkoduotus ir suarchyvuotus pačioje laikmenoje, o pasitelkti debesų technologijos idėją ir visus failus laikyti nutolusiame serveryje – t.y. debesyje. Taip saugant vartotojo duomenis failų praradimo rizika sumažėja iki minimumo laikmenos pametimo ar mechaninio pažeidimo atveju. Vos tik pasigedęs laikmenos, vartotojas gali tuoj pat informuoti administratorių, kuris kaip mat „įšaldo“ priėjimą prie vartotojo duomenų.
Sistema susideda iš kelių pagrindinių dalių, vartotojo programinės įrangos ir administratoriaus programinės įrangos, serverio, duomenų bazės, failų talpyklos. Joms tarpusavyje sąveikaujant interneto pagalba, užtikrinamas vartotojo duomenų saugumas.

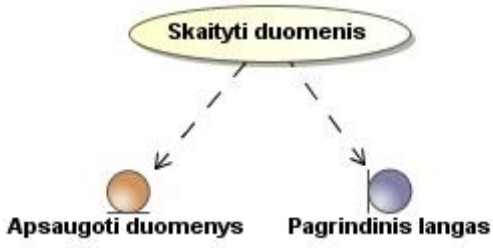
4.2. Sistemos architektūra

4.2.1. Reikalavimų analizė

Kiekvienas panaudojimo atvejis nagrinėjamas sudarant panaudojimo atvejo analizės diagramą. Diagramoje nustatomos loginės klasės, dalyvaujančios vykdant panaudojimo atvejį, jos pateikiamos žemiau.

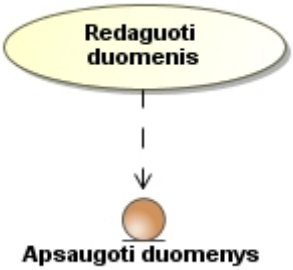


24 pav.: Panaudojimo atvejo „Prisijungti“ analizės diagrama

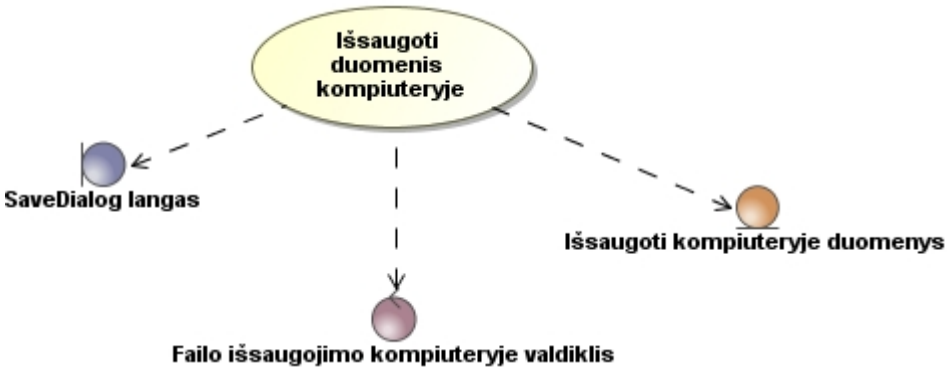


25 pav.: Panaudojimo atvejo „Skaityti duomenis“ analizės diagrama

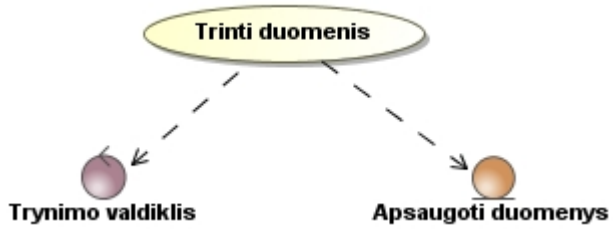
26 pav.: Panaudojimo atvejo „Įrašyti naujus duomenis“ analizės diagrama:



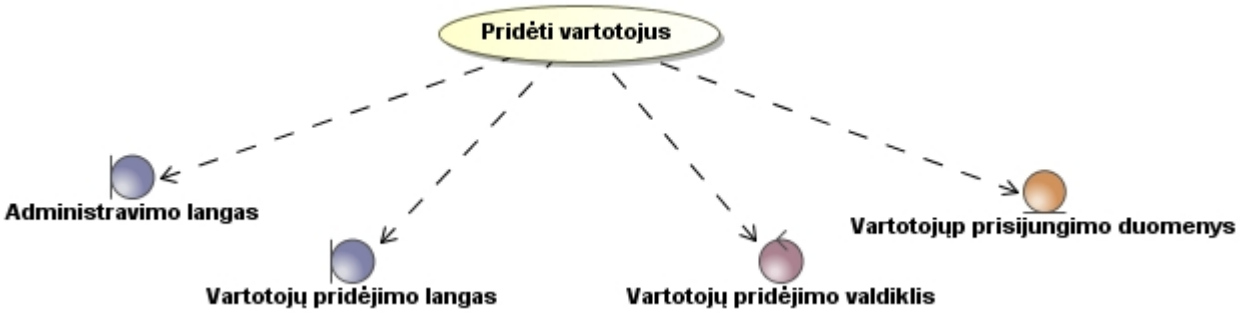
27 pav.: Panaudojimo atvejo „Redaguoti duomenis“ analizės diagrama



28 pav.: Panaudojimo atvejo „Išsaugoti duomenis kompiuteryje“ analizės diagrama



29 pav.: Panaudojimo atvejo „Trinti duomenis“ analizės diagrama



30 pav.: Panaudojimo atvejo „Pridėti vartotojus“ analizės diagrama

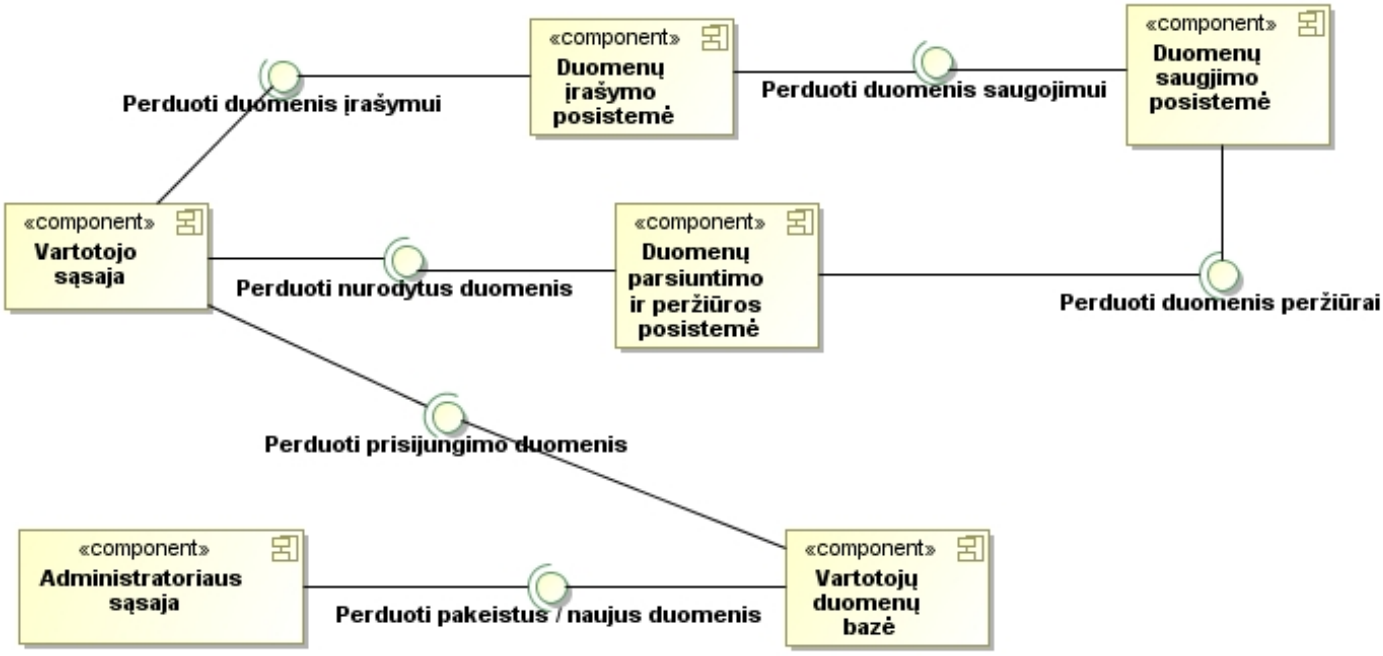
31 pav.: Panaudojimo atvejo „Pašalinti vartotojus“ analizės diagrama

32 pav.: Panaudojimo atvejo „Laikinai įšaldyti vartotojo sąskaitą“ analizės diagrama

33 pav.: Panaudojimo atvejo „Priskirti naujus prisijungimo duomenis vartotojui“ analizės diagrama

4.2.2. Loginė visos sistemos architektūra

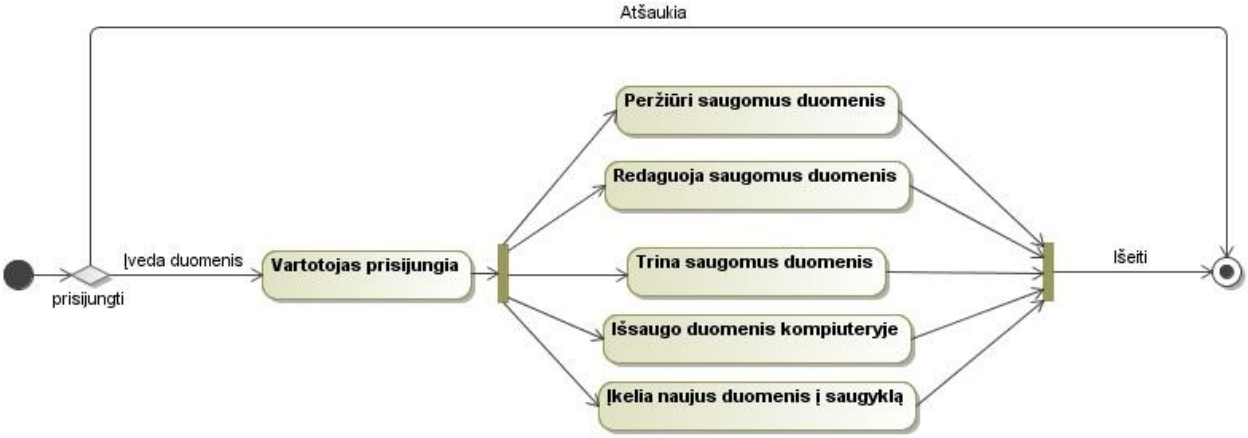
Sistema susideda iš šešių komponentų. Sistemą sudaro „Vartotojo sąsaja“, „Administratoriaus sąsaja“, „Duomenų įrašymo posistemė“, „Duomenų parsiuntimo posistemė“, „Duomenų saugojimo posistemė“ ir „Vartotojų duomenų bazė“. Vartotojo ir administratoriaus sąsajos atsakingos už grafinį atvaizdavimą, o kitos posistemės dirba su duomenimis pagal šių dviejų vartotojo sąsajų užklausas.



34 pav.: Sistemos loginė architektūra

4.2.3. **Veiklos paslaugos**

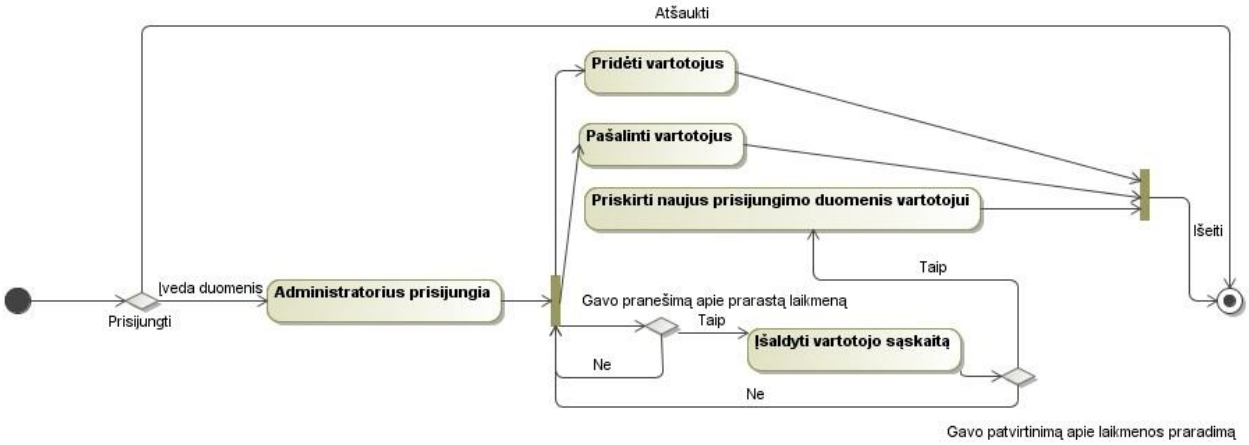
Vartotojo darbą su programa parodo veiklos diagrama. Aktyvavęs programą, vartotojui pateikiamas langas su prisijungimu, jis gali prisijungti arba atšaukti. Jei atšaukiama, programa uždaroma. Prisijungęs vartotojas gali atlikti pasirinktus veiksmus, t.y. peržiūrėti saugomus duomenis, juos redaguoti, trinti, išsaugoti pasirinktus duomenis kompiuteryje bei iš kompiuterio pasirinktą failą patalpinti į apsaugotą nuotolinę saugyklą. Veiklos diagrama pateikiama 35 paveikslėlyje.



35 pav.: Vartotojo veiklos procesas

Administratoriaus darbą su programa taip pat parodo veiklos diagrama. Administratoriui į interneto naršyklės adresų laukelį įvedus atitinkamą adresą, užkraunamas administravimo puslapis. Administratorius gali pasirinkti, arba prisijungti arba uždaryti naršyklę ar skirtuką. Prisijungęs administratorius mato visų registruotų vartotojų sąrašą. Jis gali kurti naujus vartotojus, taip pat esamus pašalinti. Jeigu administratorius gavo pranešimą telefonu, el. paštu ar kitaip, kad vienas iš vartotojų pametė laikmeną, administratorius laikinai įšaldo vartotojo sąskaitą, vartotojas per kelias dienas dar gali ieškoti laikmenos ir ją rasti, todėl sąskaita įšaldoma tik laikinai. Vartotojui patvirtinus, kad laikmena jau galutinai prarasta, administratorius jam sukuria naujus prisijungimo duomenis, o visi nutolusioje saugykloje esantys duomenys išlieka ir juos naudoti vartotojas gali prisijungęs jau su naujais duomenimis. Vartotojo sąsajos programą į laikmeną vartotojui gali įrašyti administratorius, gali įsirašyti ir pats vartotojas, jeigu yra tinkamai įgudęs.

Administratoriaus veiklos procesas pavaizduotas apačioje.

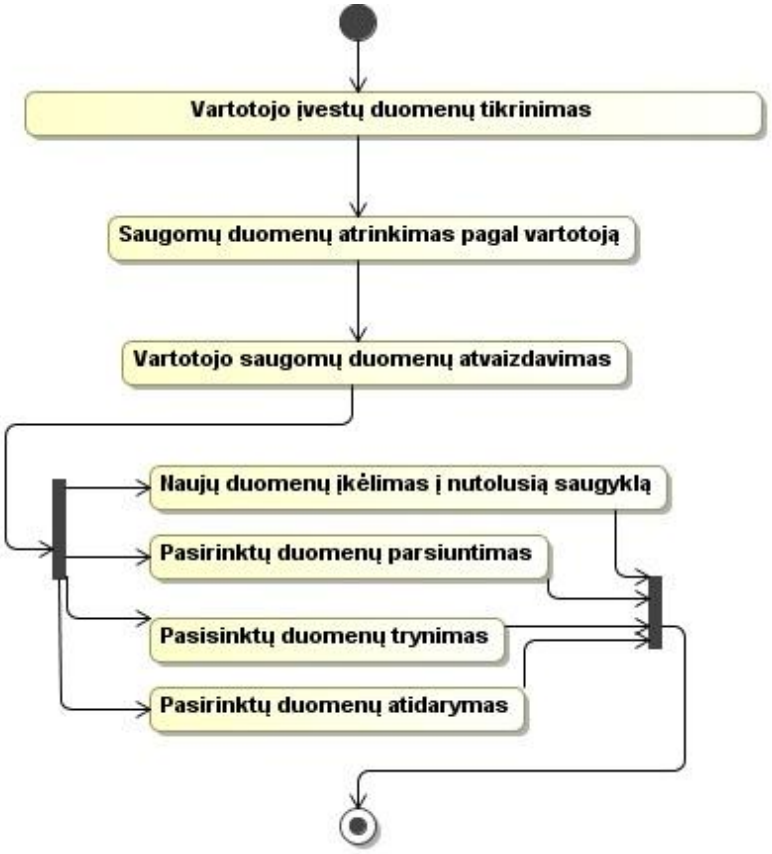


36 pav.: Administratoriaus veiklos procesas

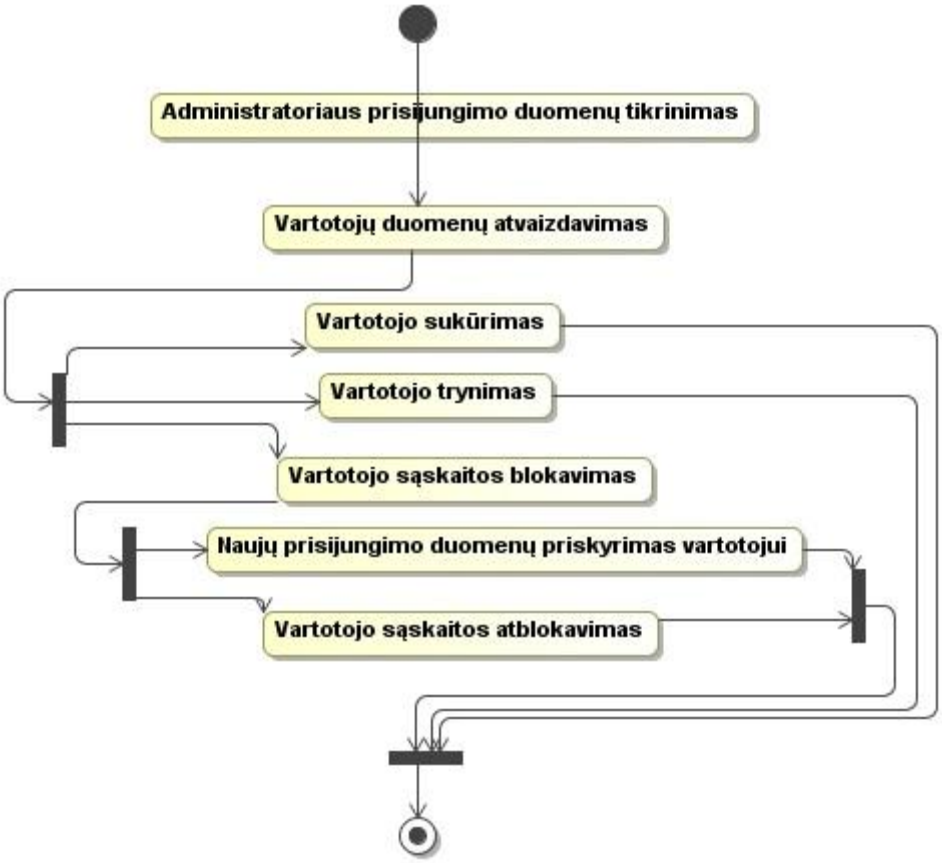
4.3. Sistemos elgsenos modelis

Sistemos elgsenos modelis vaizduoja sistemos viduje vykstančius procesus. Jis aprašomas būsenų diagrama.

Sistema naudojantis vartotojui ir administratoriui, joje vyksta skirtingi procesai, todėl modelis atvaizduojamas dvejomis būsenų diagramomis – kai sistemą naudoja administratorius ir kai sistemą naudoja vartotojas. Jos vaizduojamos apačioje.



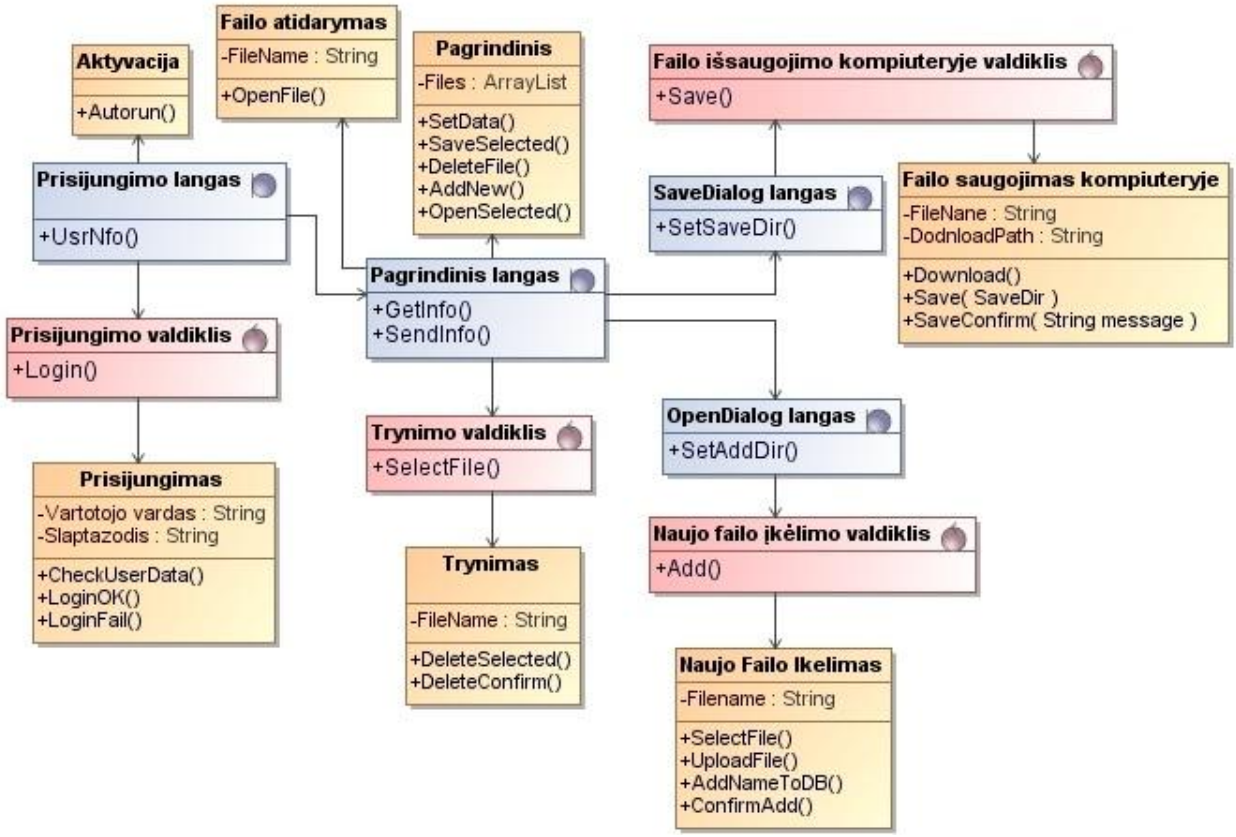
37 pav.: Sistemos būsenų diagrama, kai ja naudojasi vartotojas



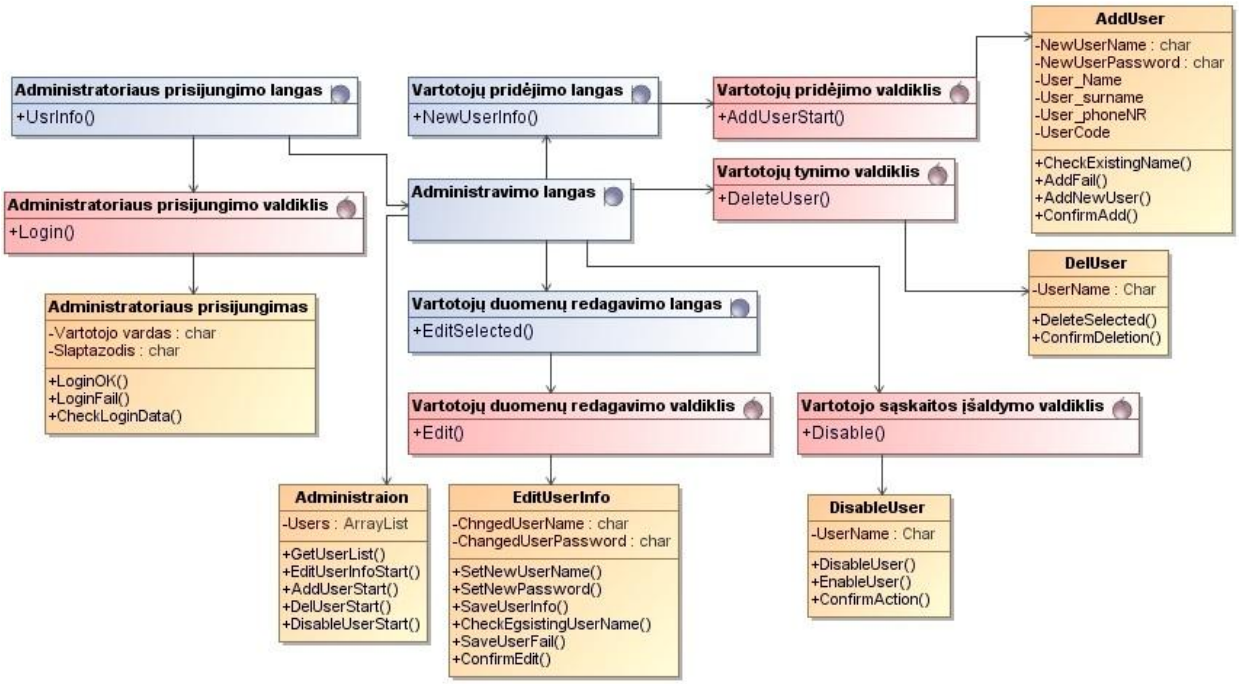
38 pav.: Sistemos būsenų diagrama, kai ja naudojasi Administratorius

4.4. Detalus projektas

Sistema susideda iš dviejų dalių, kurios realizuojamos skirtingai, vartotojo dalis veikia Windows langų principu, o administratoriaus dalis veikia interneto naršyklėje. Todėl kiekvienai sistemos daliai pavaizduota po atskirą detalaus projekto klasių diagramą. Jose vaizduojama ne tik klasės, bet ir klasių valdikliai bei langai. Detalaus projekto klasių diagramos vaizduojamos apačioje.



39 pav.: Detalaus projekto klasių diagrama vartotojo daliai



40 pav.: Detalaus projekto klasių diagrama Administratoriaus daliai

Klasių metodų aprašymai vartotojo programos daliai pateikiami 21-oje lentelėje.

21 lentelė: Klasių metodų aprašymas vartotojo programos daliai

Metodas	Aprašymas
Klasė „Prisijungimas“, skirta vartotojo prisijungimui prie sistemos	
CheckUserData()	Metodas tikrina įvestus vartotojo prisijungimo duomenis su esančiais nutolusioje duomenų bazėje
LoginOK()	Metodas leidžia pateikti į pagrindinį vartotojo langą
LoginFail()	Metodas praneša, kad įvesti blogi prisijungimo duomenys ir siūlo pakartoti arba atšaukti prisijungimą
Klasė „Pagrindinis“, skirta veiksmų valdymui iš pagrindinio lango	
SetData()	Metodas skirtas nuskaityti vartotojo saugomus failus pagal jo vartotojo prisijungimo duomenis ir juos atvaizduoti programos lange
SaveSelected()	Metodas, skirtas iškviesti naujo failo išsaugojimo kompiuteryje klasę.
DeleteFile()	Metodas, skirtas iškviesti trynimo klasę.
AddNew()	Metodas, skirtas iškviesti naujo failo įkėlimo klasę.
OpenSelected()	Metodas, skirtas iškviesti failo atidarymo klasę.
Klasė „Failo saugojimas kompiuteryje“, skirta išsaugoti pasirinktą failą į kompiuteryje nurodytą direktoriją, kuri pasirenkama SaveDialog pagalba	
Download()	Metodas, skirtas failo parsiuntimui iš nutolusios saugyklos
Save (SaveDir)	Metodas, skirtas parsiųsto failo išsaugojimui pasirinktoje direktorijoje
SaveConfirm (String message)	Metodas, skirtas failo parsiuntimo patvirtinimui.
Klasė „Naujo failo įkėlimas“, skirta pridėti naują failą į saugomų failų sąrašą ir taip pat nusiunčia jį į nutolusią failų saugyklą	
SelectFile()	Metodas, skirtas norimo įrašyti failo nurodymui
UploadFile()	Metodas, skirtas pasirinkto failo įkėlimui į nutolusią saugyklą
AddNameToBD()	Metodas, skirtas įtraukti naujai įkelto failo vardui į saugomų failų duomenų bazę
ConfirmAdd()	Metodas, skirtas naujo failo pridėjimo patvirtinimui tiek

	vartotojui, tiek programos viduje
Klasė „Trynimas“, skirta ištrinti failą iš saugomų failų sąrašo ir taip pat iš nutolusios failų saugyklos	
DeleteSelected()	Metodas, skirtas ištrinti failą iš saugyklos, bei iš sąrašo
DeleteConfirm()	Metodas, skirtas ištrynimo patvirtinimui tiek vartotojui, tiek programos viduje
Klasė „Failo atidarymas“ skirta pasirinkto failo atidarymui kompiuteryje su tam failui reikalinga taikomąja programa, kuri instaliuota kompiuteryje	
OpenFile()	Metodas, skirtas atidaryti pasirinktą failą kompiuteryje

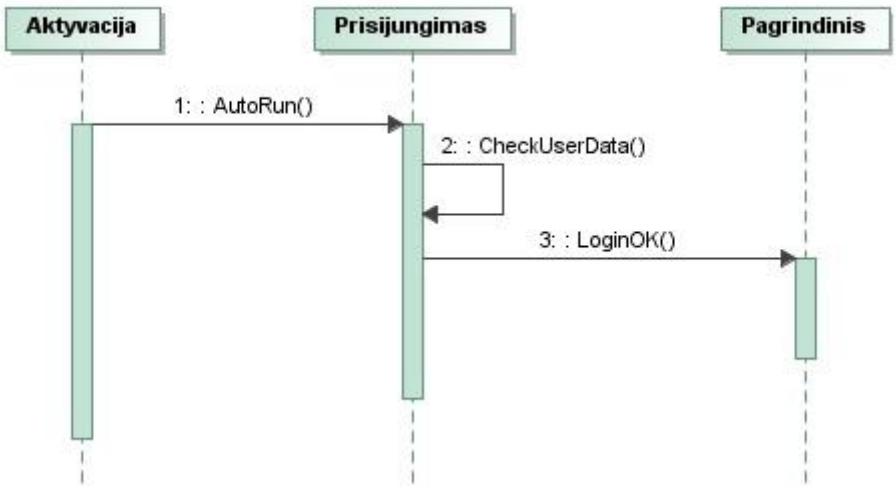
Klasių metodų aprašymai administratoriaus programos daliai pateikiami 22-je lentelėje.

22 lentelė: Klasių metodų aprašymas administratoriaus programos daliai

Metodas	Aprašymas
Klasė „Administration“, skirta valdyti tolimesnius veiksmus iš administravimo lango	
GetUserList()	Metodas, skirtas atvaizduoji administratoriui registruotų vartotojų sąrašą
EditUserInfoStart()	Metodas, skirtas aktyvuoti EdiUserInfo klasę
AddUserStart()	Metodas, skirtas aktyvuoti AddUser klasę
DelUserStart()	Metodas, skirtas aktyvuoti DelUser klasę
DisableUserStart	Metodas, skirtas aktyvuoti DisableUser klasę
Klasė „EditUserInfo“, skirta keisti vartotojo duomenims, jeigu jis pametė laikmeną, administratorius priskiria naujus prisijungimo duomenis	
SetNewUsername()	Metodas, skirtas pakeisti vartotojo prisijungimo vardą nauju
SetNewPassword()	Metodas, skirtas pakeisti vartotojo slaptažodį nauju
SaveUserInfo()	Metodas, skirtas išsaugoti pakeistą vartotojo informaciją
CheckExistingUserName()	Metodas, skirtas tikrinti ar naujai priskirtas prisijungimo vardas vartotojui nėra jau naudojamas
SaveUserFail()	Metodas, skirtas atšaukti vartotojo duomenų įrašymą ir pranešti administratoriui, kad toks vardas, kurį jis norėjo priskirti, jau egzistuoja
ConfirmEdit()	Metodas, skirtas patvirtinti, kad pakeitimai buvo atlikti

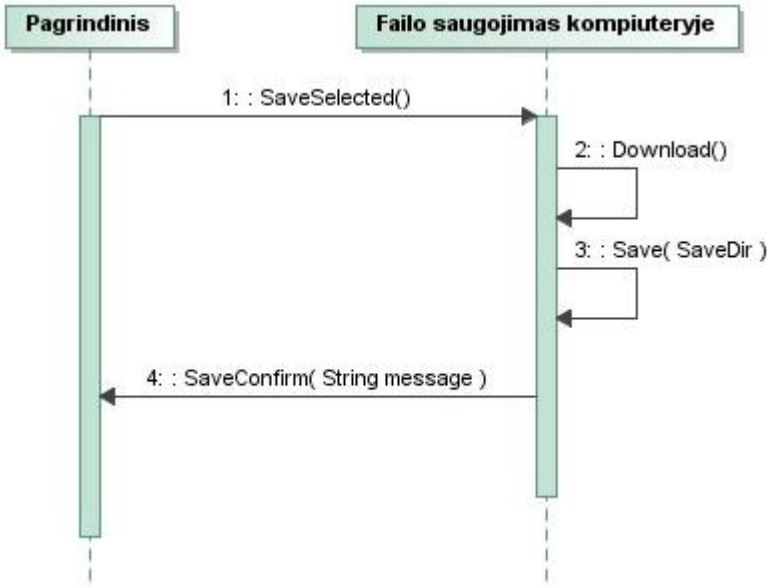
	tiek vartotojui, tiek programos viduje
Klasė „AddUser“, skirta naujo vartotojo sistemoje pridėjimui	
CheckExistingName()	Metodas, skirtas patikrinti ar įvestas vartotojo vardas nėra jau naudojamas sistemoje
AddFail()	Metodas, skirtas nutraukti naujo vartotojo pridėjimą ir administratoriui pranešti, kad toks vartotojo vardas jau egzistuoja sistemoje
AddNewUser()	Metodas, skirtas įrašyti naują vartotoją
ConfirmAdd()	Metodas, skirtas patvirtinti naujo vartotojo pridėjimą tiek administratoriui, tiek programos viduje
Klasė „DelUser“, skirta esamo vartotojo ištrynimui	
DeleteSelected()	Metodas, skirtas vartotojo ištrynimui
ConfirmDeletion()	Metodas, skirtas vartotojo ištrynimo patvirtinimui tiek administratoriui, tiek programos viduje
Klasė „DisableUser“, skirta išjungti vartotojo sąskaitą, ją padarant neaktyvią, „užšaldant“, taip pat ją galima padaryti vėl aktyvią	
DisableUser()	Metodas, skirtas padaryti vartotojo sąskaitą neaktyvią sistemoje
EnableUser()	Metodas, skirtas neaktyvią vartotojo sąskaitą sistemoje padaryti aktyvia
ConfirmAction()	Metodas, skirtas patvirtinti, kad pakeitimai buvo atlikti (sąskaita deaktyvuota ar aktyvuota) tiek administratoriui, tiek programos viduje

41 paveiksle pavaizduota vartotojo panaudojimo atvejo „Prisijungti“ sekų diagrama. Vartotojas įdeda laikmeną į kompiuterį, automatiškai aktyvuojasi programa ir iššoka prisijungimo langas. Jeigu vartotojas įveda teisingus prisijungimo duomenis, atsiranda pagrindinis programos langas.



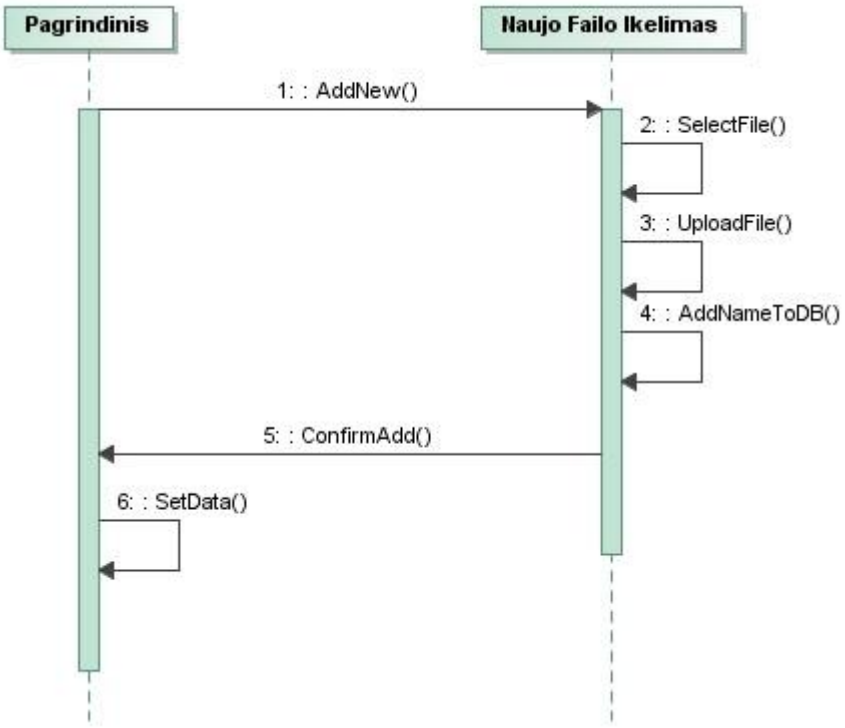
41 pav.: Vartotojo panaudojimo atvejo „Prisijungti“ sekų diagrama

42 paveikslėlyje vaizduojama vartotojo panaudojimo atvejo „Išsaugoti duomenis kompiuteryje“ sekų diagrama. Vartotojas pasirenka norimą išsaugoti failą, paspaudžia išsaugojimo mygtuką, atsidaręs *SaveDialog* langas leidžia pasirinkti direktoriją, kur norima išsaugoti ir failas išsaugomas.



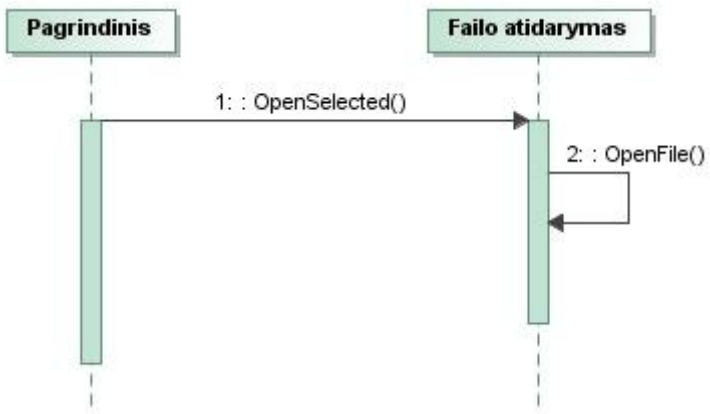
42 pav.: Vartotojo panaudojimo atvejo „Išsaugoti duomenis kompiuteryje“ sekų diagrama

43 paveikslėlyje vaizduojama vartotojo panaudojimo atvejo „Įrašyti naujus duomenis“ sekų diagrama. Vartotojas pasirenka įrašyti naują failą, paspaudžia naršymo mygtuką ir atsidaro *OpenDialog*. Vartotojas gali pasirinkti norimą failą, kuris įkeliamas į saugomų failų sąrašą ir įrašomas į nutolusią duomenų saugyklą.



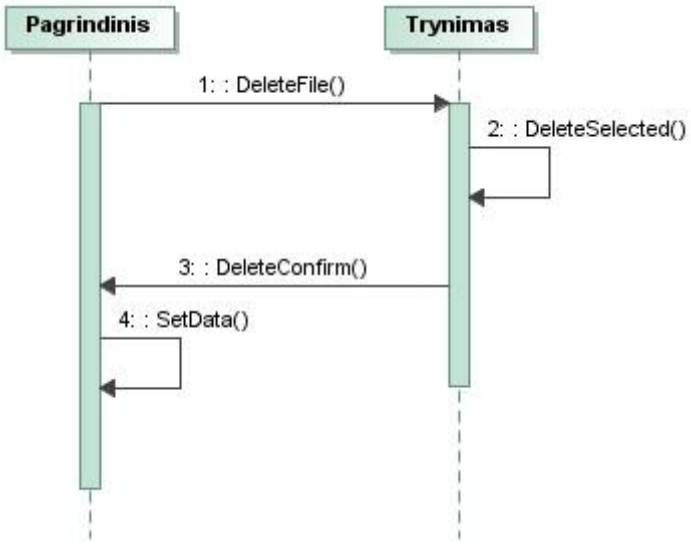
43 pav.: Vartotojo panaudojimo atvejo „Išsaugoti duomenis kompiuteryje“ sekų diagrama

44 paveikslėlyje vaizduojama vartotojo panaudojimo atvejo „Skaityti duomenis“ sekų diagrama. Vartotojas pasirenka failą iš sąrašo, paspaudžia atidaryti ir failas atsidaro, naudojant jam atidaryti skirtą taikomąją programą iš instaliuotų kompiuteryje.



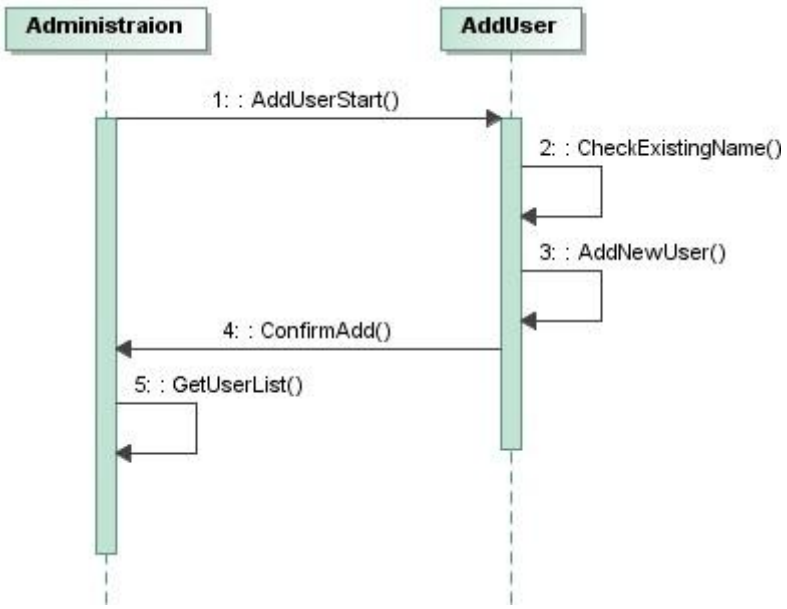
44 pav.: Vartotojo panaudojimo atvejo „Skaityti duomenis“ sekų diagrama:

45 paveikslėlyje vaizduojama vartotojo panaudojimo atvejo „Trinti duomenis“ sekų diagrama. Vartotojas iš sąrašo pasirenka nebereikalingą failą, ir paspaudžia ištrynimo mygtuką. Failas ištrinamas iš saugomų failų sąrašo ir iš nutolusios saugyklos



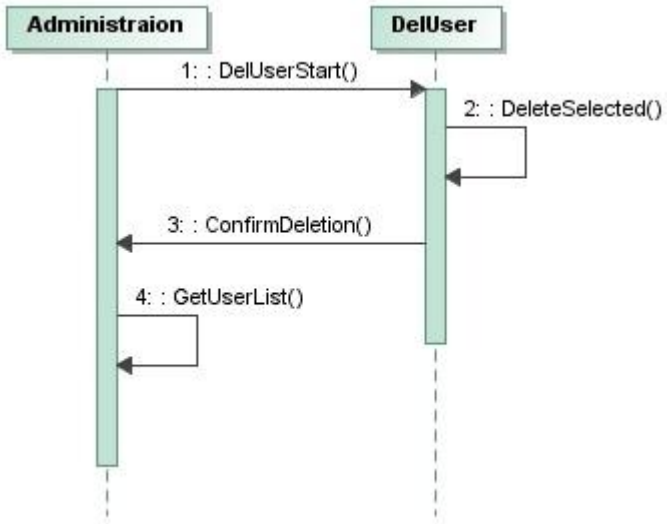
45 pav.: Vartotojo panaudojimo atvejo „Trinti duomenis“ sekų diagrama

46 paveikslėlyje vaizduojama administratoriaus panaudojimo atvejo „Pridėti vartotojus“ sekų diagrama. Administratorius pasirenka pridėti vartotoją, įveda jo duomenis ir išsaugo pakeitimus. Vartotojas atsiranda tarp jau esamų vartotojų sąraše



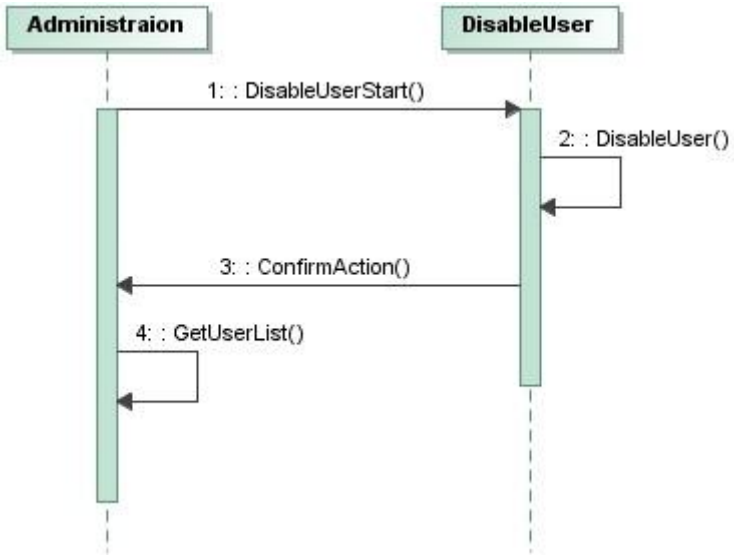
46 pav.: Administratoriaus panaudojimo atvejo „Pridėti vartotojus“ sekų diagrama

47 paveikslėlyje vaizduojama administratoriaus panaudojimo atvejo „Pašalinti vartotojus“ sekų diagrama. Administratorius pasirenka norimą pašalinti vartotoją, paspaudžia pašalinimo mygtuką ir vartotojas pašalinamas iš duomenų bazės.



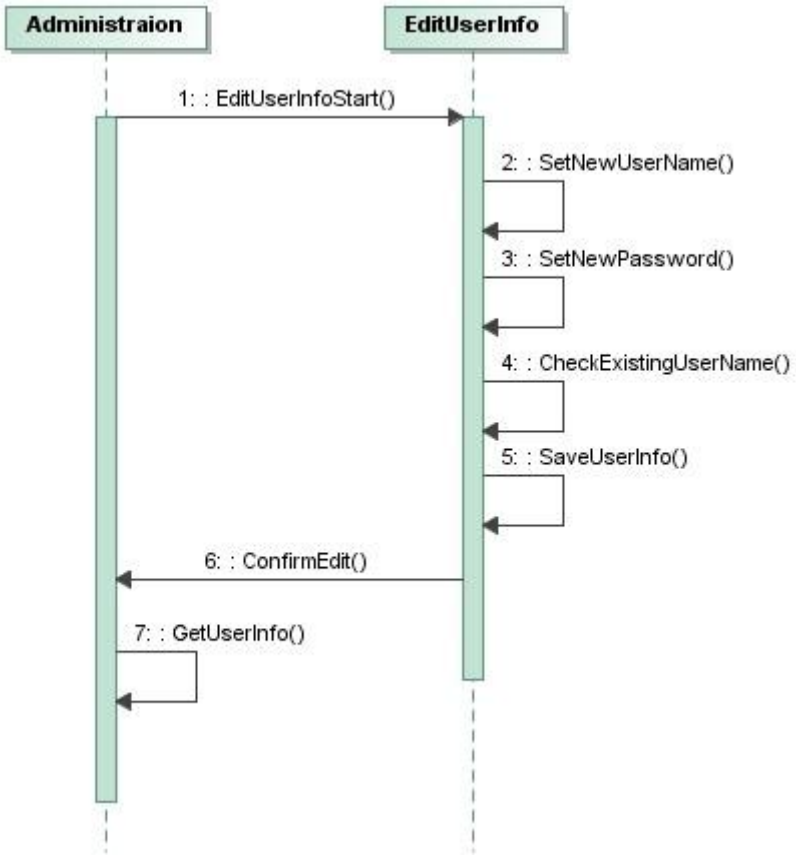
47 pav.: Administratoriaus panaudojimo atvejo „Pašalinti vartotojus“ sekų diagrama

48 paveikslėlyje vaizduojama administratoriaus panaudojimo atvejo „Laikinai įšaldyti vartotojo sąskaitą“ sekų diagrama. Administratorius gavęs pranešimą iš vartotojo, kad šis negali rasti savo laikmenos, kurioje yra saugoma prisijungimo prie nutolusios duomenų saugyklos programa, laikinai sustabdo vartotojo sąskaitos veikimą. Jis pasirenka iš sąrašo vartotojo vardą ir paspaudžia įšaldymo mygtuką.



48 pav.: Administratoriaus panaudojimo atvejo „Laikinai įšaldyti vartotojo sąskaitą“ sekų diagrama

49 paveikslėlyje vaizduojama administratoriaus panaudojimo atvejo „Priskirti naujus prisijungimo duomenis vartotojui“ sekų diagrama. Administratorius pasirenka vartotoją, kurio duomenis nori pakeisti, paspaudžia keitimo mygtuką ir atsiranda naujų duomenų įvedimo langas. Įvedęs duomenis administratorius išsaugo pakeitimus ir vartotojų sąraše rodomas vartotojas su naujais duomenimis, administratorius informuojamas kad pakeitimas buvo sėkmingas

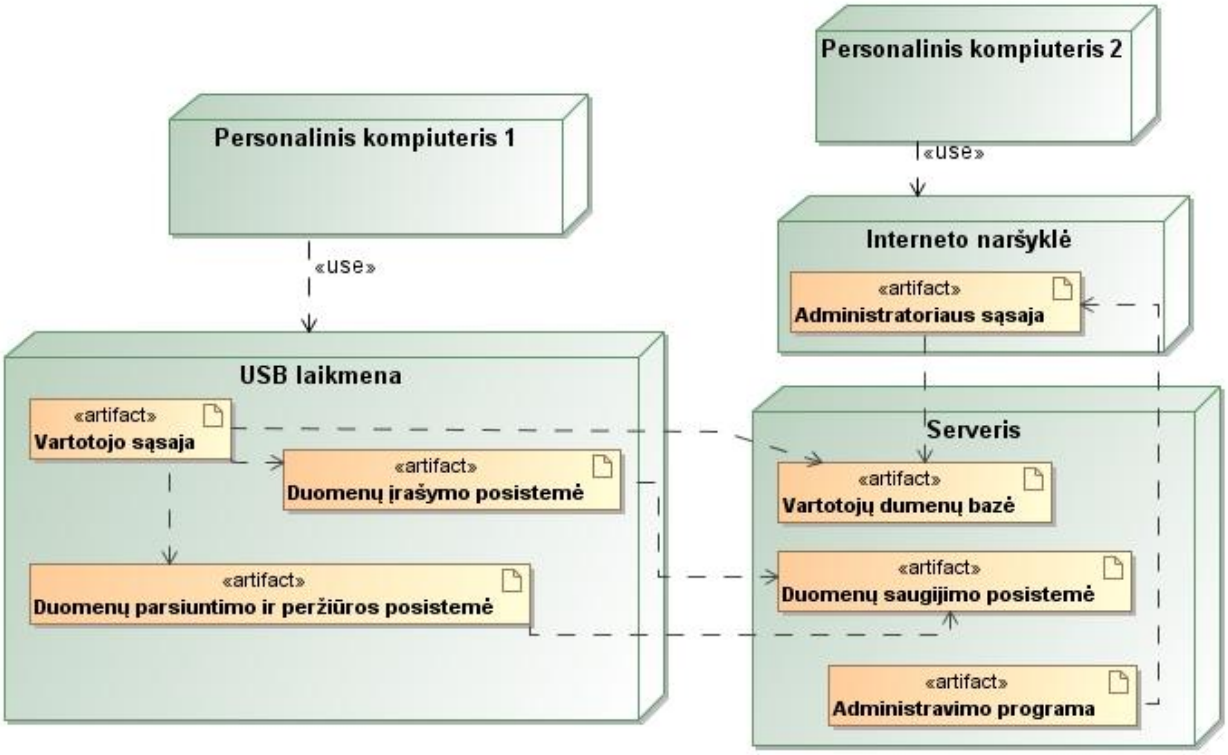


49 pav.: Administratoriaus panaudojimo atvejo „Priskirti naujus prisijungimo duomenis vartotojui“ sekų diagrama

4.5. Realizacijos modelis

4.5.1. **Programinių komponentų architektūra**

Sistemos realizacijos modelis pavaizduotas 50 paveiksle. Programai reikalingi 3 pagrindiniai komponentai, tai serveris, interneto naršyklė ir USB laikmena. Kaip papildomi komponentai yra 2 personaliniai kompiuteriai, nes be jų neveiktų USB laikmena ir administratoriaus sąsaja, kuri veikia interneto naršyklėje.



50 pav.: Sistemos realizacijos modelis

4.5.2. **Diegimo modelis**

Sistema susideda iš kelių pagrindinių posistemių: vartotojo sąsajos, duomenų įrašymo posistemės, duomenų parsiuntimo ir peržiūros posistemės, vartotojų duomenų bazės, duomenų saugojimo posistemės, administravimo posistemės ir administratoriaus vartotojo sąsajos.

Vartotojo sąsaja su duomenų įrašymo posisteme ir duomenų parsiuntimo ir peržiūros posisteme kaip viena programa įdiegiama į USB laikmeną, programa paleidžiama automatiškai įdėjus laikmeną į kompiuterį. Vartotojo sąsaja veikia su visomis Windows versijomis.

Vartotojų duomenų bazė, duomenų saugojimo posistemė bei administravimo programa įdiegiama serveryje, kad visi šie komponentai būtų pasiekiami per internetą. Duomenų bazė yra SQL tipo, o administravimo programa realizuojama PHP programavimo kalba.

Administratoriaus vartotojo sąsajos diegti nereikia, nes administravimo programos vartotojo sąsaja jau įdiegta serveryje ir pasiekiama per bet kurią interneto naršyklę.

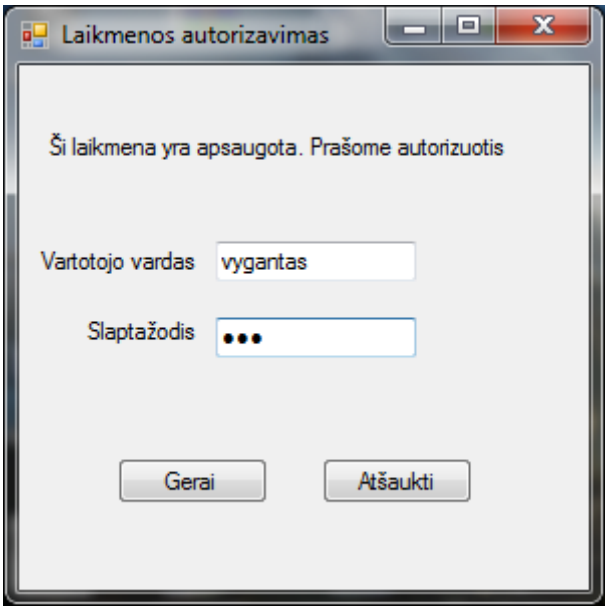
Norint naudotis USB laikmena bei interneto naršykle be abejo reikalingi kompiuteriai.

5. Realizacija

Šiame skyriuje apžvelgiama siūlomo sprendimo pagrindiniai langai ir funkcijos. Sukurtas sistemos prototipas, kuris leidžia atvaizduoti siūlomo sprendimo funkcijas bei veikimą.

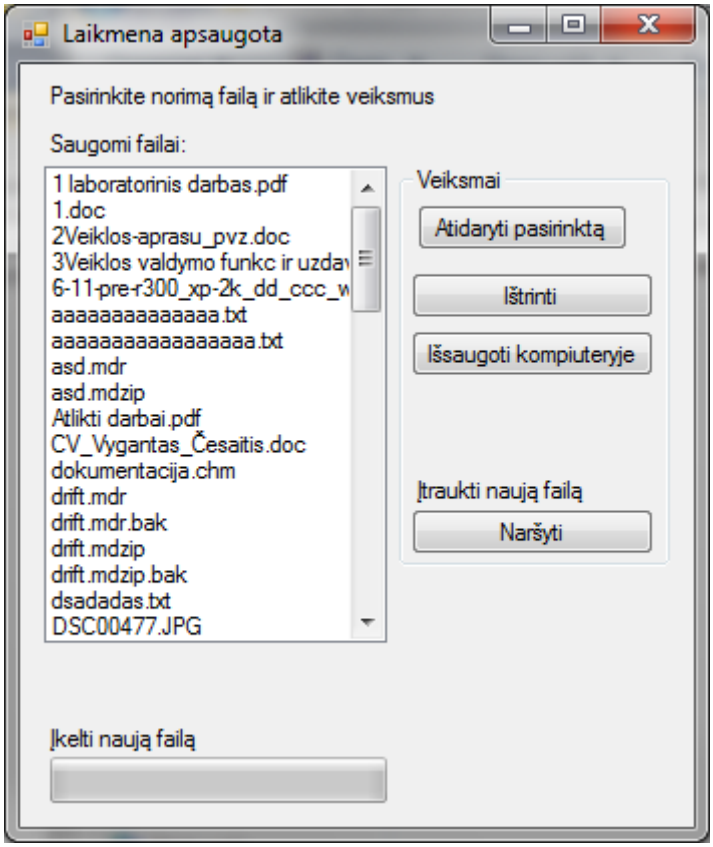
5.1. Realizacijos ir veikimo aprašymas

Pirmiausiai bus apžvelgiama vartotojui skirta sistemos dalis. Įdėjęs savo nešiojamą laikmeną į kompiuterį, vartotojas pirmiausiai pamato prisijungimui skirtą langą. Jame reikia įvesti prisijungimo vardą ir slaptažodį ir paspausti „Gerai“. Jeigu sistema nusprendžiama nesinaudoti, paspaudžiama „Atšaukti“ ir programos langas uždaromas. Prisijungimo langas vaizduojamas 51 paveikslėlyje.



51 pav.: Sistemos prisijungimo langas

Prisijungus, vartotojui pateikiamas jo saugomų failų sąrašas bei veiksmai, kuriuos vartotojas gali atlikti su saugomais failais. Vartotojas gali atidaryti iš sąrašo pasirinktą failą, ištrinti iš sąrašo pasirinktą failą, pasirinktą failą išsaugoti kompiuteryje, bei įrašyti naują pasirinktą failą iš kompiuterio. Vartotojo pagrindinis langas vaizduojamas 52 paveikslėlyje.



52 pav.: Pagrindinis vartotojo langas

Vartotojui pasirinkus failą iš sąrašo jis gali jį atidaryti paspaudęs mygtuką „Atidaryti pasirinktą“ failas atidaromas su jį atidaryti naudojama kompiuteryje įrašyta taikomąja programa, pavyzdžiui, jeigu failo galūnė .doc, jis bus atidaromas su dokumentų peržiūros ir redagavimo programa, tokia kaip *MS Word* arba *OpenOffice.org writer*.

Vartotojas gali ištrinti failą iš sąrašo, jį pasirinkęs ir paspaudęs mygtuką „Ištrinti“. Paspaudus failas bus ištrinamas tiek iš sąrašo, tiek ir iš nuotolinės saugyklos.

Paspaudęs mygtuką „Išsaugoti kompiuteryje“, vartotojas gali pasirinktą failą parsiųsti iš nuotolinės saugyklos ir išsaugoti jį savo pasirinktame aplanke. Kai failas parsiunčiamas ir išsaugojimas, programa praneša kad įrašymas buvo sėkmingas.

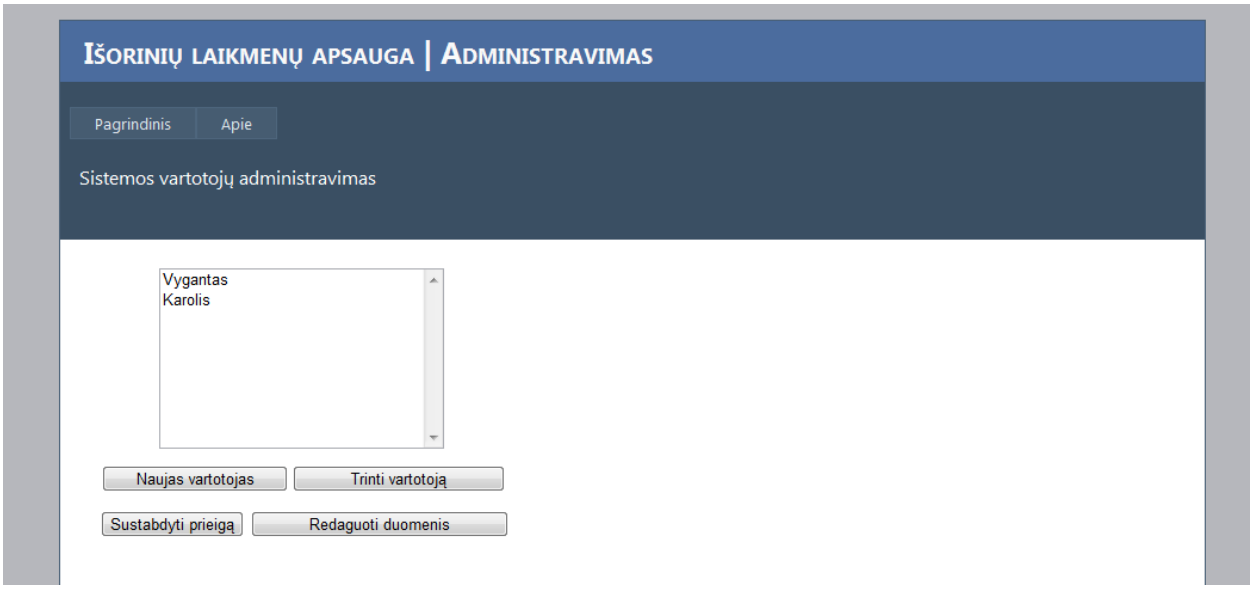
Taip pat vartotojas turi galimybę ir įkelti naują failą į nutolinę duomenų saugyklą. Paspaudęs mygtuką naršyti vartotojas išvysta *OpenDialog* sąsają, kur gali pasirinkti norimą failą iš bet kurio aplanko savo kompiuteryje. Failo įkėlimą į nuotolinę saugyklą rodo progreso juostelė, esanti programos pagrindinio lango

apačioje. Kad failo įkėlimas buvo sėkmingas vartotojas gali įsitikinti pamatęs jį saugomų failų sąraše.

53 paveikslėlis vaizduoja administratoriaus internetinės sąsajos prototipą. Prisijungęs administratorius mato registruotų vartotojų sąrašą. Jis gali sukurti naują vartotoją, ištrinti esamą vartotoją, sustabdyti prieigą ir redaguoti vartotojo duomenis.

Prieigos laikinas sustabdymas reikalingas, jeigu sistemos vartotojas pasigenda savo laikmenos ir nenori kad kas nors pasinaudotų duomenimis. Vartotojas kreipiasi į administratorių ir šis užblokuoja prieigą. Užblokavus prieigą, vartotojas gali ramiai ieškoti savo pamestos laikmenos. Jeigu ją randa, administratorius jo prieigą gali vėl atblokuoti, pažymėjęs vartotojo vardą, jeigu vartotojas užblokuotas, mygtukas „Sustabdyti prieigą“ pasikeičia į „Įgalinti prieigą“, administratoriui reikia tik paspausti mygtuką.

Administratorius gali keisti vartotojo prisijungimo duomenis. Tai reikalinga laikmenos praradimo atveju. Vartotojas nesuranda pamestos laikmenos, todėl kreipiasi į administratorių ir šis priskiria naujus prisijungimo duomenis. Priskirti naujus duomenis reikalinga, nes dauguma vartotojų prisijungimo duomenis gali būti netyčia paviešinę arba kur nors užsirašę, todėl trečiasis asmuo turėdamas prisijungimo programą gali pasinaudoti prieiga.



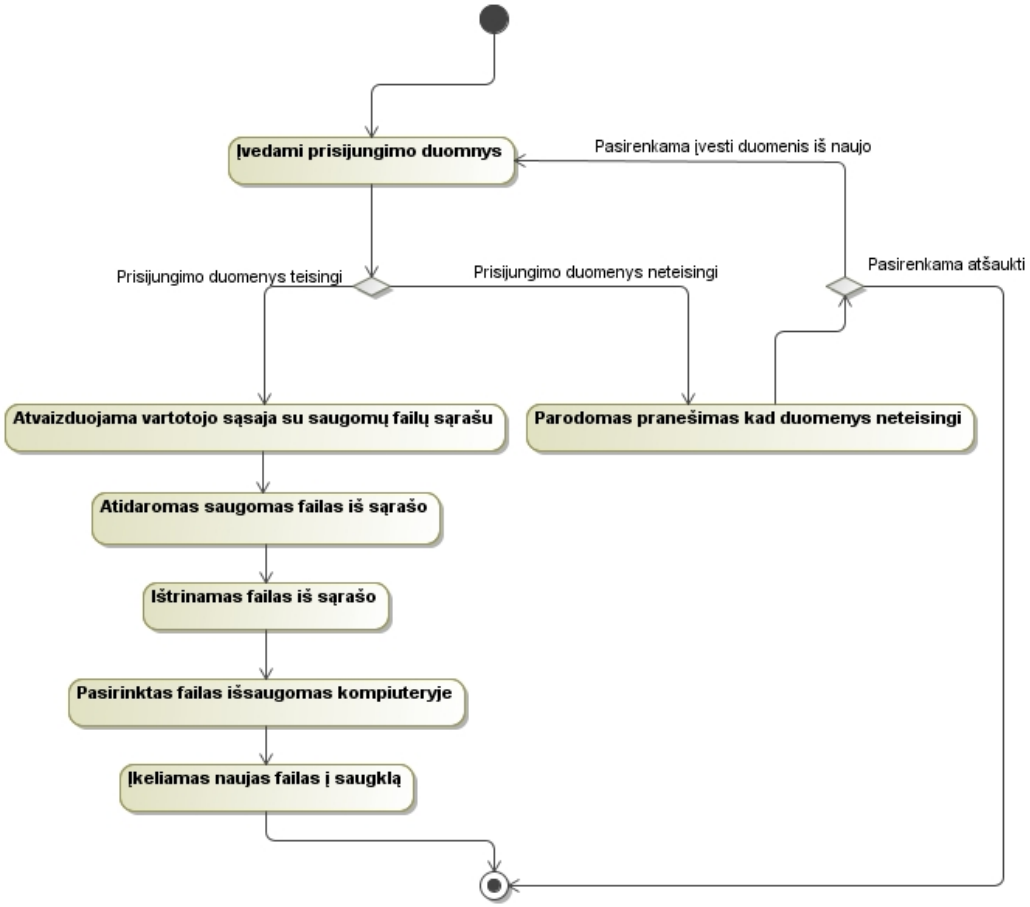
53 pav.: Pagrindinis administratoriaus langas

5.2. Testavimo modelis

Testuojant sistemą buvo naudojami įvairiausių tipų failai, nuo paprasčiausio tekstinio failo iki muzikos failų, filmų. Vartotojas gali prisijungti prie savo duomenų saugyklos tik naudodamas prisijungimui skirtą programinę įrangą ir teisingai įvedęs vartotojo vardą bei slaptažodį. Programinė įranga neskaičiuoja, keik kartų buvo įvesti neteisingi prisijungimo duomenys, dėl to yra galimybė pasinaudojus „*Brute-force*“ įsilaužimo metodą „atspėti“ prisijungimo duomenis. Tačiau tiek vartotojo vardas tiek slaptažodis administratoriaus gali būti įvesti iš daug skirtingų simbolių, be to, norint prisijungti prie duomenų saugyklos pirmiausiai reikia turėti tam skirtą programinę įrangą. Testavimas laikomas sėkmingu jeigu:

- Be klaidų startuoja prisijungimo prie vartotojo saugyklos posistemė;
- Pavyksta sklandžiai prisijungti;
- Įvedus klaidingus duomenis nepavyksta prisijungti;
- Pagrindinis vartotojo langas atvaizduoja vartotojo saugomus failus;
- Vartotojas gali atidaryti saugomą failą;
- Vartotojas gali ištrinti failą;
- Vartotojas gali išsaugoti failą kompiuteryje, bei jį atidaryti nuėjęs į tą katalogą, kuriame failas buvo išsaugotas;
- Įkėlus failą į saugyklą, failas rodomas saugomų failų sąraše.

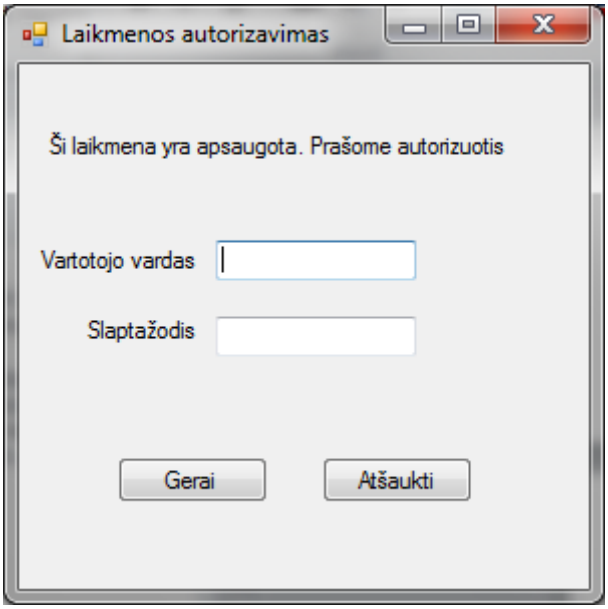
Sistemos testavimo modelis pateikiamas 54 paveikslėlyje.



54 pav.: Sistemos testavimo modelis

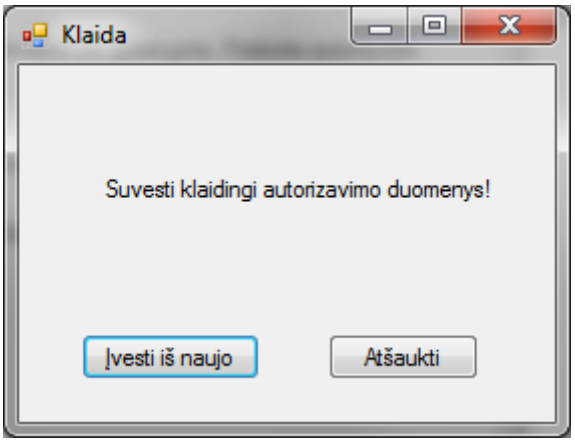
5.3. Testavimo duomenys ir rezultatai

Testavimas vykdomas pagal testavimo modelį. Paleidžiama vartotojo dalies programa. Tai parodoma 55 paveikslėlyje.



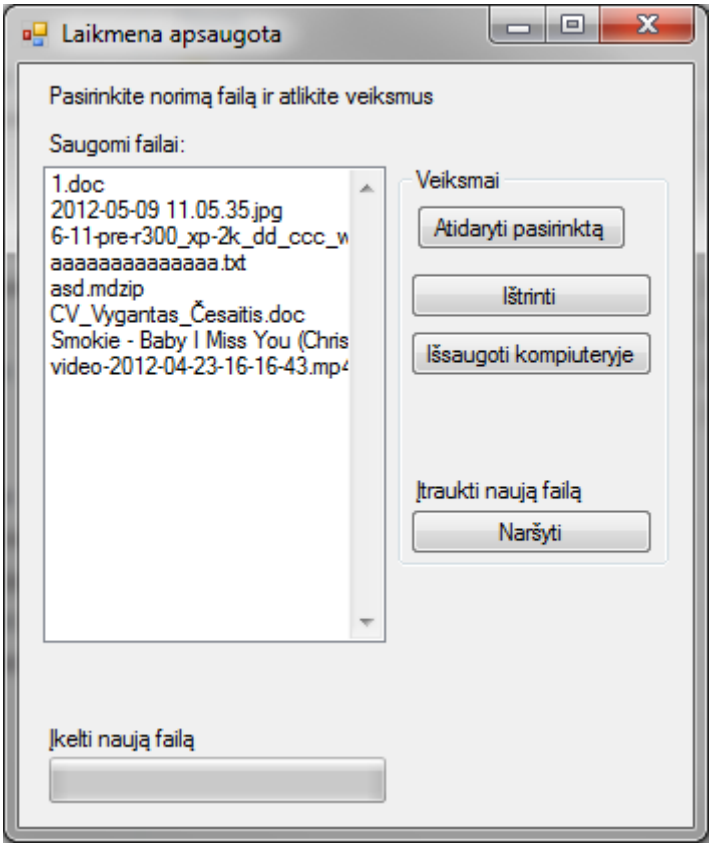
55 pav.: Sistemos prisijungimo langas

Toliau, norint patikrinti ar programa tikrai veikia tik su teisingu slaptažodžiu, įvedami klaidingi prisijungimo duomenys, rezultatas matomas paveikslėlyje 56. Paspaudus „Įvesti iš naujo“, grįžtama į pradinį prisijungimo langą, kur galima pakartotinai įvesti prisijungimo duomenis. Jei pasirenkama „Atšaukti“, programa užsidaro.



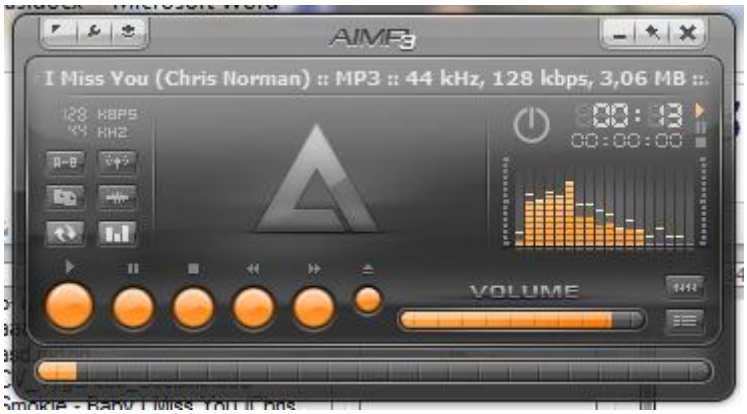
56 pav.: Pranešimas apie klaidingai suvestus duomenis

Sekančiame žingsnyje įvedami teisingi prisijungimo duomenys. Atsidaro vartotojo pagrindinis langas, kuriame yra saugomų failų sąrašas bei veiksmai, kuriuos gali atlikti vartotojas. Rezultatas vaizduojamas 57 paveikslėlyje.



57 pav.: Pagrindinis vartotojo langas

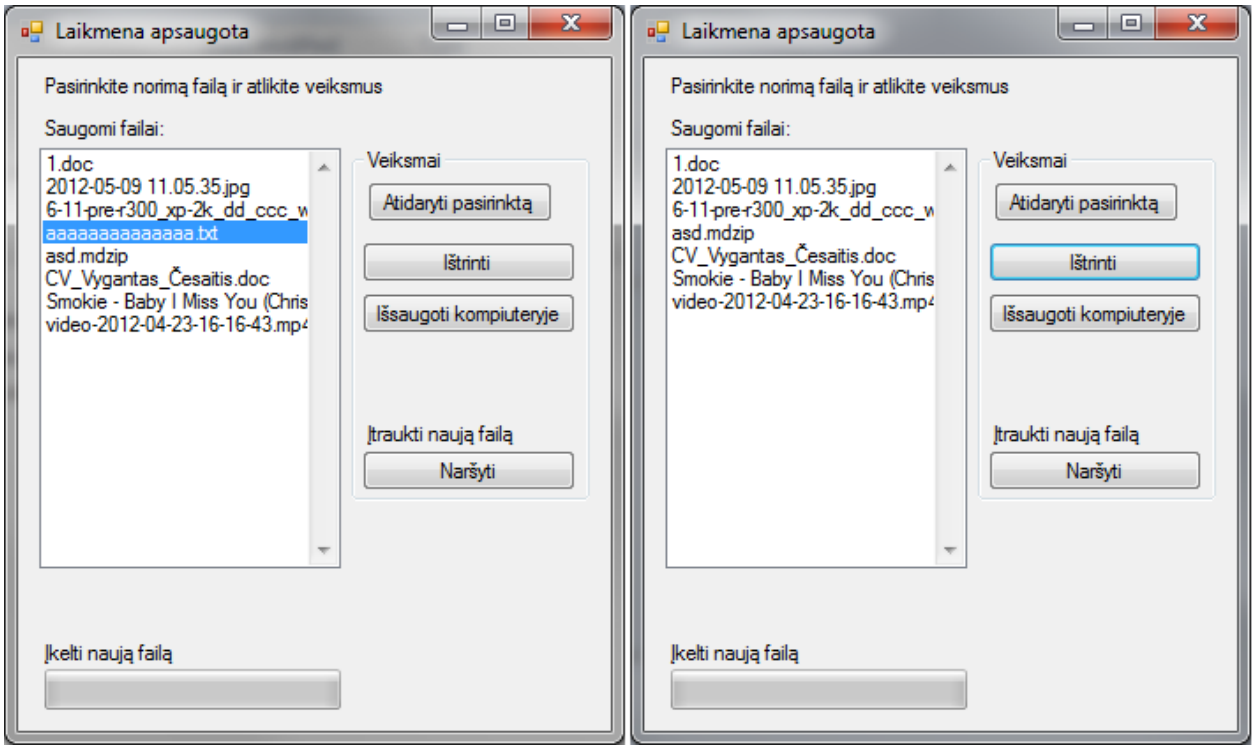
Toliau pasirenkamas priešpaskutinis failas, esantis sąraše „*Smokie - Baby I Miss You (Chris Norman).mp3*“ ir paspaudžiama atidaryti. Kadangi skaityti .mp3 audio failams kompiuteryje kaip programa pagal nutylėjimą (*default*) yra AIMP3, failas paleidžiamas naudojant ją. Rezultatas matomas 58 paveikslėlyje.



58 pav.: Grojamas iš vartotojo pagrindinio programos lange esančio sąrašo pasirinktas audio failas

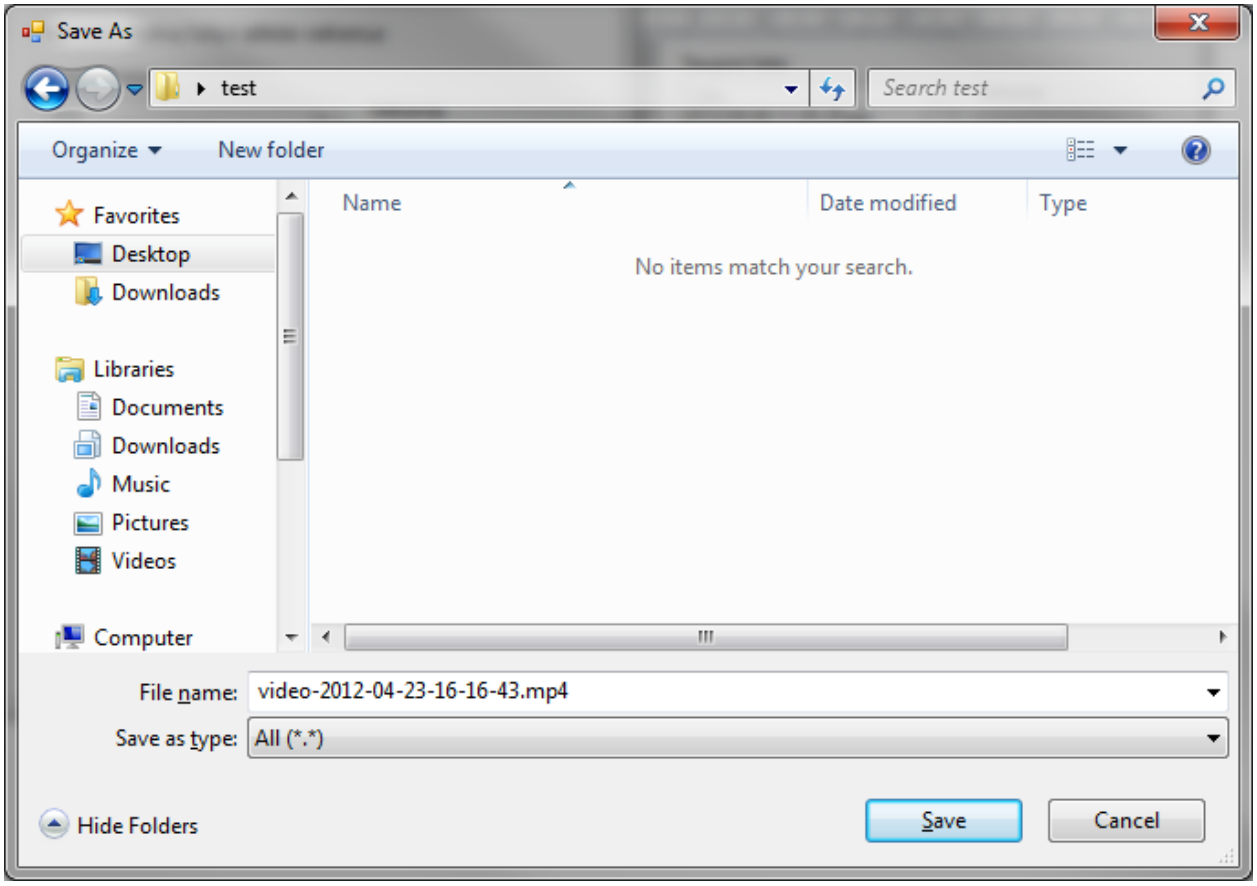
Sekančiame etape testuojamas trynimas. Pasirenkamas failas „*aaaaaaaaaaaaa.txt*“ ir paspaudžiama „Ištrinti“. Pasirinktas failas ištrinamas ir

dingsta iš sąrašo. Rezultatas matomas palyginus programos languose esančius failų sąrašus prieš atliekant trynimą ir po. Kairiame lange pažymėtas trinti paruoštas failas, o dešiniame lange jo jau nėra. Rezultatas pateikiamas paveikslėlyje 59.

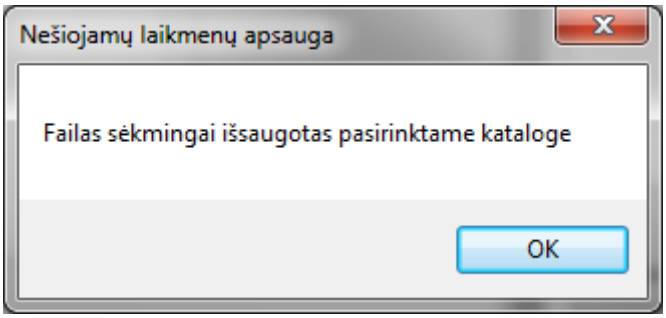


59 pav.: Vartotojo langų palyginimas prieš ir po trynimo

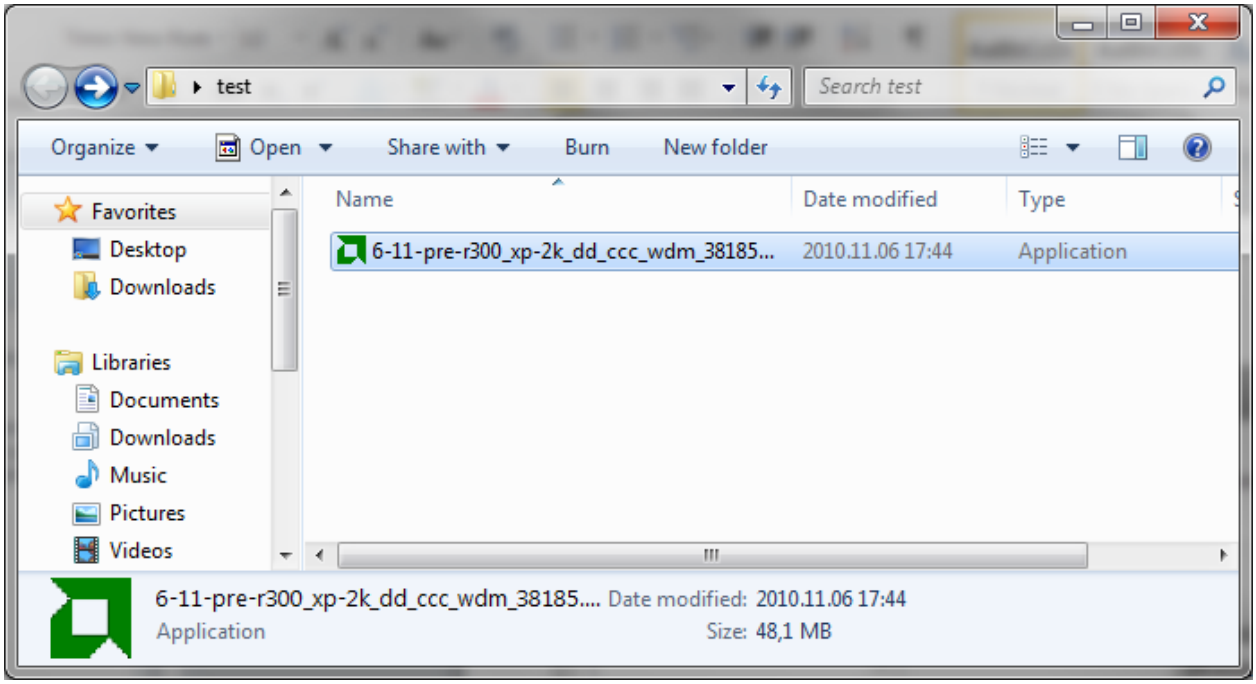
Toliau testuojamas išsaugojimas kompiuteryje. Pasirenkamas failas „*6-11-pre-r300_xp-2k_dd_ccc_wdm_38185.exe*“ ir paspaudžiama „Išsaugoti kompiuteryje“. Atsiranda *SaveDialog* ir prašoma nurodyti katalogą į kurį išsaugoti. Pasirenkamas katalogas „*test*“, kuris yra vartotojo darbalaukyje (paveikslėlis 60). Gaunamas pranešimas, kad failas sėkmingai išsaugotas nurodytame kataloge (paveikslėlis 61). Atidaromas katalogas, į kurį buvo išsaugota, matomas išsaugotas failas (paveikslėlis 62). Failas atidaromas, jis veikia.



60 pav.: SaveDialog langas failui išsaugoti

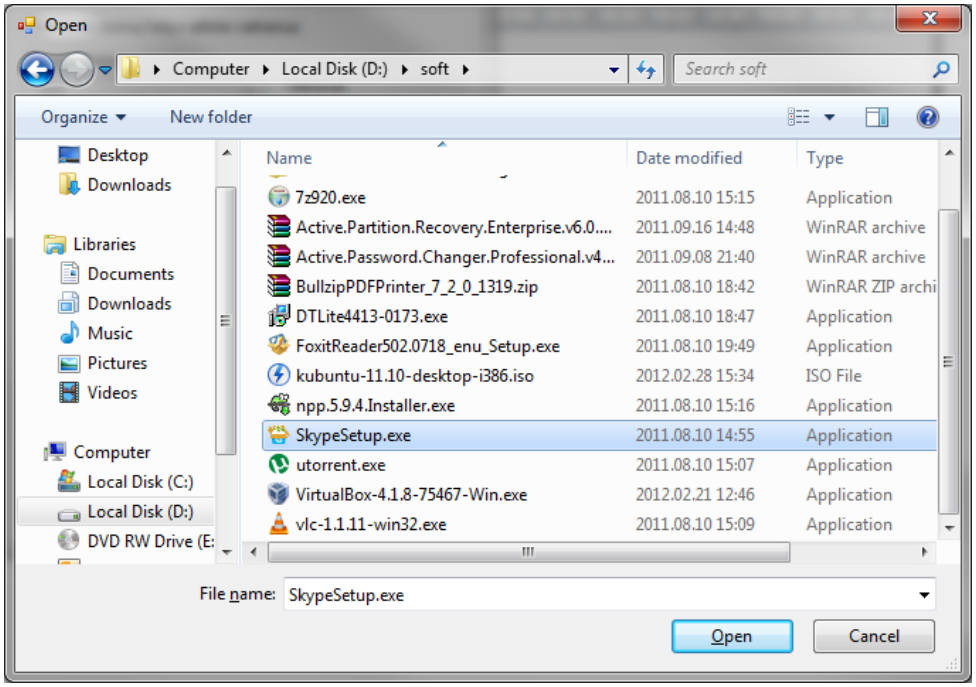


61 pav.: Pranešimas apie sėkmingai išsaugotą failą

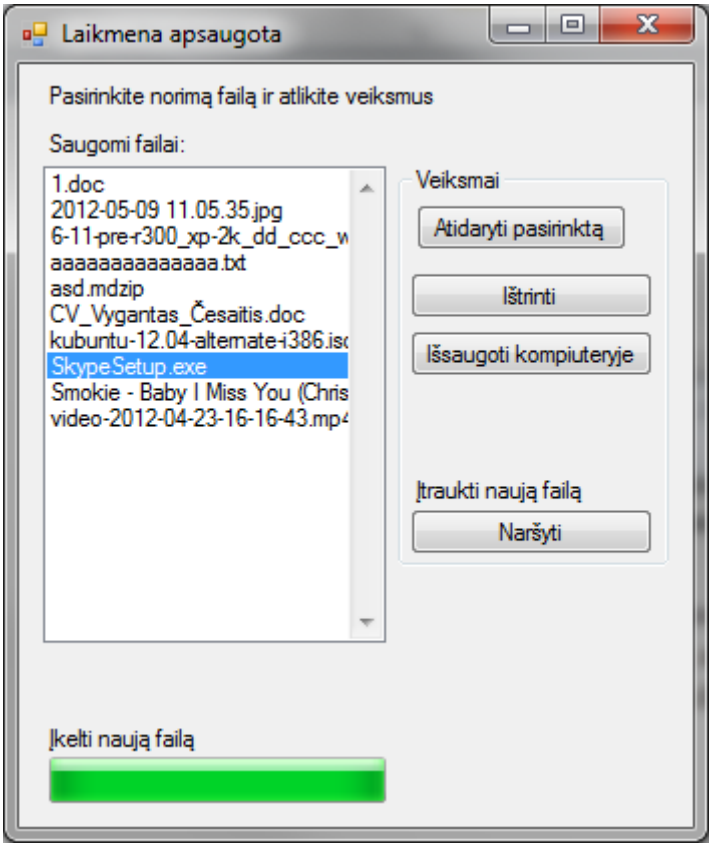


62 pav.: Failas išsaugotas pasirinktame kataloge

Paskutiniame testavimo etape testuojamas failų įkėlimas į saugyklą. Iš pagrindinio vartotojo lango paspaudus mygtuką „Naršyti“, atsiveria *OpenDialog* langas (paveikslėlis 63), leidžiantis pasirinkti norimą failą iš bet kurios direktorijos kompiuteryje. Pasirenkamas failas „*SkypeSetup.exe*“. Failas nukopijuojamas į saugyklą ir atsiranda saugomų failų sąraše (paveikslėlis 64). Kopijavimo eigą rodo progreso juostelė.



63 pav.: OpenDialog pasirenkamas failas



64 pav.: Naujai įdėtas failas rodomas sąraše

Atliktas testavimas parodė, kad sistema veikia korektiškai ir atitinka visus iš anksto iškeltus reikalavimus.

6. Eksperimentinis sistemos tyrimas

Reikia ne tik tiksliai apibrėžti pagal kokius kriterijus bus vertinama programos kokybė ir pagal juos įvertinti, bet įvertinti ir pagal paprasčiausius vartotojų iškeliamus programinės įrangos kriterijus. Tada bus galima spręsti apie programinės įrangos kokybę. Siekiant atlikti įvertinimą buvo iškeltas tikslas – pamatuoti programinės įrangos kokybę pagal paprasčiausio vartotojo poreikius ir suprantamumą.

6.1. Eksperimento planas

Sukurtas sistemos prototipas, skirtas ne tik apsaugoti duomenis nuo nesankcionuoto peržiūrėjimo bei naudojimo, bet ir apsaugoti juos nuo praradimo laikmenos praradimo atveju. Atsižvelgiant į faktą, kad vartotojams yra svarbus ne tik failų saugumas nuo trečiųjų asmenų peržiūros, bet nemažiau svarbus yra ir faktas kad duomenys nebūtų prarasti, buvo sukurtas prototipas, kuris duomenis talpina nuotolinėje saugykloje, o nuo peržiūros saugo vartotojų identifikacija. Vartotojas turi

laikmeną su prisijungimui ir failų peržiūrai reikalinga programine įranga, o vartotojus ir jų prisijungimo duomenis valdo administratorius per interneto naršykle pasiekiamą sąsają. Kadangi sistemos prototipas yra labiau orientuotas vartotojui, eksperimentas su administratoriaus sistemos dalimi nebus atliekamas.

Atliekant eksperimentą bus vertinamas vienas iš svarbiausių programinės įrangos faktorių – tai jos paprastumas ir naudojimo intuityvumas, kai ja naudojasi paprastas vartotojas. Vartotojui svarbiausia yra tai, kad programinė įranga veiktų sklandžiai ir nereikėtų papildomų informacijos šaltinių, norint ją naudoti.

Eksperimento metu, programinės įrangos vartotojo dalis bus įdiegta į 30 valstybinės įstaigos vartotojų kompiuterių. Bus pateikiamas pagrindinių funkcijų aprašymas ir nurodoma patikrinti sistemos funkcijas su 10 skirtingų tipų failų, kurie iš anksto pateikti vartotojams. Žinant darbuotojų užimtumą bus nustatytas savaitės laiko testavimo terminas, o po to pateikta įvertinimo anketa.

6.2. Eksperimento rezultatai

Eksperimentas buvo atliktas valstybinėje biudžetinėje įstaigoje į vartotojų kompiuterius įrašant vartotojo programinę dalį. Vartotojai programinę įrangą testavo savaitę laiko su visiems vienodais iš anksto atrinktais duomenimis. Buvo pateiktas pagrindinių programos funkcijų aprašymas kiekvienam iš vartotojų.

Siekiant sudaryti kuo vienodesnes sąlygas, kurios nepriklauso nuo vartotojo, programinė įranga buvo įrašyta tik į tuos kompiuterius, kuriuose buvo operacinė sistema Microsoft Windows 7 ir buvo ne mažiau nei 2 GB operatyviosios atminties. Praėjus eksperimento laikui, kiekvienas programinę įrangą naudojęs vartotojas gavo nedidelę anketą, kurioje dešimtbalėje sistemoje įvertino programinės įrangos funkcionalumą ir paprastumą pagal iš anksto kartu su pačiais vartotojais susikurtus kriterijus. Vertinimo kriterijai buvo šie:

1. Programos paleidimo paprastumas;
2. Programos stabilumas (kuo daugiau kartų nustojo veikti, tuo balas mažesnis);
3. Paprastumas prisijungti prie sistemos;
4. Intuityvumas (kuo daugiau kartų reikėjo skaityti aprašą, tuo mažesnis balas, jei užteko perskaityti vieną kartą – balas 9, jei neprireikė skaityti – 10);
5. Sritys kur gali būti pritaikoma (kuo daugiau, tuo didesnis balas);

6. Programos dialogų ir langų suprantamumas;
7. Programos funkcijų gausa duomenų valdymo srityje;
8. Programos teksto šriftas ir jo dydžio tinkamumas;
9. Duomenų išrinkimo iš sąrašo patogumas;
10. Programos funkcijų reikalingumas;

Vartotojų įvertinimai pagal kriterijus pateikiami 23 lentelėje. Kriterijai sunumeruoti pagal aukščiau pateiktą sąrašą, jie yra lentelės viršuje, lentelės kairėje pusėje pateikiami vartotojai, kadangi anketa buvo anoniminė, vartotojai pateikiami numeriais.

23 lentelė: Atliktos apklausos suvestinė

	Kriterijai									
Vartotojai	1	2	3	4	5	6	7	8	9	10
1	10	10	10	9	8	10	8	9	8	10
2	10	10	10	9	8	10	8	9	8	10
3	9	10	9	9	9	10	7	8	8	10
4	10	10	10	9	8	9	8	8	9	10
5	10	10	10	10	9	10	8	8	10	10
6	10	8	10	9	9	9	8	8	9	10
7	10	7	10	9	9	9	8	8	9	10
8	10	8	10	9	8	9	8	8	8	10
9	10	8	10	9	8	9	7	8	9	10
10	10	9	10	9	9	9	7	8	8	10
11	8	10	9	9	8	9	7	8	8	10
12	9	9	9	10	9	9	8	8	8	10
13	9	8	10	8	8	8	9	8	8	10
14	9	10	10	9	9	10	9	8	9	10
15	9	10	10	9	9	10	8	8	9	9
16	10	10	10	8	8	8	9	9	8	9
17	9	10	19	9	9	9	8	9	9	10
18	10	10	10	9	9	9	8	9	9	9
19	9	10	9	8	9	9	8	9	9	9
20	10	10	10	9	8	9	8	9	8	10

21	10	10	10	10	10	10	7	10	9	10
22	10	10	10	10	9	9	7	8	9	10
23	9	10	10	9	8	9	7	8	8	10
24	10	10	10	10	9	10	8	8	9	10
25	10	10	10	10	10	10	8	7	8	10
26	10	10	10	10	10	10	8	7	8	10
27	10	9	10	9	9	9	8	9	9	10
28	10	9	9	9	8	9	9	8	9	10
29	9	9	9	8	9	9	9	8	9	10
30	9	9	10	9	9	9	9	7	8	10
Vidurkis	9,60	9,43	9,80	9,10	8,73	9,27	7,97	8,23	8,57	9,87

Iš bendrų kriterijų įvertinimo vidurkių matoma, kuriose vietose programinė įranga turi silpniausias dalis ir kurios dalys yra stipriausios pagal vartotojus.

Apibendrinant rezultatus galima teigti, kad programinei įrangai reikėtų pridėti dar kelias funkcijas, kurias kuriant reikėtų konsultuotis su vartotojais. Taip, pat konsultuojantis reikėtų pakeisti šriftą ir jo dydį, kad programa būtų patogesnė, bei padidinti sąrašo lauko plotį arba įgalinti paslinkimą, kad būtų matomas pilnas failo pavadinimas.

Visos kiti kriterijai gavo labai aukštus įvertinimus, todėl jų tobulinti nereikia, tose vietose programinė įranga yra patogi vartotojui.

6.3. Sistemos veikimo ir savybių analizė, kokybės kriterijų įvertinimas

Programinė įranga bus vertinama pagal anksčiau atlikto programos testavimo su paprastais vartotojais eksperimento gautus duomenis. Vykdant eksperimentą, išvardinti 10 kriterijų buvo parinkti ne atsitiktinai, jie buvo sukurti konsultuojantis su tais pačiais vartotojais. Buvo priimta, kad šie kriterijai programinės įrangos vertinime yra lygiaverčiai.

Vertinant kiekvieną programinę įrangą, individualiai pagal ją, nusprendžiama kiek ir kokioje balų sistemoje programinė įranga turi surinkti, kad ji skaitytųsi kokybiška. Atsižvelgus į kai kurių apklausą atlikusių vartotojų pastabas ir į programinės įrangos nedidelį sudėtingumą, buvo nuspręsta, kad bus naudojamasi ta

pačia dešimtbale sistema ir jeigu programinė įranga surinks daugiau nei 8,5 balo ji bus laikoma kokybiška.

Kadangi buvo priimta, kad visi kriterijai šiai programinei įrangai yra lygiaverčiai, tai belieka tik išvesti visos aukščiau pateiktos 23 lentelės vidurkį.

Gautas vidurkis, suapvalinus iki dviejų skaičių po kablelio yra 9,06 balo. Galime teigti, kad programinė įranga yra kokybiška.

6.4. Sistemos taikymo rekomendacijos

Sukurtas sistemos prototipas - vartotojo programinės įrangos dalis, kuri valdo failų saugojimą nuotolinėje saugykloje. Pilnai realizavus visą sistemą, ja galėtų naudotis tiek privatūs asmenys, tiek įmonių bei įstaigų darbuotojai, visi žmonės, kuriems rūpi ne tik duomenų saugumas nuo nesankcionuoto naudojimo, bet ir tai, kad tie duomenys nebūtų prarasti paprasčiausiai pametus laikmeną.

Sistema veikia su visomis šiuo metu *Microsoft* palaikomomis *Windows* versijomis. Jokios papildomos programinės įrangos instaliuoti kompiuteryje tiek vartotojui tiek administratoriui nereikia. Administratoriaus programinė įranga yra patalpinta serveryje ir prieinama per bet kurią interneto naršyklę, o vartotojo programinė įranga patalpinta pernešamoje laikmenoje. Norint įrašyti programą į laikmeną, užtenka įkelti *autorun.inf* bei programos paleidžiamąjį failą į išorinę laikmeną. Rašant į kai kurias laikmenas, gali prireikti įrašyti dar kelis papildomus darbinius programos failus.

Sistema veikia su USB atmintinėmis, išoriniais kietaisiais diskais, CD\DVD diskais bei su visų tipų kortelėmis, kitaip tariant su visomis išorinėmis laikmenomis, kurias tik gali perskaityti kompiuteris.

7. Išvados

1. Atlikus jau esančių sprendimų analizę pastebėta, jog daugelis šiuo metu egzistuojančių sprendimų pasikliauja vienu apsaugos tipu arba saugesnė sistema yra daug sudėtingesnė ir brangesnė. Iš vartotojų analizės nustatėme, jog nei viena sistema netenkina visų vartotojų grupių poreikių. Paprastiems vartotojams reikia paprastesnės sistemos, už kurią nereikėtų mokėti daug arba ji būtų nemokama, tuo tarpu verslo klientai yra labiau linkę sumokėti daugiau už papildomas funkcijas ir saugumo garantijas.
2. Analizės metu pastebėta, jog sudėtingesni ir tuo pačiu saugesni sprendimai reikalauja ne tik papildomos programinės įrangos įdiegimo į kiekvieną kompiuterį, su kuriuo būtų naudoja USB atmintinė, bet ir specialių atmintinių su techninės įrangos saugumo priedais.
3. Išanalizavus šiuo metu populiariausius šifravimo algoritmus ir šifravimo būdus buvo išsiaiškinta, kad pats patogiausias algoritmu būtų - AES (*Rijndeal*). Jis pasižymi aukštu saugumo lygiu ir yra greitesnis, nei jo artimiausias konkurentas - *Blowfish*. Kita vertus – siekiant sukurti maksimaliai saugią sistemą *Blowfish* būtų tinkamesnis variantas. Taip pat norint užtikrinti užšifruotų duomenų validumą ir integralumą patartina naudoti kontrolinės maišos (parašo) algoritmus.
4. Atlikus eksperimentą paaiškėjo, jog paprastiems vartotojams daug aktualiau buvo jo naudojimo paprastumas ir programos vartotojo sąsajos intuityvumas. Apklausti vartotojai dažiau atkreipė dėmesį ne į programos funkcijas ar joje panaudotas technologijas saugumui užtikrinti, o į naudojimo patogumą.

8. Literatūra

1. Brian Carrier (2005). „File System Forensic Analysis“. Addison Wesley Professional.
2. John R. Douceur, William J. Bolosky (1999). „A Large-Scale Study of File-System Contents“. Microsoft Reseach, Redmond.
3. Romik Guha Anjoy, Soumya Kanti Chakraborty (2009). „Feature Based Comparison of Modern File Systems“. Mälardalen University, Sweeden.
4. Jeff Hamm (2009). „Extended FAT File System“. Paradigm Solutions, Computer Investigations and Forensics.
5. Mitsuru Matsui (1994). “The First Experimental Cryptanalysis of the Data Encryption Standard”.
6. Jerome Burke, John McDonald, Todd Austin (2000). „Architectural Support for Fast Symmetric-Key Cryptography“. Advanced Computer Architecture Laboratory, University of Michigan.
7. J. Katz; Y. Lindell (2007). „Introduction to Modern Cryptography“.
8. Anoop MS (2007). „Public key Cryptography - Applications Algorithms and Mathematical Explanations“.
9. Bart Preneel (1993). „Cryptographic Hash Functions“. Katholieke Universiteit Leuven, Laboratorium ESAT-COSIC.
10. N. Ferguson, B. Schneier (2003). „Practical Cryptography“.
11. Rūta Makūnaitė (2006). „Nematoma informacija“, [žiūrėta 2012-04-15], prieiga per internetą: http://www.elektronika.lt/straipsniai/kompiuterija/5717/nematoma-informacija/ Lietuvos Rytas, Kompiuterija – PC World
12. Stephan Katzenbeissera, Fabien Petitolas (2000). „Information Hiding Techniques for Steganography and Digital Watermaking“.
13. Joan Daemen, Vincent Rijmen (2002). „The Design of Rijndael: AES – The Advanced Encryption Standard“. Springer-Verlag.
14. Data Encryption Standard (DES), [žiūrėta 2012-04-22], prieiga per internetą: http://www-math.ucdenver.edu/~wcherowi/courses/m5410/des.pdf
15. Ross Anderson, Eli Biham, Lars Knudsen. „Serpent: A Proposal for the Advanced Encryption Standard“.
16. K. Gaj, P. Chodowiec (2000). „Implementations of the AES Candidate Algorithms using FPGA devices“. Technical Report, George Mason University.
17. R. Pereira, G. Carter (1997). „The ESP CAST128-CBC Algorithm“. Internet Engineering Task Force, IP Security Working Group.
18. C. Adams (1997). „The CAST-128 Encryption Algorithm“. Network Working Group
19. B. Schneier. „Blowfish Source Code“, [žiūrėta 2012-04-19], prieiga per internetą: http://www.schneier.com/blowfish-download.html

20. B. Schneier (1994). „Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish)“. Fast Software Encryption, Cambridge Security Workshop Proceedings.
21. Rivest R. (1992). „The MD5 Message-Digest Algorithm“, MIT and RSA Data Security, Inc.
22. Klima V. (2006). „Tunnels in Hash Functions: MD5 Collisions Within a Minute“.
23. Genova Z. (2002). „Efficient summarization of URLs using CRC32 for implementing URL switching“.
24. Rivest R. (1991). „The MD4 message digest algorithm“. Advances in Cryptology - CRYPTO '90 Proceedings, Springer-Verlag.
25. Rivest R. (1990). „The MD4 Message Digest Algorithm“. MIT.
26. Christophe De Canniere, Christian Rechberger (2006). „Finding SHA-1 characteristics: General results and applications“. Graz University of Technology, Institute for Applied Information Processing and Communications.
27. Xiaoyun Wang, Andrew C Yao, Frances Yao (2005). „Cryptanalysis on SHA-1“. Tsinghua University & Shandong University & City University of Hong Kong.

9. Priedai

1 priedas. Apklausos anketa

Išorinių atmintinių naudojimas. Apklausos anketa

1. Kiek kartų per savaitę naudojatės išorine atmintine (usb, cd, dvd, kortele ark t)?
   - 0-1
   - 2-5
   - 6 ir daugiau

2. Kur dažniausiai naudojatės išorine atmintine?
   - Namuose
   - Darbe
   - Kitur

3. Kokio tipo išorine duomenų laikmena naudojatės dažniausiai?
   - USB Flash
   - CD/DVD diskais
   - Išorinis kietasis diskas
   - Atminties kortelė
   - Kita

4. Kaip vertinate išorinės atmintinės svarbą šiomis dienomis?
   - Laikmenos labai reikalingos
   - Laikmenos nėra itin reikalingos
   - Laikmenos nėra reikalingos
5. Ar tuose kompiuteriuose, kuriuose naudojate išorines duomenų laikmenas yra interneto ryšys?
   - Yra visuose
   - Yra, bet ne visuose
   - Nėra

6. Ar žinote kas yra vadinama debesų technologija?

- Taip
- Ne

7. Ar kada bandėte savo duomenis saugoti kažkur internete?
   - Taip
   - Ne

8. Ar kada esate kaip nors praradę, pametę savo duomenų laikmeną?
   - Taip
   - Ne
9. Jeigu buvote pametę laikmeną ar praradote kiek nors svarbių duomenų?
   - Praradau
   - Nepraradau
   - Niekad nebuvau pametęs(usi)

10. Ar dažnai tenka į išorinę laikmeną rašyti konfidencialius duomenis?
    - Dažnai
    - Retai
    - Netenka

11. Jeigu tenka rašyti konfidencialius duomenis į išorinę laikmeną, ar manote, kad reikalinga kažkokia papildoma programinė įranga, kuri neleistu kitiems peržiūrėti Jūsų duomenų, pametus laikmeną?
    - Taip, reikalinga
    - Nemanau, kad tai reikalinga, užtenka saugoti pačią laikmeną
    - Nereikalinga

12. Kas Jūsų nuomone yra svarbiau, jeigu prarastumėte laikmeną?
    - Duomenų niekas neperskaitys, bet jie bus prarasti
    - Neprarasti duomenų
    - Abu faktoriai lygiaverčiai

13. Ar tam, kad pametus laikmeną, neprarastumėte duomenų, sutiktumėte juos laikyti nuotolinėje saugykloje, kuri pasiekiama internetu?
    - Taip
    - Ne

14. Kuris iš programinės įrangos vertinimo kriterijų Jums yra svarbesnis?
    - Lengvas vartotojo sąsajos suprantamumas
    - Funkcijų gausa

Apklausos rezultatai

Atlikus pateiktą aukščiau apklausą su 50 žmonių, buvo gauti rezultatai, kuriais vėliau buvo vadovaujamasi siūlant savo sprendimą magistriniame darbe. Apačioje pateikiami susisteminti apklausos rezultatai.

Kiek kartų per savaitę naudojatės išorine atmintine (usb, cd, dvd, kortele ar kt)?

Atsakymas	Kiek žmonių atsakė	Kiek žmonių atsake (proc.)
0-1	15	30
2-5	21	42
6 ir daugiau	14	28

Kur dažniausiai naudojatės išorine atmintine?

Atsakymas	Kiek žmonių atsakė	Kiek žmonių atsake (proc.)
Namuose	18	36
Darbe	31	62
Kitur	1	2

Kokio tipo išorine duomenų laikmena naudojatės dažniausiai?

Atsakymas	Kiek žmonių atsakė	Kiek žmonių atsake (proc.)
USB Flash	39	78
CD/DVD diskas	1	2

Išorinis kietasis diskas	10	20
Atminties kortelė	0	0
Kita	0	0

Kaip vertinate išorinės atmintinės svarbą šiomis dienomis?

Atsakymas	Kiek žmonių atsakė	Kiek žmonių atsake (proc.)
Laikmenos labai reikalingos	41	82
Laikmenos nėra itin reikalingos	7	14
Laikmenos nėra reikalingos	2	4

Ar tuose kompiuteriuose, kuriuose naudojate išorines duomenų laikmenas yra interneto ryšys?

Atsakymas	Kiek žmonių atsakė	Kiek žmonių atsake (proc.)
Yra visuose	48	96
Yra, bet ne visuose	2	4
Nėra	0	0

Ar žinote kas yra vadinama debesų technologija?

Atsakymas	Kiek žmonių atsakė	Kiek žmonių atsake (proc.)
Taip	32	64
Ne	18	36

Ar kada bandėte savo duomenis saugoti kažkur internete?

Atsakymas	Kiek žmonių atsakė	Kiek žmonių atsake (proc.)
Taip	7	14
Ne	43	86

Ar kada esate kaip nors praradę, pametę savo duomenų laikmeną?

Atsakymas	Kiek žmonių atsakė	Kiek žmonių atsake (proc.)
Taip	29	58
Ne	21	42

Jeigu buvote pametę laikmeną ar praradote kiek nors svarbių duomenų?

Atsakymas	Kiek žmonių atsakė	Kiek žmonių atsake (proc.)
Praradau	27	54
Nepraradau	2	4
Niekad nebuvau pametęs(usi)	21	42

Ar dažnai tenka į išorinę laikmeną rašyti konfidencialius duomenis?

Atsakymas	Kiek žmonių atsakė	Kiek žmonių atsake (proc.)
Dažnai	2	4
Retai	23	46
Netenka	25	50

Jeigu tenka rašyti konfidencialius duomenis į išorinę laikmeną, ar manote, kad reikalinga kažkokia papildoma programinė įranga, kuri neleistų kitiems peržiūrėti Jūsų duomenų, pametus laikmeną?

Atsakymas	Kiek žmonių atsakė	Kiek žmonių atsake (proc.)
Taip, reikalinga	16	32
Nemanau, kad tai reikalinga, užtenka saugoti pačią laikmeną	29	58
Nereikalinga	5	10

Kas Jūsų nuomone yra svarbiau, jeigu prarastumėte laikmeną?

Atsakymas	Kiek žmonių atsakė	Kiek žmonių atsake (proc.)
Duomenų niekas neperskaitys, bet jie bus prarasti	3	6
Neprarasti duomenų	5	10
Abu faktoriai lygiaverčiai	42	84

Ar tam, kad pametus laikmeną, neprarastumėte duomenų, sutiktumėte juos laikyti nuotolinėje saugykloje, kuri pasiekiama internetu?

Atsakymas	Kiek žmonių atsakė	Kiek žmonių atsake (proc.)
Taip	47	94
Ne	3	6

Kuris iš programinės įrangos vertinimo kriterijų Jums yra svarbesnis?

Atsakymas	Kiek žmonių atsakė	Kiek žmonių atsake (proc.)
Lengvas vartotojo sąsajos suprantamumas ir intuityvumas	38	76
Funkcijų gausa	12	24

Atsižvelgiant į apklausos rezultatus galima teigti, kad dauguma žmonių išorinę atmintinę naudoja bent vieną kartą per savaitę, dažniausiai ji jiems reikalinga darbe. Vis dar populiariausios išlieka USB Flash tipo atmintinės, kurios stipriai pirmauja prieš išorinius kietuosius diskus ir optines laikmenas. Apklausa rodo, kad daugiau nei 80 procentų apklaustųjų išorinės laikmenos yra labai svarbios. Maždaug pusė apklaustųjų bent vieną kartą gyvenime yra praradę savo laikmenas, iš kurių beveik visi, kartu prarado ir svarbius duomenis.

Maždaug pusei apklaustųjų bent jau kartais tenka rašyti konfidencialius duomenis į išorines laikmenas. Jeigu į išorines laikmenas konfidencialius duomenis rašytų visi apklaustieji, tik trečdalis naudotų papildomą programinę įrangą, kad duomenys nebūtų perskaityti trečiųjų šalių. Kiek daugiau nei pusė mano, jog svarbiausia yra saugoti pačią laikmeną ir konfidenciali informacija nebus prarasta. Paklausti, kas svarbiau praradus laikmeną, duomenų nepatekimas trečiajai šaliai, kad ir prarandant duomenis pačiam, ar jų nepraradimas, beveik visi atsakė, jog abu šie faktoriai yra lygiaverčiai. Kone visi paklausti ar laikytų duomenis internetinėje erdvėje, norėdami apsaugoti juos nuo praradimo, atsakė taip, nors kiek daugiau nei trečdalis jų nėra girdėją apie debesų technologiją, o keli net neturi interneto visuose kompiuteriuose, kuriuose naudoja išorines laikmenas.

Siekiant atlikti eksperimentinę tiriamojo darbo dalį kuo tiksliau, apklausoje buvo užduotas klausimas, kas yra svarbiau programinėje įrangoje ar funkcijų gausa ar vartotojo sąsajos paprastumas ir suprantamumas. 76 procentai apklaustųjų atsakė, kad jie labiau vertina vartotojo sąsajos paprastumą ir suprantamumą.

Apibendrinimas

Apklausa parodė, jog žmonėms svarbu ir neprarasti duomenis, kai pametama išorinė laikmena, ne tik kad duomenys būtų apsaugoti nuo nesankcionuoto vartojimo. **Apklausos rezultatai padėjo nuspręsti, koks tiriamojo darbo siūlomas sprendimas būtų geriausias ir į ką labiausiai atsižvelgia vartotojai rinkdamiesi programinę įrangą.**