KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS IR INFORMACINIŲ TECHNOLOGIJŲ SAUGOS STUDIJŲ PROGRAMA

TOMAS KREICKAMAS

PROGRAMŲ APSAUGOS, NAUDOJANT LUSTINES KORTELES, METODO SUDARYMAS IR TYRIMAS

Magistro baigiamasis darbas

Darbo vadovas
doc. dr. A. Venčkauskas

KAUNAS, 2013

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS IR INFORMACINIŲ TECHNOLOGIJŲ SAUGOS STUDIJŲ PROGRAMA

TOMAS KREICKAMAS

PROGRAMŲ APSAUGOS, NAUDOJANT LUSTINES KORTELES, METODO SUDARYMAS IR TYRIMAS

Magistro baigiamasis darbas

Recenzentas
doc. dr. T. Blažauskas
2013-05-

Darbo vadovas
doc. dr. A. Venčkauskas
2013-05-

Darbą atliko:
Tomas Kreickamas
2013-05-22

KAUNAS, 2013

AUTORIŲ GARANTINIS RAŠTAS
DĖL PATEIKIAMO KŪRINIO

2013 - 05 - 22 d.
Kaunas

Autoriai, **Tomas Kreickamas**
(vardas, pavardė)

_____,

patvirtina, kad Kauno technologijos universitetui pateiktas baigiamasis bakalauro (magistro) darbas (toliau vadinama – Kūrinys) „**Programų apsaugos, naudojant lustines korteles, metodo sudarymas ir tyrimas**“
(kūrinio pavadinimas)

pagal Lietuvos Respublikos autorių ir gretutinių teisių įstatymą yra originalus ir užtikrina, kad

1) jį sukūrė ir parašė Kūrinyje įvardyti autoriai;
2) Kūrinys nėra ir nebus įteiktas kitoms institucijoms (universitetams) (tiek lietuvių, tiek užsienio kalba);
3) Kūrinyje nėra teiginių, neatitinkančių tikrovės, ar medžiagos, kuri galėtų pažeisti kito fizinio ar juridinio asmens intelektinės nuosavybės teises, leidėjų bei finansuotojų reikalavimus ir sąlygas;
4) visi Kūrinyje naudojami šaltiniai yra cituojami (su nuoroda į pirminį šaltinį ir autorių);
5) neprieštarauja dėl Kūrinio platinimo visomis oficialiomis sklaidos priemonėmis.
6) atlygins Kauno technologijos universitetui ir tretiesiems asmenims žalą ir nuostolius, atsiradusius dėl pažeidimų, susijusių su aukščiau išvardintų Autorių garantijų nesilaikymu;
7) Autoriai už šiame rašte pateiktos informacijos teisingumą atsako Lietuvos Respublikos įstatymų nustatyta tvarka.

Autoriai

Tomas Kreickamas	
(vardas, pavardė)	(parašas)
(vardas, pavardė)	(parašas)
(vardas, pavardė)	(parašas)
(vardas, pavardė)	(parašas)

SANTRAUKA

Taikomųjų programų piratavimas – procesas, kurio metu nelegaliai atkuriama ir neturint tam teisės platinama taikomoji programa. Ši problema nėra nauja, tačiau efektyvių apsaugos priemonių nuo jos šiandien dar nesukurta. Dėl šios priežasties 2011 m. nelegalios programinės įrangos buvo parsisiųsta už daugiau nei 60 mlrd. JAV dolerių ir ši suma kasmet auga.

Atlikus taikomųjų programų grėsminių analizę išsiaiškinome, kad didžiausia problema – atvirkštinė inžinerija. Šią problemą padedančias išspręsti apsaugos priemones suskirstėme į programines ir aparatūrines. Atlikus programinių apsaugos priemonių analizę išsiaiškinome, kad geriausiai nuo atvirkštinės inžinerijos padeda apsisaugoti kodo šifravimas arba glaudinimas. Atlikus aparatūrinių apsaugos priemonių analizę išsiaiškinome, kad apsaugai nuo piratavimo dažniausiai naudojami apsaugos raktai.

Išanalizavus programinių ir aparatūrinių apsaugos priemonių privalumus ir trūkumus sukūrėme kompleksinį apsaugos metodą. Šis metodas remiasi kritinių (vertingiausių) programos modulių šifravimu ir vykdymu saugiame įrenginyje. Šiame darbe kaip saugų įrenginį naudojame lustines korteles. Šie įrenginiai buvo pasirinkti dėl jų nedidelės kainos ir teikiamo didelio saugumo lygio.

Atlikę sumodeliuoto metodo programinę realizaciją jį ištyrėme greitaveikos aspektu ir nustatėme, kad modulio užimančio 6KB iššifravimas lustinėje kortelėje trunka tik 2% viso programos vykdymo laiko, todėl didelės įtakos programos vykdymo laiko išaugimui neturi. Didžiausią įtaką apsaugotos programos vykdymo laiko padidėjimui turi modulio vykdymas (59,37% viso laiko) ir licencijos iššifravimas privačiuoju raktu (14,33% viso laiko) lustinėje kortelėje bei pranešimų siuntimas tarp programos ir lustinės kortelės (14,88% viso laiko). Siekiant sumažinti apsaugotos programos vykdymo laiką pirmiausia reikėtų optimizuoti šių trijų užduočių, kurios daugiausiai prisideda prie programos vykdymo laiko išaugimo, programinius kodus. Taip pat nustatėme, kad apsaugotos programos vykdymas nuo neapsaugos programos vykdymo skiriasi 4 kartus.

Apibendrinant galima teigti, kad mūsų pasiūlytas apsaugos metodas nuo piratavimo galėtų pasitarnauti mažinant piratavimo lygį, tačiau atliekant modulių vykdymą lustinėje kortelėje išauga taikomosios programos vykdymo laikas.

SUMMARY

Software piracy is copying and distributing of software illegally and without permission. This problem is not new but effective protective measures against it until today are not developed. Therefore, in 2011 illegal software has been downloaded for more than 60 billion USA dollars and that amount is growing every year.

After software threats' analysis we found out that the biggest problem is reverse engineering. Measures which can help to solve this problem we divided into software-based and hardware-based protection. After software-based protection analysis we found out that one of best measures against reverse engineering is code encryption or packaging and one of the best hardware-based protection tools is using of dongle keys.

After analysis of advantages and disadvantages of software-based and hardware-based protection we developed method against software piracy. This method relies on the encryption of critical (most valuable) program modules and its safe execution in a safe device. In this paper, as a safe device we will use smart cards. These devices were chosen for their low cost and high level of safety.

After implementation of simulated method we found out that decryption of module, which size is ~6KB, in smart card takes only 2% of the total program execution time, so this task does not have significant impact on program execution time. The biggest impact on increasing of protected program execution time have the module performance (59,37% of the total time), license decryption with private key (14.33% of the total time) and communication among protected program and smart card (14,88% of the total time). To reduce the execution time of the protected program we should first optimize these three program codes. Also we found out that the execution of protected program is slower 4 times than the execution of program without any protection.

In summary we could say, that our proposed method of protection against piracy could serve to reduce the level of piracy, but execution critical program modules in smart card increases execution time.

T**URINYS**

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

ĮVADAS

XXI a. – informacijos amžius. Čia pagrindinį vaidmenį vaidina informacija ir ja besinaudojanti informacinė visuomenė. Informacinė visuomenė, tai visuomenė, kuri yra išsilavinusi, nuolat besimokanti, savo veiklą ir priimamus sprendimus grindžianti žiniomis ir informacija bei plačiai besinaudojanti nuotolinio ryšio ir informacinių technologijų teikiamomis galimybėmis. Kompiuteris tapo neatsiejama šios visuomenės gyvenimo dalimi, kadangi būtent kompiuteris, ir jame įdiegtos specifinės taikomosios programos, leidžia išspręsti dauguma kasdieninių problemų bei užduočių. Taikomųjų programų kūrimas dažniausiai reikalauja daugybės pastangų ir materialių investicijų, todėl programos vertinamos kaip turinčios didelę intelektinę vertę.

Taikomųjų programų kūrėjams intelektinė nuosavybė yra pamatinis šio verslo produktas, iš kurio gaunama didžioji dalis pajamų, todėl intelektinės nuosavybės apsaugojimas šiame versle tampa ypač aktualus. Taikomųjų programų apsauga ypač turėtų būti suinteresuoti brangių taikomųjų programų paketų kūrėjai, kadangi kuo brangesnė taikomoji programa, tuo labiau ji domina programinės įrangos piratus.

Deja, tačiau dauguma taikomųjų programų neretai yra silpnai apsaugotos. Silpnais apsaugos mechanizmais dažniausiai pasinaudoja taikomųjų programų piratai, kurie apėję apsaugos priemones nukopijuoja programą ir taip taikomųjų programų kūrėjams pridaro materialių nuostolių. Dėl šios priežasties taikomųjų programų kūrėjai visą laiką ieško pačių efektyviausių būdų kaip apsaugoti šią intelektinę nuosavybę nuo neautorizuoto jos naudojimo.

Atsižvelgdami į prieš tai išsakytas mintis, šiame darbe išanalizuosime grėsmes programų saugai, išsiaiškinsime esamas saugumo priemones, sukursime metodą apsaugantį taikomąsias programas nuo piratavimo panaudojant lustines korteles bei atliksime pasiūlyto metodo kiekybinį tyrimą.

Lustinė kortelė – plastikinė kortelė, kurioje integruotas lustas. Šios kortelės yra plačiai paplitusios (atsiskaitymo kortelės, praėjimo kontrolės kortelės ir t.t.), todėl pasižymi dideliu saugumu. Šiame darbe siūlomo apsaugos metodo pagrindą sudaro lustinė kortelė. Užšifruoti kritiniai taikomosios programos kodo fragmentai siunčiami į lustinę kortelę kur yra iššifruojami ir įvykdomi, kai likęs neapsaugotas kodas yra vykdomas kompiuteryje.

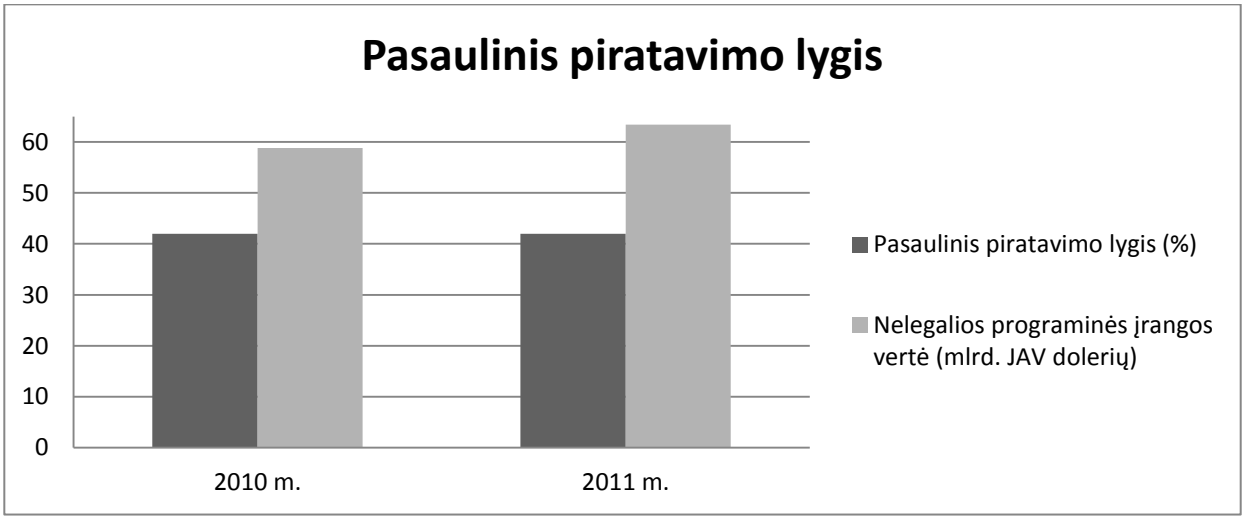
Šis darbas parašytas studijuojant informacijos ir informacinių technologijų saugą Kauno technologijos universitete.

Darbo problematika ir aktualumas

Taikomosios programos ir jas sudarantys algoritmai – intelektine nuosavybe, dėl kurios vyksta nuolatinė kova tarp taikomųjų programų kūrėjų ir ja nelegaliai bandančių pasinaudoti piratų. Kol

taikomųjų programų kūrėjai ieško efektyviausių būdų apsaugoti taikomąsias programas nuo nelegalaus jų naudojimo, piratai elgiasi priešingai – ieško trumpiausių būdų kaip apeiti taikomųjų programų kūrėjų sukurtus saugumo mechanizmus.

Deja, bet šiandieną taikomųjų programų piratai šią kovą iš dalies laimi. Kad ir koks apsaugos mechanizmas būtų sukurtas, jis nėra amžinas ir galiausiai yra apeinamas. Idealiausias šios problemos sprendimas – sukurti apsaugos mechanizmą, kuris būtų atsparus įvairioms atakoms ir nereikalautų didelių išlaidų jį įdiegiant. Tačiau šiai dienai tokio apsaugos mechanizmo nėra. Patikimo metodo, apsaugančio taikomąsias programas nuo piratavimo, nebuvimą įrodo ir *Business Software Alliance* (*BSA*) atliekami tyrimai [1] [2]. Tyrimo rezultatai pateikiami 1.1 pav.

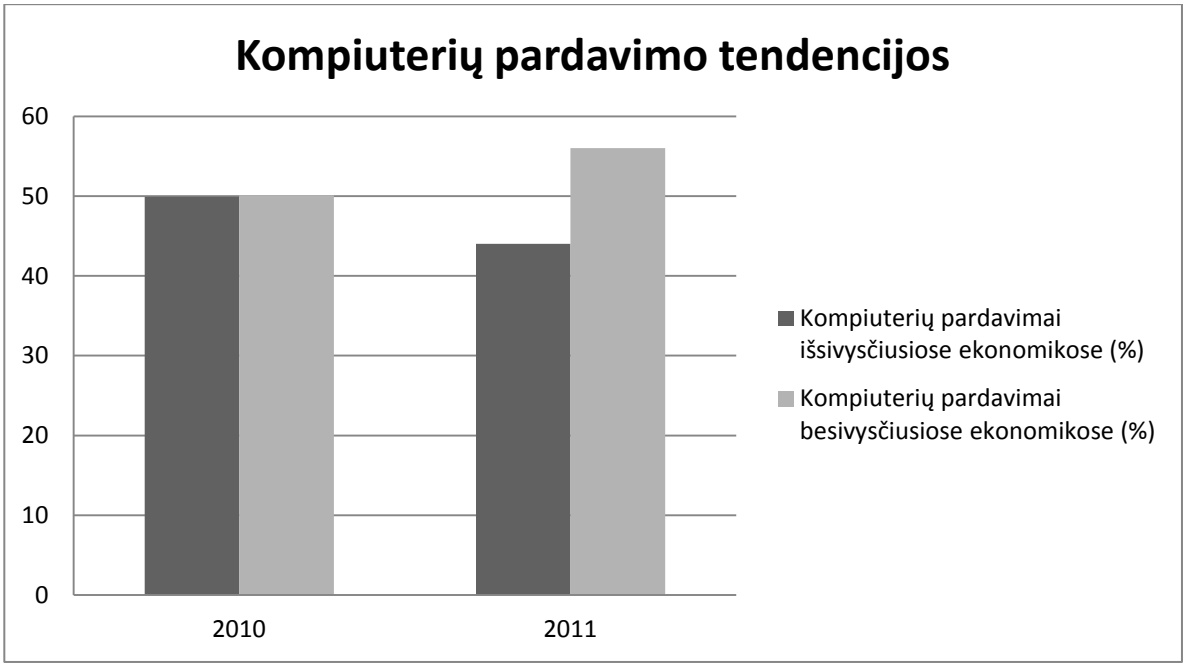


1.1 pav. Pasaulinis piratavimo lygis 2010-2011 metais

Nors piratavimo lygis 2010-2011 m. išliko toks pats (42%), tačiau piratinės įrangos vertė išaugo nuo 58,8 mlrd. JAV dolerių iki 63,4 mlrd. JAV dolerių. Lietuvoje piratavimo lygis pastaraisiais metais nekinta, tačiau išlieka labai aukštas (54%). Nelegalios programinės įrangos vertė Lietuvoje sudaro apie 38 mln. JAV dolerių.

Piratinės programinės įrangos vertė auga dėl sparčiai augančio kompiuterių pardavimo besivystančiose rinkose. Kompiuterių pardavimo tendencijos 2010-2011 metų laikotarpiu pateikiamos 1.2 pav. 2010 m. metus galime laikyti lūžio metais, kai kompiuterių pardavimas besivystančiose ekonomikose buvo didesnis (50,1%) nei išsivysčiusiose ekonomikose (49,9%). Šešėlinės programinės įrangos rinkos didėjimą lemia ir tai, kad pastaraisiais metais asmeninių kompiuterių kainos krinta, o programinės įrangos priešingai – auga.

Ekonomistai yra paskaičiavę [3], jog per ketverius metus sumažinus pasaulinį piratavimo lygį 10% (kasmet po 2,5%) būtų galima sukurti apie pusę milijono naujų darbo vietų.



1.2 pav. Kompiuterių pardavimai 2010-2011 metais

Darbo tikslas ir uždaviniai

Šio magistrinio darbo tikslas – **sukurti programų apsaugos nuo piratavimo metodą, panaudojant lustines korteles, ir jį ištirti.** Užsibrėžtam tikslui pasiekti turėsime įgyvendinti šias užduotis:

- atlikti taikomųjų programų grėsmių analizę ir išsiaiškinti metodus naudojamus taikomųjų programų apsaugų apėjimui;
- atlikti taikomųjų programų apsaugos metodų nuo piratavimo analizę ir juos palyginti;
- sukurti taikomųjų programų apsaugos nuo piratavimo metodą, panaudojant lustines korteles;
- ištirti sukurtą apsaugos metodą.

Darbo struktūra

Dokumentą sudaro 5 skyriai, kuriuose nagrinėjami darbo tematika aktualūs klausimai. Įvade supažindinama su darbo specifika, pateikiama darbo aktualumą pagrindžianti statistika, suformuluojamas darbo tikslas, kuris išskaidomas į uždavinius.

Analizės skyriuje analizuojama su darbo problematika susijusi kitų autorių informacija, atliekama aptiktų metodų lyginamoji analizė. Didelis dėmesys šiame skyriuje skiriamas lustinių kortelių analizei. Skyriaus pabaigoje pateikiamos analizės išvados.

Apsaugos metodo kūrimo skyriuje modeliuojamas apsaugos nuo piratavimo metodas panaudojant lustines korteles. Šiame skyriuje pateikiama apsaugos metodo koncepcija, aprašomi

taikomosios programos apsaugojimo, diegimo ir vykdymo veiksmai. Taip pat pateikiamas protokolas, kurio pagalba apsaugotas programos modulis gali būti vykdomas lustinėje kortelėje. Skyriaus pabaigoje pateikiamos metodo kūrimo išvados.

Sumodeliuoto metodo tyrimo skyriuje aprašoma šio metodo realizacija bei jo tyrimas. Analizuojama kokią įtaka apsaugotos programos vykdymo laikui turi skirtingų šifravimo algoritmų, raktų ilgių ir modulio užšifravimo procento pasirinkimas. Šiame skyriuje pateikiama informacija kiek kiekvienas papildomas apsaugos komponentas pailgino programos vykdymo laiką. Skyriaus pabaigoje pateikiamos išvados.

Išvadų skyriuje pateikiamas darbo metu gautų rezultatų apibendrinimas. Išvados formuojamos darbo uždavinių pagrindu.

2. TAIKOMŲJŲ PROGRAMŲ GRĖSMIŲ IR APSAUGOS METODŲ NUO PIRATAVIMO ANALIZĖ

Kadangi patikimo ir ilgalaikio apsaugos mechanizmo nuo piratavimo nėra, todėl apsaugos mechanizmai turi būti kuriami taip, kad apsaugos apėjimo kaina būtų didesnė nei pačios taikomosios programos kaina ir pareikalautų daugiau pastangų nei atkuriant taikomąją programą po apsaugos apėjimo. Šiame skyriuje analizuosime pažeidžiamumus, kurie leidžia nelegaliai naudotis taikomosiomis programomis, išsiaiškinsime taikomųjų programų apsaugos nuo esamų grėsmių metodus ir juos palyginsime.

2.1. Programinės įrangos licencijavimas

Egzistuoja įvairių programinės įrangos licencijų modelių – vienos platinamos nemokamai, už kitų naudojimąsi reikia susimokėti. Žemiau pateikiame apibendrintą licencijavimo modelių [4] [5] lentelę (2.1 lentelė).

2.1 lentelė. Licencijavimo modeliai

Pavadinimas	Aprašymas	Apsauga
Atviro kodo / Nemokama programinė įranga (angl. *Open Source / Free Software*)	Programinės įrangos licencijavimo modelis, kuris nedraudžia programinės įrangos kopijavimo. Vartotojas gali nemokamai parsisiųsti ir koreguoti taikomosios programos išeities kodą.	Apsaugos nuo piratavimo nenaudojamos.
Nemokama programinė įranga (angl. *Freeware*)	Labai panašus licencijavimo modelis į Atviro kodo / Nemokamos programinės įrangos, tačiau čia programos išeities kodas nėra atviras, todėl vartotojas negali jame daryti jokių pakeitimų.	Apsaugos nuo piratavimo nenaudojamos.
Viešai prieinama programinė įranga (angl. *Shareware*)	Šiai licencijavimo kategorijai priklauso programinė įranga, kurią jūs galite parsisiųsti ir išbandyti nemokamai, tačiau su tam tikrais apribojimais (apribotą laiko tarpą, su apribotu funkcionalumu ir t.t.). Norint pratęsti programos veikimo laiką arba gauti pilną programinės įrangos funkcionalumą reikia įsigyti pilną programinės įrangos licenciją.	Naudojamos apsaugos nuo piratavimo.
Vieno įrenginio licencija (angl. *Node Locked Licence*)	Programinė įranga įdiegiama tik viename įrenginyje. Įdiegus programinę įrangą kituose įrenginiuose už juos mokama papildomai. Trumpai tariant – 1 licencija, 1 kompiuteris.	Naudojamos apsaugos nuo piratavimo.
Plaukiojanti licencija (angl. *Floating Licence*)	Alternatyva vieno įrenginio licencijai. Šio licencijavimo modelio esmė – nusipirkus 1 licenciją, programinę įrangą galima diegti nustatytame skaičiuje kompiuterių ir ja naudotis tuo pačiu metu visuose kompiuteriuose.	Naudojamos apsaugos nuo piratavimo.

Kaip matome iš informacijos pateiktos 2.1 lentelėje, nemokamai programinei įrangai apsaugos nuo piratavimo metodai netaikomi. Visai kitoks vaizdas yra tuomet, kai už programinės įrangos naudojimą reikia susimokėti. Čia taikomi įvairiausi apsaugos nuo piratavimo metodai, kuriuos analizuosime tolimesniuose skyreliuose.

2.2. Piratavimo būdai ir apsaugos mechanizmų apėjimo priemonės

Išsiaiškinome naudojamus licencijavimo modelius ir kada naudojami apsaugos nuo pirato metodai. Toliau panagrinėkime piratavimo formas [6] [7], kurios pridaro daugiausiai žalos taikomųjų programų kūrėjams. Apibendrinta piratavimo formų klasifikacija pateikiama 2.2 lentelėje.

2.2 lentelė. Piratavimo formos

Pavadinimas	Paaiškinimas
Vartotojo piratavimas (angl. *User piracy*)	Piratavimo būdas, kai asmuo neteisėtai padaro legalios programinės įrangos kopiją ir ją naudoja asmeniniams tikslams.
Internetinis piratavimas (angl. *Internet piracy*)	Piratavimo būdas, kai į kompiuterį įdiegiama ar išorinėse laikmenose išsaugoma nelegaliai parsisiųsta programinės įrangos kopija.
Kietojo disko užpildymas (angl. *Hard disk loading*)	Piratavimo būdas, kai nelegaliai parsisiųsta programinė įranga arba jos kopija, pažeidžianti licencijavimo sąlygas, įrašoma į parduodamus kompiuterius. Šis piratavimo būdas plačiai paplitę Lietuvoje!
Programinės įrangos klastotės (angl. *Software counterfeiting*)	Piratavimo būdas, kai neteisėtai daromos programinės įrangos kopijos, įskaitant naudojimosi instrukcijas ir t.t.
Serverio perkrovimas (angl. *Server overloeading*)	Piratavimo būdas, kuomet serveryje įdiegta programine įranga bando pasinaudoti daugiau vartotojų, nei leidžia licencija.
Vertingų algoritmų vagystė	Algoritmų vagystės metu panaudojant rankines ar automatines atvirkštinės inžinerijos (angl. *Reverse engineering*) priemones piktavaliai siekia išgauti vertingus algoritmus, kurie toliau gali būti panaudojami konkurentų kuriamose taikomosiose programose.

Prieš pradedant nagrinėti apsaugos priemones nuo piratavimo panagrinėsime, kokios priemonės dažniausiai naudojamos apsaugos mechanizmų apėjimui.

Programinės įrangos piratai norėdami apeiti apsaugos mechanizmus dažniausiai naudojasi atvirkštine inžinerija. Atvirkštinė inžinerija – procesas, kurio metu mašininis kodas verčiamas į asemblerio kalbos kodą ir atliekama jo analizė [8] [9] [10]. Išskiriama statinė ir dinaminė atvirkštinė inžinerija.

Statinė atvirkštinė inžinerija – programinės įrangos kodo analizavimo metu kodas nevykdomas. Šio proceso metu analizuojamas dvejetainis failas. Dinaminė atvirkštinė inžinerija – kodo analizavimo metu kodas yra vykdomas. Šio proceso metu sekamas instrukcijų vykdymas. Priemonės, kurios dažniausiai naudojamos apsaugos mechanizmų apėjimui [11] – derintuvai (angl. *Debuggers*) ir ardymo priemonės (angl. *Disassemblers*).

Derintuvai – programos, leidžiančios piktavaliui sekti programinės įrangos veikimą instrukcija po instrukcijos. Programos veikimas gali būti sustabdomas bet kurioje jos vykdymo vietoje. Literatūroje dažniausiai minimi derintuvai: *OllyDbg, SoftIce.*

Ardymo priemonės – programos, leidžiančios programos kodą transformuoti į asemblerio kalbą. Literatūroje dažniausiai minima ardymo priemonė – *IDA Pro.*

2.3. Apsaugos nuo piratavimo priemonės

Išsiaiškinome, jog apsaugos mechanizmai nuo piratavimo taikomi programinei įrangai, už kurią reikia susimokėti. Nustatėme kokie piratavimo būdai padaro daugiausiai žalos programinės įrangos kūrėjams ir kad dažniausiai apsaugų nuo piratavimo apėjimui naudojama atvirkštinė inžinerija. Šiame skyrelyje analizuosime apsaugos priemones, kurios naudojamos apsaugoti programinę įrangą nuo neautorizuoto jos naudojimo.

Nors nelegalios programinės įrangos naudojimą draudžia įstatymai, tačiau jų nepakanka pažaboti augančiam piratavimo lygiui visame pasaulyje. Dėl šios priežasties programinės įrangos kūrėjai pradėjo taikyti papildomas apsaugos priemones, kurios gali būti suskirstytos į dvi pagrindines kategorijas [12] [13] [14]:

- programines apsaugos priemones (angl. *Software-based protection*) ir
- aparatūrines apsaugos (angl. *Hardware-based protection*)priemones.

Reiktų paminėti ir organizacines priemones [15], kurių pagalba siekiama sumažinti pasaulinį piratavimo lygį. Iš tokių priemonių paminėtinos:

- programinės įrangos įsigijimo supaprastinimas, dėl ko legalios programinės įrangos įsigijimas taptų lengvesnis;
- efektyvesnio legalios programinės įrangos klientų aptarnavimo įdiegimas;
- legalios programinės įrangos kainų sumažinimas;
- nuolatinių patikrinimų įmonėse ir organizacijose, kurios naudoja nelegalią programinę įrangą, skatinimas;
- skatinimas legalizuoti turimą nelegalią programinę įrangą (nemokamai, taikant nuolaidas ir t.t.) ir kt.

Taip pat reikėtų paminėti neseniai Jungtinėse Amerikos valstijose (JAV) pradėtą diegti „Šešių kirčių“ kovos su piratavimu programą [16]. Ši programa, tai įspėjimo apie autorių teisių pažeidimą sistema (angl. *Copyright Alert System*). Naudodamiesi šia sistema interneto tiekėjai galės sekti kiek ir iš kur žmogus siunčiasi duomenų „*peer-to-peer*“ (*P2P*) tinkluose. Pirmą kartą aptikus, kad nelegaliai siunčiamasi intelektinė nuosavybė, asmuo bus įspėjamas. Jei įspėjimo bus nepaisoma ir toliau nelegaliai siunčiamasi intelektinė nuosavybė, asmuo bus įspėjamas dar pora kartų. Jei ir tai nepadės, tuomet bus imtasi griežtesnių priemonių (pvz. lėtins teikiamos jungties greitį, blokuos prisijungimą prie populiariausių svetainių, nukreips į specialias svetaines ir kt.). JAV didieji interneto paslaugų tiekėjai tvirtina, kad tai ne baudžiamoji, o auklėjamoji priemonė ir drastiškesnių veiksmų (pvz. visiškas interneto atjungimas) bent šiuo metu imtis neketina. Be to, jei būtų pradėta teisminiai

procesai dėl nelegalaus turinio siuntimosi, klientui išsiųsti pranešimai taptų rimtu įrodymu tokiose bylose.

2.3.1. Programinės apsaugos priemonės

Programinės įrangos apsaugai nuo piratavimo (įskaitant ir nuo vertingų algoritmų vagystės) taikomi šie programiniai apsaugos metodai [12] [17] [18]:

- programinio kodo supainiojimas (angl. *Code obfuscation*);
  - schematinis supainiojimas (angl. *Layout obfuscation*);
  - duomenų supainiojimas (angl. *Data obfuscation*);
  - valdymo srauto supainiojimas (angl. *Control flow obfuscation*);
- apsauga nuo programų derinimo (angl. *Antidebugging*);
- apsauga nuo programinio kodo išskaidymo (angl. *Antidesassembly*);
  - bevertis kodas (angl. *Junk code*);
  - valdymo srauto slėpimas (angl. *Control flow hidding*);
  - kodo šifravimo technologija (angl. *Code encryptiom technology*);
- programų vandens ženklinimas (angl. *Watermarking*);
- programos kodo glaudinimas arba šifravimas (angl. *Code extracting / encryption*);
- sisteminiai parašai (angl. *System signatures*).

Programinio kodo supainiojimas

Šios programinės apsaugos priemonės tikslas – programos valdymo srautą ir/arba duomenų struktūras padaryti kuo sunkiau suprantamas piktavaliams. Originalios ir supainiotos programos funkcionalumas turi išlikti toks pats. Kodo supainiojimo metodo saugumas stipriai priklauso nuo pasirinktos transformacijos *K*. Parametras nusakantis supainiojimo patikimumą vadinamas stiprumu (angl. *Potency*) ir nusako kaip sunku yra suprasti ir analizuoti supainiotą kodą. Supainiojimo transformacijas galima suklasifikuoti į šias grupes:

- **Schematinis supainiojimas.** Šios programinės apsaugos priemonės tikslas – iš programos pašalinti visą nereikalingą informaciją (komentarus, nenaudojamas klases ir metodus ir kt.) ir apjungti esamą kodą tarpusavyje. Tokio supainiojimo pavyzdžiu galėtų būti dviejų klasių apjungimas į vieną.
- **Duomenų supainiojimas.** Šios programinės apsaugos priemonės tikslas – pakeisti programos duomenis ir jų struktūrą. Pakeitimai atliekami panaudojus įvairius suskaldymo metodus, pvz.: paveldėjimo ryšių pakeitimą, masyvų pertvarkymą, eilučių kodavimą, kintamųjų suskaldymą ir kt. Suskaldymo metodai klasifikuojami į:
  - **išskaidyto medžio** – panaudojant išskaidymo medžius statiniai kintamieji pakeičiami dinaminiais;

- **dalybos su liekana naudojimą** – naudojantis dalyba su liekana sudėtiniai skaičiai išskaidomi į mažesnius skaičius;
- **eilės tvarkos sukeitimo sąrašų naudojimą** – naudojant eilės tvarkos sukeitimo sąrašus supainiojami sveikieji skaičiai;
- **loginių operacijų naudojimą** – loginių operacijų (OR, AND, XOR ir t.t.) panaudojimas kintamųjų išskaidymui;
- **masyvų pertvarkymą** – masyvų pertvarkymas atliekamas klastojant masyvus (pakeičiant tam tikrus jų parametrus), atliekant kelių masyvų tarpusavio apjungimą arba vieno masyvo išskaidymą į kelis mažesnius masyvus.

- **Valdymo srauto supainiojimas.** Šios programinės apsaugos priemonės tikslas – į programą įterpti srauto supainiojimo šakas siekiant valdymo srautą padaryti kaip įmanoma sunkiau suprantamą. Taip pat galimas valdymo srauto pertvarkymas. Valdymo srauto supainiojimas klasifikuojamas į:
  - **valdymo srauto sukaupimą** (angl. *Control aggregation obfuscation*) – naudojantis įterpimo ir matmenų keitimo būdais pakeičiami taikomosios programos metodai ir jų kvietimo tvarka;
  - **valdymo srauto tvarkos supainiojimą** (angl. *Control ordering obfuscation*) – keičiama instrukcijų iškvietimo tvarka;
  - **valdymo srauto skaičiavimų supainiojimą** (angl. *Control computation obfuscation*) – įterpiant fiktyvias instrukcijas bandoma paslėpti tikrąjį kontrolės srautą.

Apsauga nuo programų derinimo

Šios programinės apsaugos priemonės tikslas – apsaugoti taikomąsias programas nuo derintuvų naudojimo taip apsunkinant atvirkštinės inžinerijos procesą. Išskiriami penki metodai, naudojami derintuvų aptikimui:

- **laiko stebėjimo** – vykdant programą yra lyginami du laikai tarp instrukcijų vykdymo. Jei laikai skiriasi, tai yra signalas programai, kad naudojamas derintuvas;
- **išimtinių situacijų stebėjimo** – stebint operacinės sistemos procesus tikrinama ar procesas derinamas. Jei aptinkama derinimo požymių, programa į tai reaguoja ir paleidžiamas atitinkamas apsaugos mechanizmas;
- **žinomų derinimo priemonių, programinių langų pavadinimų aptikimo** – tikrinami operacinės sistemos procesai ir ieškoma paleistų derinimo priemonių (pvz. *OLLYDBG.EXE, windbg.EXE* ir kt.);

- **vėliavėlių tikrinimo** – apsaugota programa jos vykdymo metu tikrina *BeingDebugged* ir/arba *NtGlobalFlag* vėliavėles ir taip nustato ar yra naudojami vartotojo lygio derintuvai;
- **atskaitos taškų aptikimo** – apsaugota programa jos vykdymo metu tikrina ar instrukcijos *INT1* ir *INT3* nebuvo pertrauktos. Jei šios instrukcijos buvo pertrauktos, tai signalas kad yra naudojamas derintuvas.

Apsauga nuo programinio kodo išskaidymo

Šios programinės apsaugos priemonės tikslas – apsaugoti taikomąsias programas nuo kodo išskaidymo linijinio nuskaitymo (angl. *Liner-scanning*) ir rekursijos žingsnių (angl. *Recursion-marching*) technologijų. Išskiriami šie apsaugos būdai nuo kodo išskaidymo:

- **Bevertis kodas** (angl. *Junk code*). Bevertis kodas skirstomas į bevertes instrukcijas ir beverčius duomenis. Bevertės instrukcijos ir duomenys turi būti įterpiami taip, jog atrodytų lyg būtų programos dalimi ir išsaugotų programos vientisumą.
- **Valdymo srauto slėpimas** (angl. *Control flow hidding*). Dažniausiai naudojama šie valdymo srauto slėpimo būdai:
  - **šuolių lentelės apgaulės** (angl. *Jump table cheat*). Šia technika siekiama kiek galima labiau apsunkinti supratimą apie tai, kaip veikia programa. Tai įgyvendinama panaudojant netikrus programos šuolius tarp jos dalių;
  - **nukreipimo funkcijos** (angl. *Branch function*). Šia technika siekiama pakeisti procedūrų kvietimo tvarką. Tai įgyvendinama originalios paprogramės vykdymą pakeičiant nukreipimo funkcija, kuri paprogramės funkcijas vykdo netiesiogiai. Papildomai pridedama grįžimo adresų.
- **Kodo šifravimo technologija** (angl. *Code encryption technology*). Šios programinės apsaugos priemonės tikslas – užšifruoti programos kodą panaudojant duomenų šifravimo technologijas. Ši apsaugos priemonė gerai apsaugo nuo programos išskaidymo į dalis. Vienas iš kodo šifravimo pavyzdžių – savarankiškai besikeičiantis kodas (angl. *Self-modyfying code*). Šio metodo pagalba paslepiamas tikrasis programos kodas, naudojant perteklines instrukcijas (angl. *Dummy instructions*). Kai tik programa paleidžiama, slepiami programos kodai keičiasi, todėl vykdant šifruotas instrukcijas bus gaunami klaidų pranešimai ir programos kodo skaidymo procesas nepavyks.

Programų vandens ženklinimas

Šios programinės apsaugos tikslas – į programinę įrangą įterpti slaptą informaciją, kuri nusako autorystės teises. Tokia apsauga naudojama nelegalios programinės įrangos naudojimo sekimui ar

programinio kodo panaudojimo kituose produktuose atsekimui. Vandens ženklai klasifikuojami į dvi grupes: statiniai ir dinaminiai (Velykų kiaušinis, duomenų struktūrų, vykdymo eigos). Statinius vandens ženklus lengviau aptikti nei dinaminius.

Programos kodo glaudinimas arba šifravimas

Šios programinės apsaugos tikslas – suglaudinti arba užšifruoti programos kodą (ar jo dalį). Taip pakeistas kodas (ar jo dalis) gali būti įvykdytas tik atlikus atstatomuosius veiksmus. Originali glaudinimo paskirtis – sumažinti vykdomojo failo dydį, tačiau šiandieną šis metodas naudojamas apsaugant programinę įrangą nuo atvirkštinės inžinerijos. Suglaudintas failas gali būti papildomai apsaugotas slaptu raktu.

Panašiu principu veikia ir kodo šifravimas. Kodas užšifruojamas slaptu raktu ir saugomas šifruotoje formoje. Vykdant programą kodas (ar jo dalis) iššifruojamas ir įvykdomas. Po vykdymo iššifruotas kodas dažniausiai pašalinamas iš atminties. Skirtingai nei kodo glaudinime, kodo šifravime būtinas slaptas raktas. Čia iškyla problema – kur saugoti slaptą raktą, kad jis nepakliūtų piktavaliams.

Sisteminiai parašai

Programų apsaugai nuo neteisėto [19] naudojimosi ir platinimo naudojamos funkcijos, susiejančios programos kodą vykdantį procesą su kompiuteriu, kuriame programinė įranga buvo įdiegta. Kiekvieną kartą prieš vykdant programą reikia atlikti tokius veiksmus:

- atlikti kompiuterio, kuriame ji paleista, aparatūrinės ir programinės įrangos analizę ir jos pagrindu suformuoti vykdymo aplinkos dabartinius parametrus.
- patikrinti vykdymo aplinkos parametrų autentiškumą – palyginti esamus parametrus esu etaloniniais (etaloniniai parametrai sudaromi įdiegiant programinę įrangą), kurie saugomi standžiajame diske ar kitoje išorinėje laikmenoje;
- blokuoti programos veikimą, jei parametrai nesutampa.

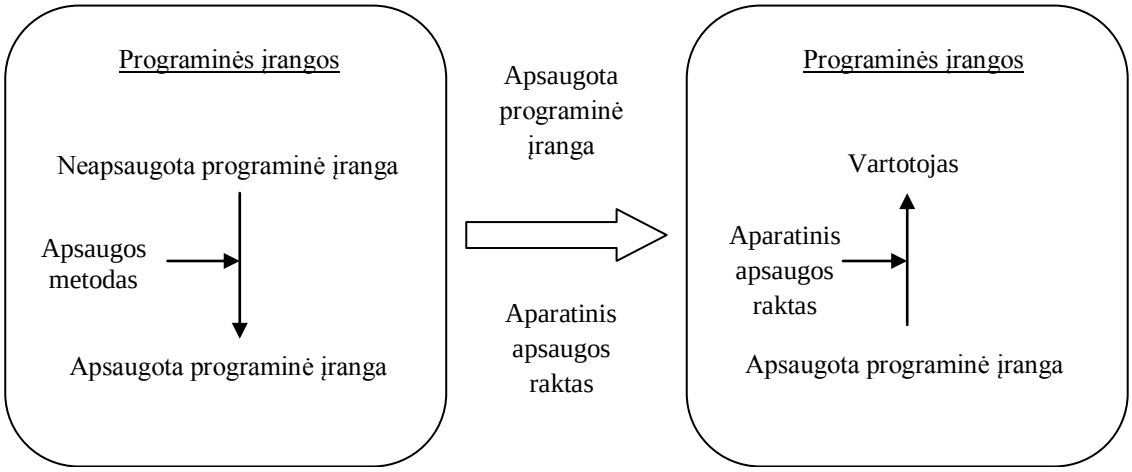
2.3.2. Aparatūrinės apsaugos priemonės

Naudojant programines apsaugos priemones kompiuteris laikomas saugia aplinka (angl. *Trusted enviroment*) programos vykdymui. Visai kitoks vaizdas yra tuomet, kai naudojamos aparatūrinės apsaugos priemonės. Čia kompiuteris jau laikomas nebesaugia aplinka programų vykdymui (angl. *Untrusted enviroment*). Tokiu atveju prie kompiuterio jungiamas saugus, sunkiai suklastojamas įrenginys, kuris laikomas saugia taikomosios programos vykdymo aplinka. Vartotojas norėdamas pasinaudoti taikomąja programa privalo turėti apsaugos schemoje numatytą aparatūrinį įrenginį, kitaip jis negalės visiškai naudotis taikomąja programa arba pasieks tik dalinį jos funkcionalumą. Įvairiuose literatūros šaltiniuose [12] [13] [15] išskiriamos šios aparatūrinės apsaugos priemonės:

- apsaugos raktai (angl. *Dongles*);
- kriptografiniai lustai (angl. *Cryptographic chipset*);
- saugūs procesoriai (angl. *Secure processors*);

Apsaugos raktai

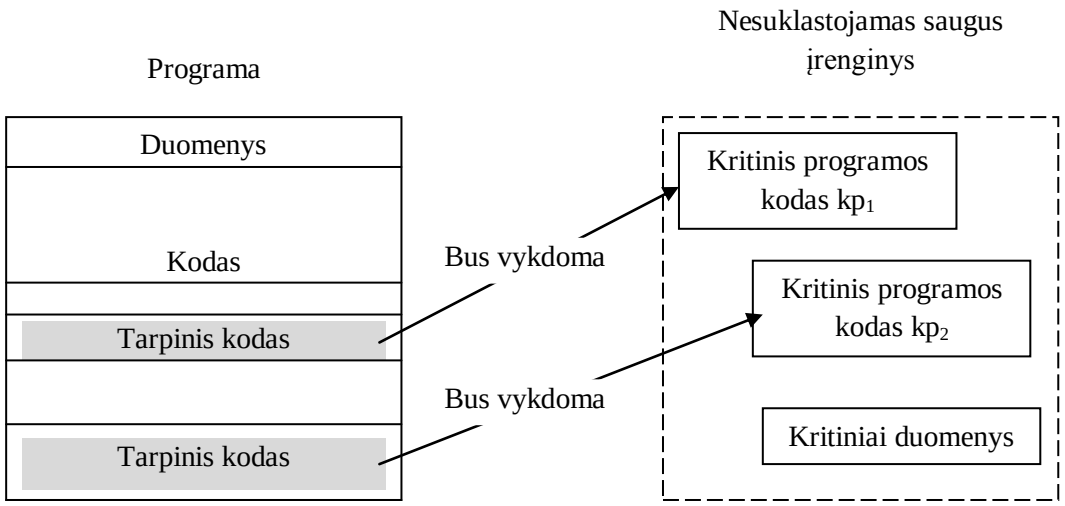
Apsaugos raktas – nedidelis aparatūrinis įrenginys (panašus į *USB* raktą), kuris išoriškai prijungiamas prie kompiuterio tam, kad būtų galima pasinaudoti apsaugotos programos funkcionalumu. Toks įrenginys dažniausiai jungiamas per USB prievadą. Apsaugos raktais pagrįsta apsaugos metodą galima pavaizduoti apibendrinta schema, kuri pateikiama 2.1 pav.



2.1 pav. Supaprastinta apsaugos rakto panaudojimo schema

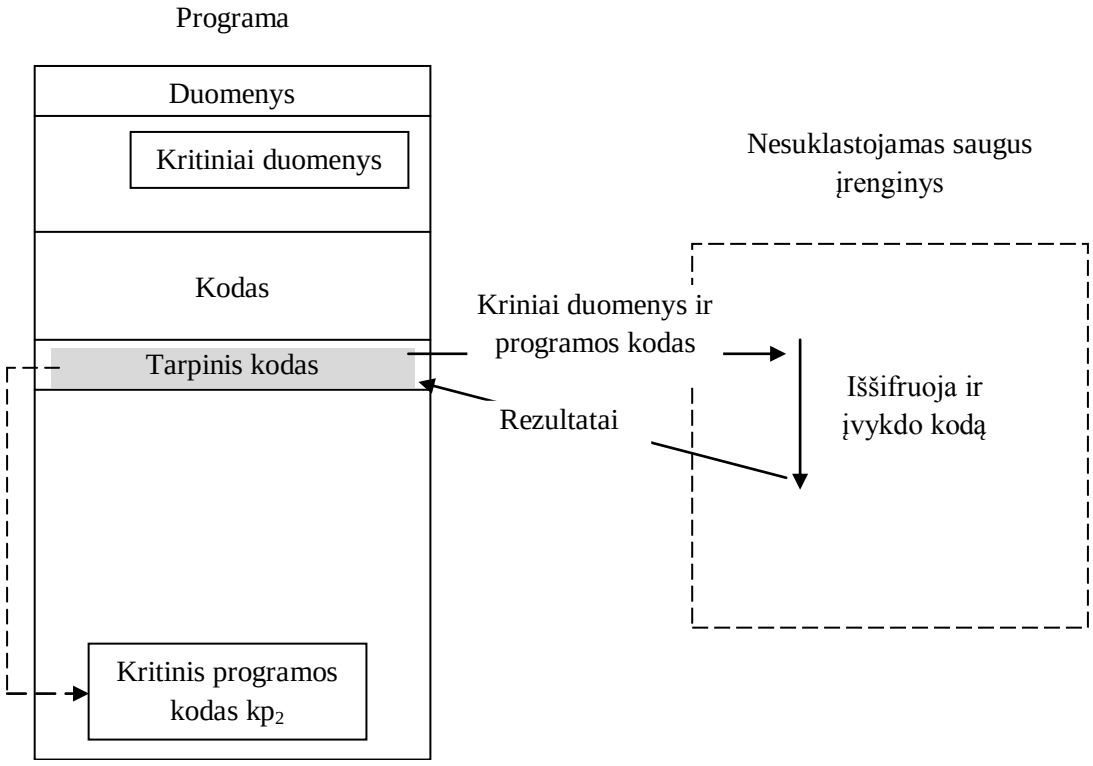
Programinės įrangos apsauga naudojant tokį apsaugos raktą klasifikuoja į šias grupes [13]:

- apsaugos rakte saugoma taikomosios programos licencija. Kiekvieną kartą paleidus programą tikrinama ar licencijoje esantis serijinis numeris bei kita informacija yra teisinga.
- apsaugos rakte saugomas slaptas raktas. Kompiuteryje saugoma programa yra iš dalies arba visiškai užšifruota. Kiekvieną kartą paleidus programą ji yra iššifruojama raktu, kuris saugomas apsaugos rakte.
- Vertingiausi taikomosios programos kodo fragmentai yra saugomi apsaugos rakte. Paleidus programą šie kodo fragmentai vykdomi ne kompiuteryje, o apsaugos rakte. Rezultatai iš apsaugos rakto gražinami kompiuteryje veikiančiai programai. Tokios apsaugos principinė schema pateikiama 2.2 pav.



2.2 pav. Kritiniai kodo fragmentai saugomi ir vykdomi apsaugos rakte

- apsaugos raktas naudojamas kaip saugus papildomas procesorius taikomosios programos kritinių kodo fragmentų vykdymui. Šiuo atveju programos kodas padalinamas į privatų kodą, kuris yra užšifruojamas, ir viešąjį kodą. Tiek viešas, tiek privatus kodas yra saugomas kompiuteryje, t.y. nesaugioje aplinkoje. Programos vykdymo metu viešasis kodas vykdomas kompiuteryje, o privatus kodas siunčiamas į apsaugos raktą ir ten pagalbinio procesoriaus pagalba iššifruojamas bei įvykdomas. Rezultatai gražinami atgal į nesaugią aplinką. Tokios apsaugos principinė schema pateikiama 2.3 pav.



2.3 pav. Kritinio kodo vykdymas saugiame įrenginyje

Kriptografiniai lustai

Kriptografiniai lustai siūlo kriptografines paslaugas, kurios leidžia patikrinti ir patvirtinti kompiuteryje naudojamų taikomųjų programų legalumą. Programinė įranga tikrinama šia tvarka: 1) *BIOS*; 2) *OS*; 3) Trečiųjų šalių programinė įranga. Naudojantis šia apsaugos schema tik patikima trečiųjų šalių programinė įranga, kuri buvo patvirtinta sertifikavimo institucijos, gali būti paleidžiama tokiame kompiuteryje. Nors tokia apsaugos priemonė gali ženkliai sumažinti piratinės programinės įrangos naudojimą, tačiau ji yra brangi.

Saugūs procesoriai

Naudojant saugius procesorius taikomosios programos yra vykdomos saugioje aparatūrinėje aplinkoje, kur piratinės programos gali neveikti. Tokiuose procesoriuose naudojama XOM (*eXecute Only Memory*) architektūra. Šios architektūros procesoriuose programa vykdoma apsaugotoje atmintyje. Vykdomos programos duomenys yra apsaugomi taip, kad tik atitinkamas procesas juos pasiekti. Jei duomenys turi būti perkelti į nesaugią aplinką, jie visada šifruojami.

2.4. Programinių ir aparatūrinių apsaugos priemonių palyginimas

Atlikus esamų apsaugos priemonių analizę, galime palyginti programines ir aparatūrines apsaugos priemones.

Programinių apsaugos priemonių gausa suteikia didelę pasirinkimo laisvę renkantis norimus apsaugos metodus. Vienas pagrindinių šių priemonių privalumų – tokios apsaugos įdiegimas yra sąlyginai nebrangus. Be to, tai lengviau įdiegiama ir lankstesnė apsauga nei aparatūriniai apsaugos sprendimai. Dėl šių priežasčių ir programinės įrangos pardavimo kaina yra mažesnė, nei naudojant aparatūrines apsaugos priemones. Tačiau nepaisant paprastumo ir sąlyginai nedidelių sąnaudų diegiant tokią apsaugą, programinės apsaugos priemonės turi ir trūkumu. Literatūros šaltiniuose [13] [18] dažnai minima, kad programinės apsaugos priemonės yra trumpalaikės ir neretai greitai apeinamos. Taip yra todėl, kad nuolat auga skaičiuojamoji kompiuterių galia. Be to, apsaugota programa yra saugi tik tol, kol ji nevykdoma. Pradėjus vykdyti programą ji siunčiama į atmintį ar procesorių, kur piktavaliai gali ją analizuoti.

Aparatūrinės apsaugos priemonės, priešingai nei programinės apsaugos priemonės, yra brangios, o tai reiškia, kad programinės įrangos kaina galutiniam vartotojui išauga. Kainos augimas susijęs su papildomus aparatūrinės įrangos įsigijimu. Be to, tokios apsaugos įdiegimas programinėje įrangoje reikalauja daugiau pastangų nei programinių priemonių įdiegime. Nors aparatūrinių apsaugos nuo piratavimo metodų įdiegimas programinėje įrangoje yra brangus, tačiau jie geriau apsaugo nuo neautorizuoto programinės įrangos naudojimo. Kitas aparatūrinių apsaugos priemonių trūkumas kai kalbame apie apsaugos raktus – tai kad jie gali pasimesti. Be to, atsiradus saugumo

spragų taikomosiose programose jų ištaisymas ir programinės įrangos atnaujinimas pareikalautų daugybės pastangų ir išlaidų.

Tiek programinės, tiek aparatūrinės apsaugos nuo piratavimo priemonės turi savų privalumų ir trukumų. Kaip jau ankščiau minėjome, universalios apsaugos nuo piratavimo nėra, todėl reikia taikyti kompleksines apsaugos priemones. Šias priemones sudaro programinės ir aparatūrinės apsaugos priemonės apjungtos į vieną apsaugos schemą.

Tačiau vien tik atsitiktinių apsaugos priemonių sudėjimas į vieną metodą reikiamos apsaugos neduos. Šį faktą patvirtina atlikti tyrimai [20]. Remiantis šiuo tyrimu, net ir apjungus aparatūrinius ir programinius apsaugos metodus apsauga gali būti greitai apeita. Tyrimo rezultatai pateikiami 2.3 lentelėje.

Atliekant šį tyrimą programa naudodamasi apsaugos rakto *API* (*Application programming Interface*) patikrindavo rakto *ID* (identifikatorius). Visas kodas buvo apsaugomas papildomomis apsaugos priemonėmis. Iš aštuonių apsaugos variantų tik vieno nepavyko apeiti. Šiame apsaugos variante buvo naudojamas apsaugos raktas, kodo suglaudinimas ir papildoma apsauga – *Crypto Obfuscator*. Šio tyrimo autoriai pabrėžia, kad dažniausiai kiekvienas glaudinimo programinis paketas turi atvirkštinį veiksmą atliekantį paketą, todėl tokia apsauga nėra visiškai patikima.

2.3 lentelė. Kompleksinės apsaugos apėjimo laikas **[20]**

	Naudojama tik apsaugos rakto apsauga			Naudojama apsaugos rakto ir kodo sumaišymo apsauga			Naudojama apsaugos rakto ir suglaudinimo apsauga	
Programavimo kalba	C++	.NET	Java	.NET		Java	C++	.NET
Papildoma apsauga	Nėra	Nėra	Nėra	Dotfuscator	Crypto Obfuscator	Pro Guard	SoftwarePassport (Armadillo)	Crypto Obfuscator
Laikas (minutėmis)	~10	~5	~5	~3	~8	~7	~15	Nepavyko apeiti

2.5. Simetrinės ir asimetrinės šifravimo sistemos

Norint apsaugoti svarbius duomenis (mūsų atveju ir programos modulius) naudojami kriptografiniai algoritmai, kurių dėka duomenys ir kodas yra šifruojami. Pagal naudojamų raktų skaičių kriptografiniai šifravimo algoritmai skirstomi į:

- **Simetrinius**. Duomenų užšifravimui ir iššifravimui naudojamas vienas slaptas raktas. Šiam tipui priskiriami blokiniai ir srautiniai šifrai. Blokiniais šifrais užšifruojami dideli fiksuoto ilgio duomenų blokai, o srautiniais – dažniausiai vieno bito ar baito dydžio blokai.
- **Asimetrinius** (viešojo rakto). Šio tipo šifrai duomenų užšifravimui naudoja viešąjį raktą, o iššifravimui – slaptą privatųjį raktą.

Simetrinių ir asimetrinių šifravimo sistemų lyginamoji [21] analizė pateikiama 2.4 lentelėje.

2.4 lentelė. Simetrinių ir asimetrinių šifravimo sistemų palyginimas

	Privalumai	Trūkumai
Simetrinės šifravimo sistemos	Simetrinio šifravimo algoritmai pasižymi itin sparčiu veikimu. Simetrinėse šifravimo sistemose naudojami raktai yra 4–5 kartus trumpesni nei asimetrinėse šifravimo sistemose. Simetrinio šifravimo algoritmo pagrindu gali būti sudaromi kiti kriptografiniai algoritmai.	Simetrinis šifravimo raktas turi likti slaptas (jį gali žinoti tik bendraujančios šalys). Egzistuoja raktų apsikeitimo ir saugojimo problema. Dideliame tinkle reikia saugoti daug raktų porų (raktų skaičius priklauso nuo šalių skaičiaus kvadrato). Raktas tarp dviejų šalių turi būti dažnai keičiamas (rekomenduojama kiekvieno ryšio seanso metu).
Asimetrinės šifravimo sistemos	Privatusis raktas turi būti slaptas, o viešasis raktas autentiškas. Raktų administravime turi dalyvauti patikima trečioji šalis *. Ji turi būti sąžininga visų vartotojų atžvilgiu, negali turėti prieigos vartotojų slaptos informacijos. Atsižvelgiant į naudojimo būdą, viešojo ir privačiojo raktų pora gali būti nekeičiama pakankamai ilgai, t. y. raktų pora skirta daugkartiniam naudojimui.	Populiariausių viešojo rakto algoritmų šifravimo sparta yra kelis šimtus kartų mažesnė nei labiausiai paplitusių simetrinio rakto sistemų. Raktų ilgis yra gerokai didesnis nei simetrinio šifravimo sistemose, siekiant užtikrinti tokį pat saugumo lygį.

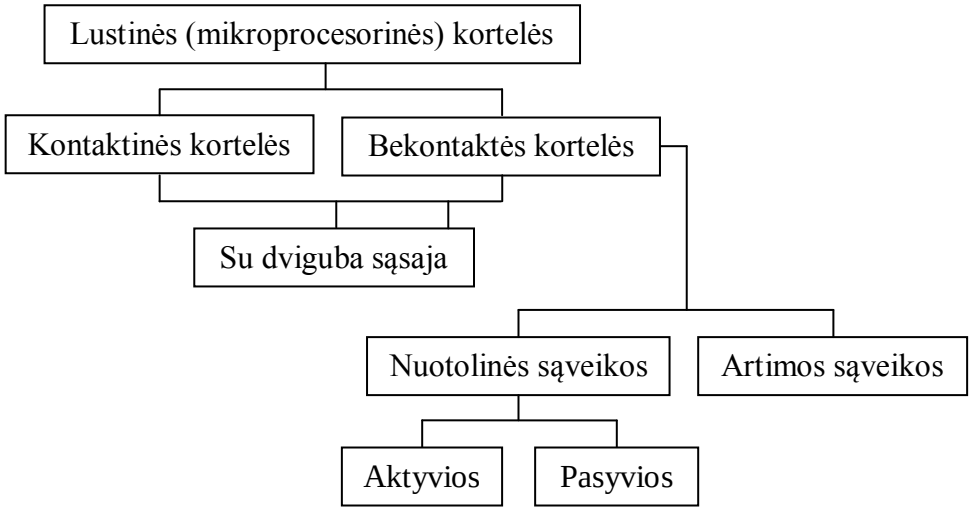
* Patikima trečioji šalis (angl. *Trusted third party*) atlieka raktų generavimo ir paskirstymo funkcijas, taip išsprendžiant raktų paskirstymo problemą.

Apibendrinant 2.4 lentelėje pateiktą informaciją galima teigti, kad kriptografinės sistemos pasirinkimas priklauso nuo to, kam ji bus naudojama. Simetrinės kriptografinės sistemos naudojamos greitam informacijos šifravimui. Asimetrinės kriptografinės sistemos privalumas, jog informacija šifruojama vienu raktu (slaptu raktu), o iššifruojama su kitu raktu (viešu raktu).

2.6. Lustinių kortelių analizė

Lustinė kortelė [19] [22] [23] (dar vadinama mikroprocesorine elektronine arba tiesiog elektronine kortele) – plastikinė, kompaktiška ir saugi duomenų saugykla su integruotu programuojamu mikroprocesoriumi. Pagrindinis šių kortelių pranašumas prieš atminties korteles yra tas, kad kortelės mikroprocesorius gali apdoroti įeinančią ir vidinėje atmintyje saugomą informaciją. Lustinės kortelės be įprastinių operacijų (įvairios finansinės operacijos ir t.t.) atlieka ir kriptografines funkcijas (autentifikavimas, identifikavimas, šifravimas, elektroninis parašas ir t.t.).

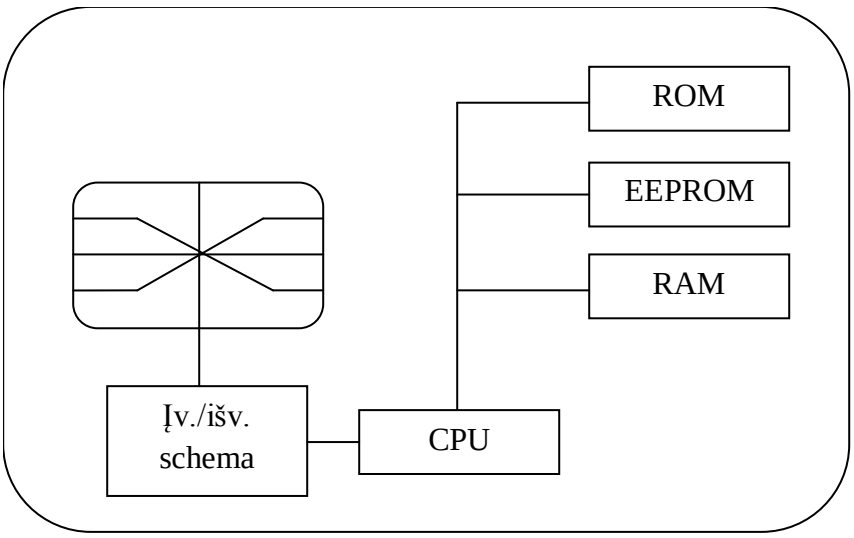
Šiuolaikinių lustinių kortelių galimybės pasiekė pirmųjų asmeninių kompiuterių lygį [19] [22] [25]. Pagrindinis skirtumas tarp šiuolaikinių lustinių kortelių ir pirmųjų asmeninių kompiuterių – lustinės kortelės neturi savo maitinimo šaltinio (išskyrus aktyvias bekontaktes lustines korteles), ir yra maitinamos skaitymo/rašymo įrenginio pagalba. Egzistuoja plati kortelių klasifikacija, tačiau mus domina tik lustinės kortelės, todėl toliau nagrinėsime tik lustinių kortelių tipus. Lustinių kortelių klasifikacija pateikiama 2.4 pav.



2.4 pav. Lustinių kortelių klasifikacija

2.6.1. Kontaktinės lustinės kortelės

Kontaktinės lustinės kortelės (2.5 pav.) – labiausiai paplitęs lustinių kortelių tipas. Šios kortelės paviršiuje turi metalinius kontaktus ryšiui su skaitymo įrenginiu užmegzti.



2.5 pav. Kontaktinės lustinės kortelės struktūra

Mikroprocesorius (CPU) yra lusto „širdis“. Visi skaičiavimai ir duomenų srautai vyksta naudojantis šiuo komponentu (kartais integruojamas ir bendras kriptografinis procesorius). Lustinė kortelė gali turėti tris skirtingos rūšies atmintis (2.5 pav.):

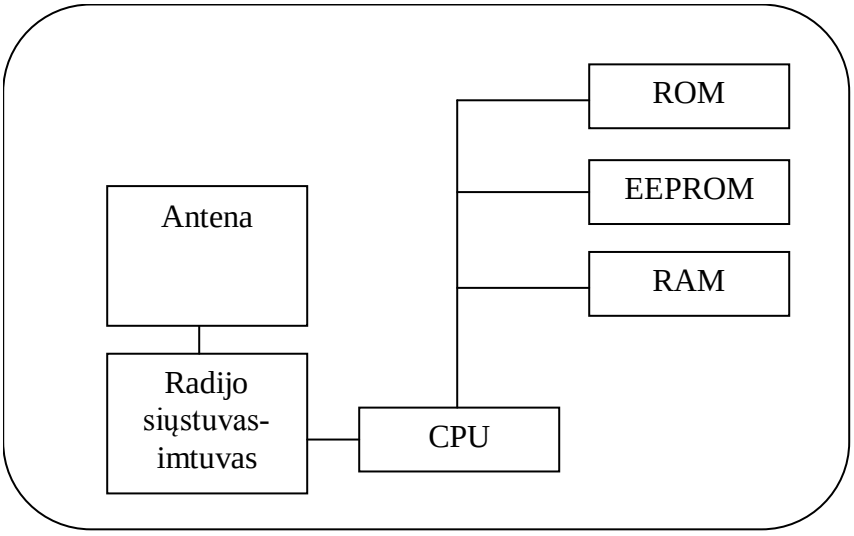
- **Skaitomoji atmintis** (angl. *ROM – read only memory*) – išliekančioji atmintis, kuri skirta tik skaitymui. Šios atminties turinys nustatomas gamybos metu.

- **Darbinė atmintis** (angl. *RAM – random access memory*) – neišliekančioji atmintis, kuri skirta saugoti programos kintamiesiems ir steko informacijai. Išjungus maitinimo šaltinį atmintis išsivalo.
- **EEPROM** (angl. *Electronically erasable programmable read only memory*) – išliekančioji atmintis, kurios turinys gali būti perrašomas ir po gamybos etapo. Priešingai nei *RAM* atminties atveju, išjungus maitinimo šaltinį, *EEPROM* atmintis neišsivalo, t.y. duomenys neprarandami.

Kontaktinių lustinių kortelių charakteristikas aprašo tarptautinis standartas *ISO/IEC 7816*. Šis standartas sudarytas iš 14 dalių [26] [22].

2.6.2. Bekontaktės lustinės kortelės

Bekontaktės lustinės kortelės (2.6 pav.) su skaitymo įrenginiu sąveikauja radijo bangomis nesinaudojant jokiu fiziniu kontaktu (naudojama fazinė signalo moduliacija).



2.6 pav. Bekontaktės lustinės kortelės struktūra

Bekontakčių lustinių kortelių trūkumai palyginus su kontaktinėmis kortelėmis yra ribotas kriptografinių funkcijų skaičius ir ribota atmintis. Be to, norint sėkmingai atlikti tam tikrą operaciją, reikia išlaikyti nustatyto dydžio atstumą tarp kortelės ir skaitymo įrenginio.

Bekontaktės lustinės kortelės dar skirstomos pagal ryšio nuotolį:

- **artimo ryšio** (angl. *Proximity, close coupling cards*) korteles reikia priglausti prie nuskaitymo įrenginio 2 mm atstumu. Šių kortelių pranašumas prieš kontaktines lustines korteles yra tai, kad jos neturi besidėvinčių kontaktų ir turi greitesnę sąveiką tarp kortelės ir įrenginio. Pranašumas prieš tolimo ryšio korteles – užtikrina didesnį duomenų perdavimo saugumą, nes neleidžia „pasiklausymo“ atakų;
- **tolimo ryšio** (angl. *Remote couple cards*) kortelės.

Artimo ryšio bekontakčių lustinių kortelių charakteristikas aprašo tarptautinis standartas *ISO/IEC 14443* (standartas sudarytas iš 4 dalių [27]), tuo tarpu tolimo ryšio korteles aprašo standartas *ISO/IEC 15693* (standartas sudarytas iš 3 dalių). Standartuose nurodyti bekontakčių lustinių kortelių veikimo atstumai (artimo ryšio – 10 cm ir mažiau, tolimo ryšio – nuo 1 m iki 1,5 m), veikimo dažniai, komunikavimo protokolai ir kitos svarbiausios charakteristikos.

Bekontaktės lustinės kortelės dar skirstomos pagal maitinimo šaltinį:

- Pasyvios kortelės neturi maitinimo šaltinio (energijos gauna per elektromagnetinį lauką, kuris indukuoja elektros srovę antenoje);
- aktyvios kortelės priešingai nei pasyvios, turi savo maitinimo šaltinį.

2.6.3. Hibridinės lustinės kortelės ir lustinės kortelės su dviguba sąsaja

Hibridinės lustinės kortelės – kontaktinių ir bekontakčių lustinių kortelių derinys. Tarp lusto, turinčio kontaktinę sąsają, ir lusto su bekontakte sąsaja nėra ryšio, duomenų mainai galimi tik per išorinį įrenginį.

Kortelės su dviguba sąsaja – vienas lustas turi ryšį su kontaktine ir bekontakte sąsaja, bendra prieiga prie atminties ir mikroprocesoriaus.

2.7. Lustinių kortelių sauga

Lustinės kortelės yra populiarus sukčių taikinys, nes:

- atlikus sėkmingą ataką, įsilaužėlis gauna materialinę naudą;
- lustinės kortelės yra pigios ir lengvai įsigyjamos, todėl įsilaužėliui sudaromos sąlygos kortelės saugumo apėjimo „treniruotėms“;
- lustinės kortelės yra mobilios, todėl įsilaužėlis gali jomis pasinaudoti bet kurioje aplinkoje.

Nors gamintojai skiria didelį dėmesį kortelių saugumui, tačiau 100% kortelės saugumo užtikrinti neįmanoma ir tuo naudojasi įsilaužėliai. Galima išskirti tris pagrindinius atakų tipus [28]:

- **loginės atakos** – spragos, kurios aptinkamos kortelės programinėje įrangoje;
- **fizinės atakos** – spragos, kurios aptinkamos naudojantis kortelės analize arba modifikavimu;
- **išorinių kanalų atakos** (angl. *Site channel attacks*) – spragos, kurios aptinkamos naudojantis fizikiniais reiškiniais analizuojant ar modifikuojant lustinės kortelės elgseną.

2.7.1. Loginės atakos prieš lustines korteles

Loginės atakos susijusios su paslėptų ydų išnaudojimu. Galimos loginės saugos grėsmės:

- Paslėptų komandų išnaudojimas (angl. *Hidden commands*) – komandos, kurios pasilikę nuo prieš tai naudotos programos ir yra aktyvios, tačiau pasislėpę. Šios komandos gali būti panaudotos perimant ar modifikuojant duomenis lustinėje kortelėje.
- Parametrų sugadinimas ir buferio perpildymas (angl. *Parameter poisioning and buffer overflow*) – neleistinos parametro reikšmės ar jo ilgio panaudojimas. Gali iššaukti rezultatą.
- Prieiga prie failų (angl. *File access*) – komandoms suteikiama daugiau prieigos teisių prie specifinių failų, nei to reikia.
- Kenkėjiškos programos (angl. *Malicious applets*) – parsisiuntus kenkėjišką programą ar programą su ydomis galima pažeisti kitas programas esančias lustinėje kortelėje.
- Komunikacijos protokolas (angl. *Communication protocol*) – siunčiant tinkamai suformuotas žinutes, galima apgauti kortelę. Galimas dalinis ar visos atminties atskleidimas.
- Kriptografiniai protokolai, jų projektavimo ir realizavimo klaidos (angl. *Crypto-Protocol, Design and Implemention*) – naudojami nepatikrinti protokolai, kurie gali turėti vienokias ar kitokias ydas.

Loginių atakų sėkmė stipriai priklauso nuo programos sudėtingumo. Seniai žinoma tiesa – kuo daugiau kodo eilučių ir kuo jis sudėtingesnis, tuo didesnė tikimybė, jog tame kode bus padaryta klaidų. Priemonės ir strategijos, padedančios apsisaugot nuo loginių atakų:

- programinę įrangą kurti nedideliais funkciniais blokais, kurie yra lengvai suprantami ir greitai patikrinami;
- matematinių modelių naudojimas funkcijų patikimumui įrodyti;
- testavimas;
- naudoti jau sukurtas programas ar metodus, kurie yra išbandyti;
- objektiškai orientuotos programavimo kalbos naudojimas;
- vertinimo laboratorijų paslaugų naudojimas, sertifikatų gavimas ir t.t.

Nepaisant visų išsakytų apsaugos priemonių, lustinės kortelės ir toliau lieka pažaidžiamos loginių atakų metu. Programų sudėtingumas lemia didesnį saugumo spragų skaičių. Kruopštus projektavimas, realizavimas ir testavimas gali tik sumažinti šių spragų skaičių.

2.7.2. Fizinės atakos prieš lustines korteles

Fizinės atakos pagrįstos fiziniu kortelės lusto tyrinėjimu. Galimos fizinės saugos grėsmės:

- Cheminiai tirpikliai, ėsdinimas ir dažymo medžiagos (angl. *Chemical solvents, etching and staining materials*) – atakos pagrįstos apsauginio lusto sluoksnio pašalinimu. Pašalinus apsauginė sluoksnį galima atlikti optinę (naudojami mikroskopai) ir elektroninę analizę.
- Zondavimo stotys (angl. *Probe station*) – zondavimo adatos pridedamos prie lusto perdavimo laidų (sukuriami nauji duomenų perdavimo kanalai).

Nors fizinės atakos yra labai pavojingos, tačiau jos turi ir savų trūkumų. Visų pirma fizinės atakos yra invazinės ir dažniausiai naikinančios. Kortelė po fizinės atakos dažniausiai tampa nepanaudojama. Antra, tokioms atakoms dažnai reikia sudėtingos ir brangios laboratorinės įrangos. Iš kitos pusės, stambios laboratorijos siūlo savo įrangą, ar netgi padeda atlikti kortelių analizę, kaip paslaugą, todėl žmonėms bandantiems įsilaužti į lustines korteles fiziškai tampa netgi paprasčiau. Lustinių kortelių gamintojams žinomos fizinės atakų formos, todėl bandoma jas apsaugoti įvairiais metodais:

- tranzistorių dydis sumažėjo iki tokio dydžio, kad jie tapo per maži stebėti ir analizuoti mikroskopu bei per maži, kad būtų galima panaudoti zondavimo adatas;
- atakoms „jautrios“ duomenų perdavimo linijos paslepiami po keletu sluoksnių;
- naudojamas apsauginis lusto sluoksnis, kurį pažeidus ištrinama atmintis;
- naudojami sensoriai, kurie stebi aplinkos parametrus ir esant įsibrovimo požymiams užblokuoja lustą;
- duomenų perdavimo linijų „supynimas“;
- funkcinių blokų sumaišymas (angl. *Glue logic*).

Yra daug būtų užkirsti kelią fizinėms atakoms, tačiau ne visi lustinių kortelių gamintojai jas naudoja dėl ekonominių priežasčių. Fizinei apsaugai nuo galimo padirbinėjimo naudojamos priemonės:

- reljefinis įspaudas – terminiu būdu užrašomi įrašai (numeris, vardas, pavardė ir .t.t);
- hologramos;
- difrakcijos tinklelis;
- piešiniai ir užrašai, matomi ultravioletinėje šviesoje ir t.t.

2.7.3. Išorinių kanalų atakos prieš lustines korteles

Išorinių kanalų atakos skiriasi nuo fizinių atakų tuo, kad stebi luste vykstančius procesus ir fiksuoja aplinkos duomenis. Be to, šios atakos nėra invazinės. Išorinių kanalų atakos skirstomos į dvi kategorijas:

1. Išorinių kanalų analizė

   Fizikinių reiškinių analizavimas stebint elektrinės grandinės elgesį:

   - energijos suvartojimas (angl. *Power consumption*) – teoriškai gali suteikti informacijos apie duomenų apdorojimą kortelėje;
   - elektromagnetinė radiacija (angl. *Electromagnetic radiation*) – teoriškai gali suteikti informacijos apie vykstančius procesus;
   - laikas (angl. *Time*) – teoriškai gali nusakyti procesų parametrus.

2. Manipuliavimas išoriniais kanalais

   Fizikiniai reiškiniai, kurie gali sutrikdyti lusto darbą:

   - įtampa (angl. *Voltage*) – pakeitus įtampą, galima sutrikdyti lusto elgseną;
   - elektromagnetinė radiacija (angl. *Electromagnetic radiation*) – stiprus elektromagnetinis impulsas gali sugadinti lustą, arba priversti jį elgtis kitaip;
   - temperatūra (angl. temperature) – pakeitus temperatūrą galima priversti įrenginį elgtis kitaip nei įprasta;
   - Šviesa ir rentgeno spinduliai (angl. *Light and X-Rays*) – puslaidininkiai yra jautrūs šviesai, todėl paveikus juos šviesa ar spinduliais galima iškreipti įprastą lusto elgesį;
   - dažnis (angl. *Frequency*) – pakeitus mikroprocesoriaus dažnį (aukščiau maksimalios ribos ar stipriai sumažinus) galimas lusto elgsenos pasikeitimas, kuris gali atverti saugumo spragas.

Apsaugos priemonės naudojamos nuo išorinių kanalo atakų grėsmių:

- elektromagnetinės emisijos mažinimas;
- amplitudinių triukšmų didinimas;
- laikinis triukšmas, procesų pertraukimas ir t.t.
- aplinką stebintys sensoriai;

2.8. Išvados

Šiame skyrelyje pateikiamos taikomųjų programų grėsmių ir apsaugos metodų nuo piratavimo analizės išvados. Atlikus literatūros šaltinių analizę nustatėme, kad:

1) apsaugos metodai nuo piratavimo taikomi tik mokamai programinei įrangai;
2) didžiausią grėsmę programų saugai kelia atvirkštinė inžinerija;
3) derintuvai ir ardymo priemonės yra dažniausiai programinės įrangos apsaugos apėjimui naudojamos priemonės;
4) apsaugos nuo piratavimo metodai skirstomi į programinius ir aparatūrinius;
5) nuo atvirkštinės inžinerijos geriausiai apsaugo programos kodo glaudinimas ir šifravimas;
6) aparatūrinės apsaugos priemonės apsaugai naudojamos rečiau, tačiau jos žymiai padidina programinės įrangos apsaugą nuo piratavimo;
7) dažniausiai naudojama aparatūrinė apsaugos priemonė – apsaugos raktai;
8) programinės apsaugos priemonės yra pigesnės ir lankstesnės, tačiau jos pasižymi silpnesne apsauga nei esant aparatūrinėms apsaugos priemonėms;
9) tik kompleksinė apsauga, kurią sudaro tiek programinės, tiek aparatūrinės apsaugos priemonės, gali tinkamai apsaugoti nuo nelegalaus programinės įrangos naudojimo;
10) simetrinės šifravimo sistemos naudoja vieną slapta šifravimo raktą ir naudojamos kur reikia greito duomenų šifravimo;
11) simetrinių šifravimo sistemų slaptieji raktai turi būti dažnai keičiami;
12) asimetrinės šifravimo sistemos yra lėtesnės, tačiau duomenų šifravimui ir iššifravimui naudoja atskirus raktus;
13) asimetrinių šifravimo sistemų raktų pora skirta ilgalaikiam naudojimui;
14) lustinės kortelės dėl plataus jų paplitimo ir taikomų apsaugos priemonių gali būti panaudojamos programinės įrangos apsaugai nuo piratavimo;
15) būtina apsaugoti ryšio kanalą tarp lustinės kortelės ir taikomosios programos.

3. PROGRAMŲ APSAUGOS NUO PIRATAVIMO METODAS, NAUDOJANT LUSTINES KORTELES

Analizės dalyje išsiaiškinome, jog norint apsaugoti taikomąsias programas nuo piratavimo, vien tik programinių ar aparatūrinių apsaugos metodų nepakanka, todėl reikia naudoti kompleksines apsaugos priemones. Atlikus apsaugos nuo piratavimo priemonių analizę nustatėme, kad geriausia programinė apsauga nuo piratavimo – kodo šifravimas arba glaudinimas. Taip pat nuo atvirkštinės inžinerijos neblogai apsaugo kodo sumaišymo metodai. Kaip papildoma apsauga nuo piratavimo gali būti naudojami ir sisteminio parašo metodai. Iš aparatūrinių apsaugos priemonių atsižvelgiant į kainos ir apsaugos lygio santykį išskyrėme apsaugos raktus, tačiau kaip alternatyvą šiems raktams šiame darbe naudosime lustines korteles, kurios pasižymi dideliu saugumu ir nedidele kaina.

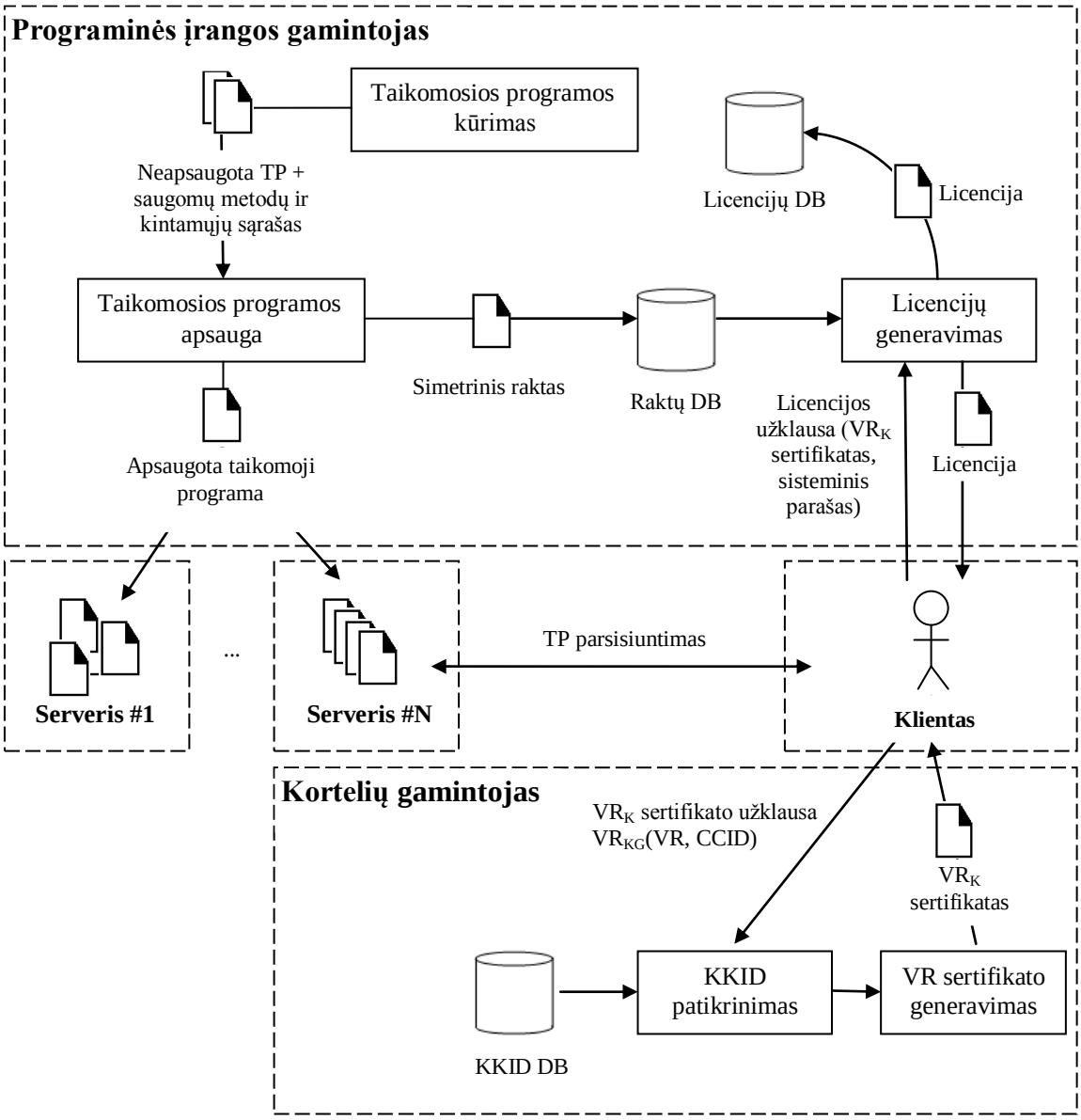
Siūlomame apsaugos nuo piratavimo modelyje, kritiniai programos moduliai (pvz. biblioteka) yra šifruojami slaptu raktu. Neužšifruoti programos moduliai apsaugomi kodo supainiojimo metodu. Programos paleidimo metu pirmiausiai patikrinama sistemos parašo teisingumas. Jei sistemos parašas sutampa su etaloniniu parašu, tuomet vykdoma programa. Neužšifruoti programos moduliai vykdomi kompiuteryje, o užšifruoti moduliai siunčiami į lustinę kortelę kur yra iššifruojami ir įvykdomi. Rezultatai grąžinami programai, o kritinis modulis ištrinamas.

3.1. Apsaugos metodo koncepcija

Siūlomas apsaugos metodas yra pagrįstas atskirų taikomosios programos modulių šifravimu. Reikia paminėti, kad ne viskas taikomosiose programose turi būt šifruojama. Šifravimo algoritmai naudojami tiems moduliams ir duomenims, kurie turi didžiausią vertę. 3.1 pav. pateikiama siūlomo apsaugos metodo nuo piratavimo koncepcija [15]. Koncepcijoje išskiriami 3 tipų aktoriai:

- **klientas** – asmuo, kuris siekia įsigyti ir naudotis sukurta taikomąją programa;
- **programinės įrangos gamintojas** – organizacija, kuri kuria ir apsaugo sukurtas taikomąsias programas bei pagal kliento pateiktą užklausą generuoja taikomosios programos naudojimosi licencijas;
- **kortelių gamintojas** – organizacija, kuri gamina lustines korteles, jas parduoda ir yra atsakinga už viešojo rakto sertifikato išdavimą kortelės turėtojui ir programinės įrangos gamintojui;

Bazinėje koncepcijoje atvaizduoti serveriai yra trečiųjų šalių paslaugos, kuriomis naudojantis platinamos sukurtos taikomosios programos. Taip pat reikia paminėti, kad bazinėje koncepcijoje buvo panaudota primityvi viešojo rakto sertifikato išdavimo schema, kur kortelių gamintojas sertifikuoja tiek kliento, tiek ir programinės įrangos gamintojo viešuosius raktus.



3.1 pav. Siūlomo saugos sprendimo koncepcija

Mūsų siūlomas taikomųjų programų apsaugos nuo piratavimo metodas remiasi lustinėmis kortelėmis, kurios naudojamos kaip saugus papildomas procesorius kritinių taikomosios programos modulių vykdymui saugioje aplinkoje. Lustinės kortelės turėtų pasižymėti kriptografinėmis savybėmis, t.y. lustinėse kortelėse turi būti leista iššifruoti programos modulius ir sugeneruoti viešojo ir privačiojo raktų porą. Svarbus aspektas kalbant apie lustines korteles – lustinėje kortelėje saugomas privatusis raktas negali būti naudojamas už jos ribų, t.y. privatusis raktas saugomas ir veiksmai su juo atliekami tik lustinėje kortelėje.

Siūlomo apsaugos sprendimo koncepcijoje (3.1 pav.) stačiakampėmis figūromis identifikuojami pagrindiniai procesai, kurie atliekami siekiant apsaugoti taikomąsias programas nuo nesankcionuoto jų naudojimo. Toliau pateikiamas šių procesų sąrašas ir trumpas aprašymas:

- **taikomosios programos kūrimas** – procesas, kurio metu programinės įrangos gamintojas sukuria taikomąją programą;
- **taikomosios programos apsauga** – procesas, kurio metu programinės įrangos gamintojas apsaugo taikomąją programą nuo nesankcionuoto jos naudojimo;
- **licencijos generavimas** – procesas, kurio metu programinės įrangos gamintojas pagal kliento atsiųstą užklausą sugeneruoja taikomosios programos naudojimosi licenciją. Licencijų generavimas plačiau aprašytas 3.1.1. skyrelyje;
- **KKID patikrinimas** – procesas, kurio metu kortelių gamintojas atsiųstą slaptą lustinės kortelės identifikatorių palygina su esančiu duomenų bazėje ir leidžia arba ne išduoti viešojo rakto sertifikatą;
- **VR sertifikato generavimas** – procesas, kurio metu kortelių gamintojas sugeneruoja lustinės kortelės viešojo rakto sertifikatą ir jį nusiunčia klientui.

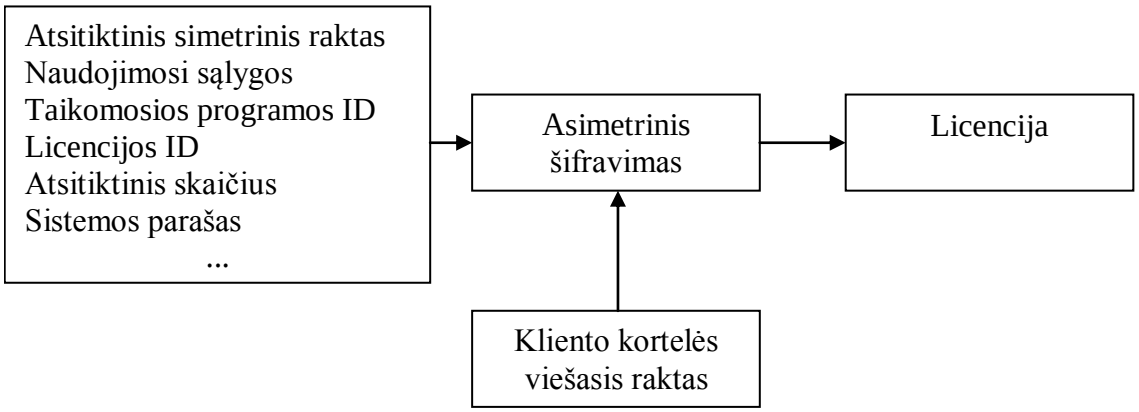
Iš koncepcijos (3.1 pav.) matome, kad programinės įrangos gamintojas sukūręs taikomąją programą atlieka taikomosios programos apsaugos procesą. Šiame procese kritiniai programos moduliai užšifruojami simetriniu raktu, kuris patalpinamas į raktų duomenų bazę. Plačiau apie taikomosios programos apsaugą aprašysime tolimesniuose skyreliuose. Apsaugota taikomoji programa talpinama į serverius iš kur gali būti parsisiųsta nemokamai. Klientas, parsisiutęs apsaugotą programą, kreipiasi į kortelių gamintoją prašydamas (kortelių gamintojui siunčia savo viešąjį raktą ir kortelės identifikacinį numerį užšifruotą kortelių gamintojo viešuoju raktu) sugeneruoti lustinės kortelės viešojo rakto sertifikatą. Kortelių gamintojas patikrinęs kortelės identifikacinį numerį išduoda arba ne viešojo rakto sertifikatą. Šis procesas atliekamas tik vieną kartą įsigijus lustinę kortelę. Visais kitais atvejais, kai jau yra sugeneruota kortelės viešojo rakto sertifikatas, šio proceso kartoti nereikia.

Turėdamas lustinės kortelės viešojo rakto sertifikatą klientas siunčia užklausą programinės įrangos gamintojui ir prašo sugeneruoti taikomosios programos licenciją. Kliento užklausą programinės įrangos gamintojui sudaro kliento viešojo rakto sertifikatas ir etaloninis sisteminis parašas, kurie užšifruojami programinės įrangos gamintojo viešuoju raktu. Licencijų generavimą plačiau panagrinėsime sekančiame skyrelyje.

Sugeneruota licencija užšifruojama kliento lustinės kortelės viešuoju raktu ir siunčiama atgal klientui. Licencijos kopija patalpinama duomenų bazėje tam, kad pametus kortelę klientas galėtų lengvai atkurti savo turimas licencijas. Gavęs sugeneruotą licenciją klientas gali naudotis apsaugota taikomąją programa.

3.1.1. Licencijos generavimo etapas

Šiame skyrelyje trumpai aprašomas licencijos generavimo etapas. Šio etapo tikslas – pagal kliento pateiktą užklausą sugeneruoti taikomosios programos naudojimosi licenciją [15]. Licencijos generavimo veiksmų seka pateikiama 3.2 pav.



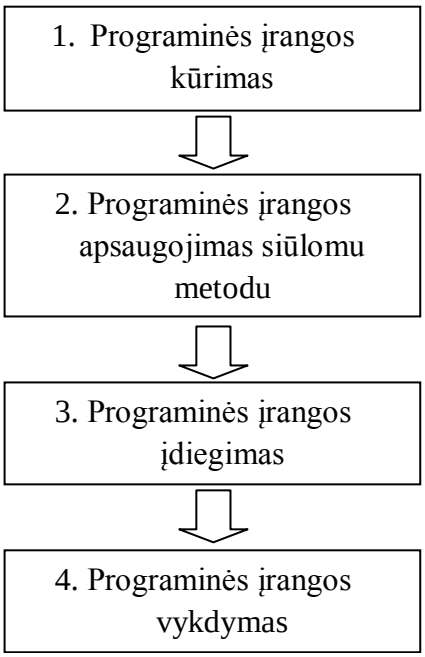
3.2 pav. Kliento licencijos generavimo schema

Taikomųjų programų kūrėjas gavęs kliento licencijos užklausą, pagal toje užklausoje pateiktus duomenis sugeneruoja naują licenciją. Kiekviena naujai sugeneruota licencija yra unikali, t.y. turi savo unikalų identifikatorių. Pagrindinis elementas, kuris turi būt kiekvienoje licencijoje yra atsitiktinis simetrinis raktas, kuriuo yra užšifruoti taikomosios programos moduliai. Kadangi kiekviena taikomosios programos versija yra šifruojama skirtingu simetriniu raktu, todėl licencijoje nurodomas taikomosios programos identifikatorius. Licencijoje taip pat patalpinamas ir kliento kompiuterio sisteminis parašas, kuris apsaugo nuo taikomosios programos nesankcionuoto naudojimosi kompiuteriuose, naudojančiuose bendrą kortelių skaitytuvą.

Visa ši informacija (3.2 pav. pateikti tik baziniai licencijos elementai) yra užšifruojama asimetriniu šifravimo algoritmu (pvz. RSA, DSA), panaudojant kliento lustinės kortelės viešąjį raktą. Taip užšifruota licencija galės būti iššifruota tik toje lustinėje kortelėje, kuri turi viešąjį raktą atitinkantį privatųjį raktą.

3.2. Apsaugos nuo piratavimo metodas

Siūlomas apsaugos nuo piratavimo metodas yra paremtas kritinių programos modulių šifravimu ir vykdymu saugioje aplinkoje. Apsaugotos programos supaprastintą gyvavimo ciklą galime atvaizduoti 4 etapais (3.3 pav.)



3.3 pav. Keturi apsaugos modelio etapai

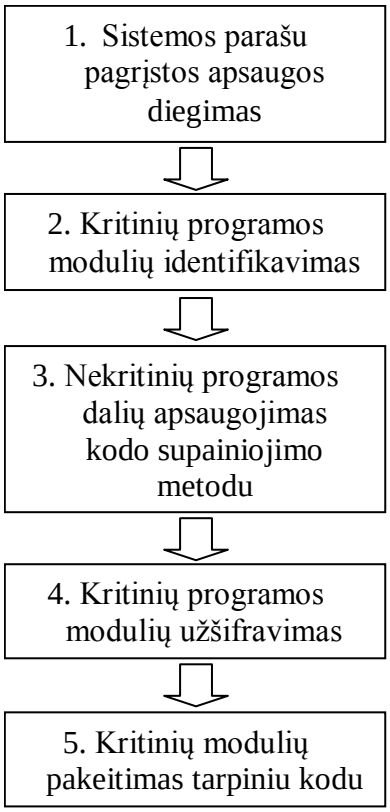
Pirmame etape projektuojama ir kuriama programinė įranga naudojantis saugiais projektavimo ir kūrimo metodais. Patvirtinus programinės įrangos sukūrimą suformuluojamas sąrašas, kuriame nurodoma ką reikia apsaugoti (algoritmų pavadinimai, duomenys). Šis sąrašas kartu su išeities kodu perduodamas sekančiam etapui – programinės įrangos apsaugojimas siūlomu metodu. Antras, trečias ir ketvirtas etapai bus nagrinėjami sekančiuose skyreliuose.

3.3. Taikomosios programos apsaugojimas siūlomu metodu

Kaip matome iš 3.3 pav., apsaugos etapas seka iš karto po to, kai yra sukuriama programinė įranga, t.y. programinės įrangos kūrimas ir apsaugojimas siūlomu apsaugos metodu yra atskiri procesai. Toks kūrimo ir apsaugos procesų atskyrimas suteikia daugiau lankstumo programinės įrangos kūrimui [13].

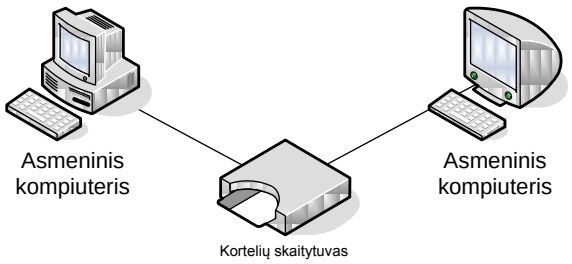
Kadangi apsaugos diegimo etapas yra kritinis saugumo atžvilgiu, todėl šiame etape dažniausiai naudojamasi automatizuotomis apsaugos diegimo priemonėmis (pvz. *JcpExternalizer* [13]). Apsaugos procesą atliekant žmogui, kai programinį kodą sudaro dešimtys ar šimtai tūkstančių kodo eilučių yra didelė tikimybė, jog bus įvelta klaidų, kuriomis galės pasinaudoti programinės įrangos piratai.

Gautas abstraktus saugotinų algoritmų ir duomenų sąrašas suvedamas automatizuotoje apsaugos diegimo priemonėje. Toliau vykstančius veiksmus galima pavaizduoti 3.4 pav. pateikta veiksmų seka.



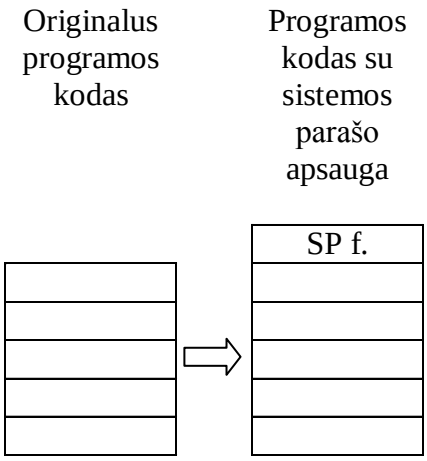
3.4 pav. Programinės įrangos apsaugos veiksmų seka

Pirmiausia saugomoje programoje įdiegiama sistemos parašu pagrįsta apsauga. Programoje diegiami tik kompiuterio komponentų identifikavimo algoritmai. Sistemos parašo tikrinimas, kaip vėliau pamatysime, atliekamas lustinėje kortelėje. Sisteminio parašo apsaugos įdiegimas padeda apsisaugoti nuo piratavimo atvejų, kai dviejų kompiuterių vartotojai naudojasi vienu kortelių skaitytuvu (3.5 pav.).



3.5 pav. Dviejų kompiuterių su bendru kortelių skaitytuvu schema

Taikomąją programą bus galima vykdyti tik tame kompiuteryje, kurio sisteminis parašas sutaps su licencijoje esančiu etaloniniu parašu, kuris sugeneruojamas taikomosios programos diegimo metu. Programinės įrangos kodo pasikeitimai po sistemos parašo apsaugos įdiegimo pavaizduoti 3.6 pav.



3.6 pav. Programos kodo pasikeitimas po sistemos parašo apsaugos įdiegimo

Toliau atliekamas kritinių programos modulių identifikavimas. Kaip jau minėjome šio skyrelio pradžioje, šį procesą atlieka automatizuotos programinės priemonės. Vienas iš algoritmų, kuriuo naudojantis identifikuojami kritiniai programos moduliai yra pateikiamas 3.7 pav.

input : $P \in \wp(L)$, $S_c \in VBB_P$, $S_d \subseteq PP \times Var$
output: $P_c \in VBB_P$, $P_d \subseteq PP \times Var$

$P_c \leftarrow \bigcup_{sc \in P_c} \{x \mid x \in sc\}$;
$P_d \leftarrow \emptyset$;
foreach *value* $v \in S_d$ **do**
  $P_d \leftarrow P_d \cup$ `Correlated`(v);
foreach *program point* $pp \in PP$ **do**
  if `Write`$(pp) \cap P_d \neq \emptyset$ **or** `Read`$(pp) \cap P_d \neq \emptyset$ **then**
    $P_c \leftarrow P_c \cup \{pp\}$;
  else
    foreach $t \in PP$ *s.t* $t \prec pp$ **and** $degree(t) > 1$ **do**
      if $pp \in DZ(t)$ **and** `Write`$(t) \cap P_d \neq \emptyset$ **or** `Read`$(t) \cap P_d \neq \emptyset$ **then**
        $P_c \leftarrow P_c \cup \{pp\}$;
//Build virtual boxes from P_c;
$VBBs \leftarrow \emptyset$;
foreach $cc \in$ `ConnectedComponents`(P_c) **do**
  $vbb \leftarrow cc$;
  $eps \leftarrow \{pp \in cc \mid pred(pp) \cap cc = \emptyset\}$;
  $ep_{vbb} \leftarrow$ `FirstCommonAncestor` (CFG_P, eps);
  $vbb \leftarrow vbb \cup \{ep_{vbb}\}$;
  foreach $ep \in eps$ **do**
    foreach $n \in PP$ ***s.t*** $ep_{vbb} \prec n \prec ep$ **do**
      $vbb \leftarrow vbb \cup \{n\}$;
  $VBBs \leftarrow VBBs \cup \{vbb\}$;
return $(VBBs, P_d)$

3.7 pav. Kritinio kodo identifikavimo algoritmas **[13]**

Šio algoritmo įvestis yra pirmame etape (3.3 pav.) sudarytas abstraktus saugotinų modulių ir duomenų sąrašas, o išvestis – konkretūs moduliai ir duomenys, kuriuos reikia apsaugoti.

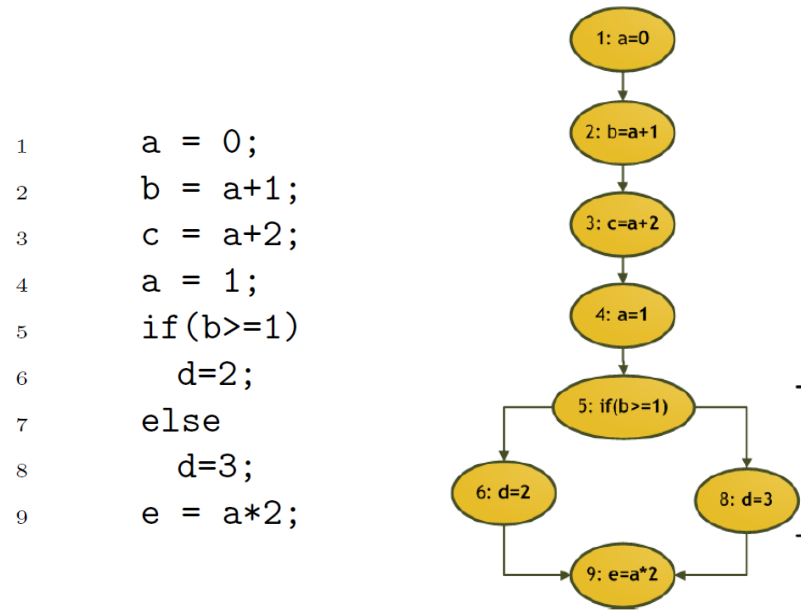
Toliau pateiksime šio algoritmo veikimo pavyzdį. Sakykime, kad identifikavimo algoritmu mes norime identifikuoti visas kritines programos, pateiktos 3.8 pav., kodo dalis pagal abstrakčius įvesties duomenis:

- saugomi moduliai: $S_c = \{\{9\}\}$. Nurodome, kad norime apsaugoti 9 eilutėje esantį algoritmą ar kodo eilutę;
- saugomi duomenys (kintamieji): $S_d = \{(a, 3)\}$. Nurodome kad norime apsaugoti kintamąjį *a*, esantį 3 kodo eilutėje.

Įvykdžius algoritmą gausime, kad mums reikia apsaugoti šiuos:

- duomenis (kintamuosius): $P_d = \{(a, 3), (a, 2), (b, 2), (a, 1), (c, 3), (b, 5), (d, 6), (d, 8)\}$ ir šiuos
- modulius (kodo eilutes): $P_c = \{(1, 2, 3), (5, 6, 8, 9)\}$

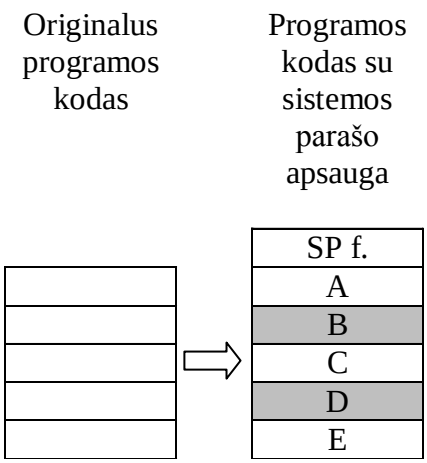
Pritaikius identifikavimo algoritmą paprastai programai (3.8 pav.) matome, kad iš devynių kodo eilučių reikia apsaugoti net 8. Pokyčiai po identifikavimo veiksmų atlikimo pavaizduoti 3.9 pav.



3.8 pav. Saugomos programos kodas ir kontrolės srauto grafikas **[13]**

Tiesa, aštuonias iš devynių kodo eilučių turėsime apsaugoti tuomet, kai naudosime labai detalų identifikavimo procesą. Literatūroje [13] minima, kad galima pasirinkti ne tokį detalų identifikavimo procesą, tačiau taikomosios programos apsauga bus atitinkamai silpnesnė. Be to, reikia paminėti kad

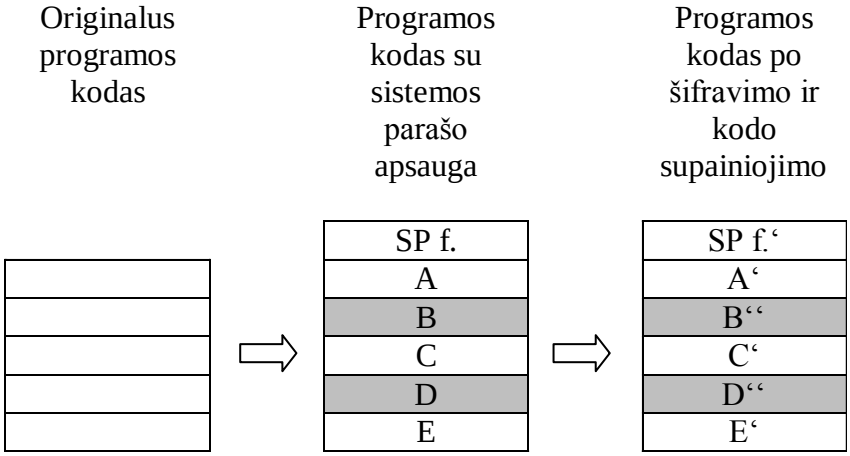
labai detalus identifikavimo procesas užtrunka labai ilgai, todėl jei nereikalaujama labai aukšto saugumo lygio, sumažinus identifikavimo detalumo lygį taikomoji programa bus apsaugota greičiau.



3.9 pav. Pokyčiai po identifikavimo veiksmų atlikimo

Šiame paveiksle (3.9 pav.) tamsesne spalva pažymėtos kodo eilutės (B ir D) yra kritinės užduotų parametrų, kuriuos identifikavome pirmame etape (3.3 pav.), atžvilgiu. Identifikavus kritinius programos modulius toliau vyksta jų apsaugojimas.

Atsižvelgiant į simetrinių ir asimetrinių šifravimo algoritmų privalumus ir trūkumus, kritinių modulių šifravimui naudosime simetrinius algoritmus, kadangi jie žymiai spartesni nei asimetriniai. Kodas, kuris buvo pažymėtas kaip nekritinis, papildomai apsaugomas kodo sumaišymo metodu. Tokių papildomų apsaugos priemonių įdiegimas dar labiau apsunkina atvirkštinės inžinerijos naudojimą. Pokyčiai po kritinių modulių šifravimo ir kodo supainiojimo priemonių įdiegimo pavaizduoti 3.10 pav.



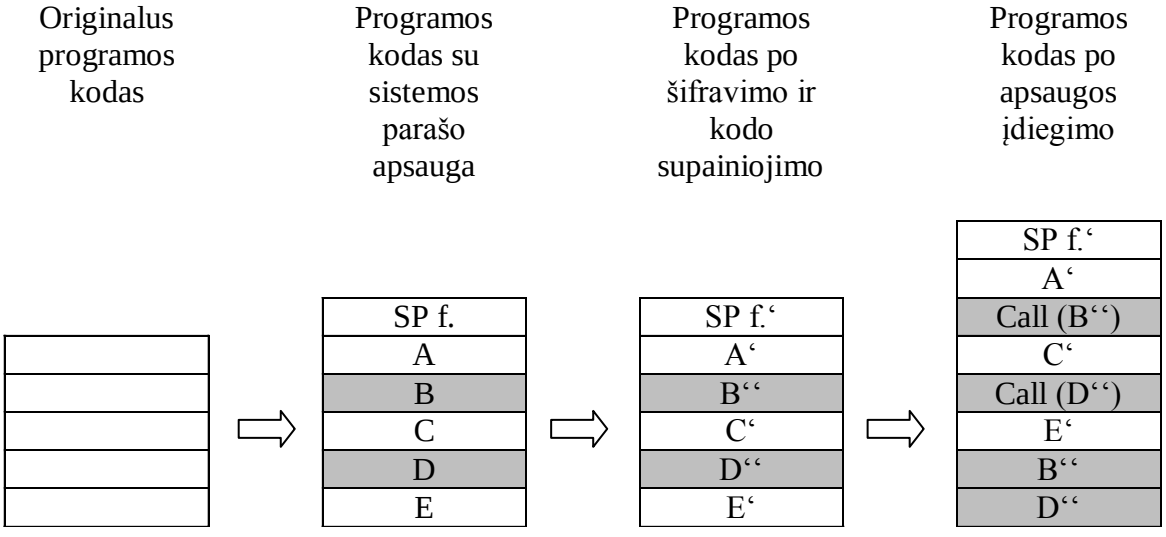
3.10 pav. Pokyčiai po kritinių modulių šifravimo ir kodo supainiojimo veiksmų atlikimo

Šiame paveiksle (3.10 pav.) nekritiniams programos moduliams (SP f, A, C, E) pritaikyta kodo sumaišymo technika (SP f‘, A‘, C‘, E‘). Tamsesne spalva pažymėtiems kritiniams programos moduliams (B, D) pritaikyta kodo šifravimo technologija (B‘, D‘). Keletas simetrinių šifravimo algoritmų pateikiami 3.1 lentelėje.

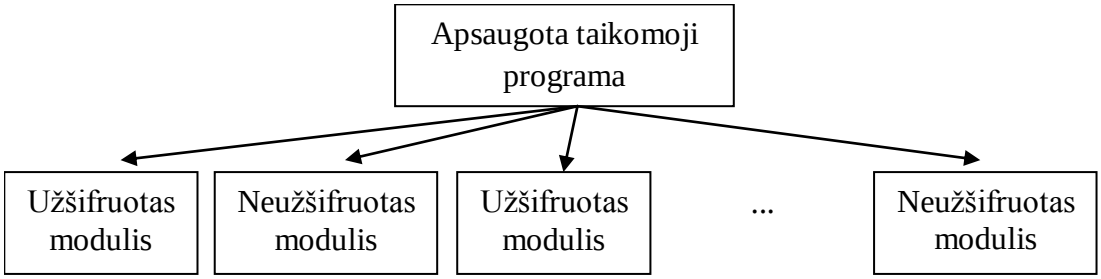
3.1 lentelė. Simetriniai šifravimo algoritmai ir jų raktų ilgiai

Eil. Nr.	Algoritmas	Rakto ilgis (bitais)
1	DES	64
2	TripleDES	192
3	IDEA	128
4	BlowFish	32-448
5	AES (RijnDael)	128, 192, 256
6	RC6	128, 192, 256

Toliau seka užšifruotų kritinių programos modulių pakeitimas tarpiniu kodu (3.11 pav. tamsesne spalva pažymėtos „*Call (B‘‘)*“ ir „*Call (D‘‘)*“ eilutės) bei papildomų funkcijų diegimas (pvz. PIN kodo apsauga). Užšifruoti moduliai „iškerpami“ (angl. *Extract*) iš programos, vietoj jų įterpiant tarpinį kodą. Tarpinis kodas vykdant kritinį programos modulį atlieka valdymo funkcijas ir siunčia šį kodą į lustinę kortelę (2.3 pav.). Iškirptas užšifruotas programos modulis išsaugomas kartu su taikomąją programa. Pokyčiai po tarpinio kodo įdiegimo pavaizduoti 3.11 pav. Apsaugotos taikomosios programos struktūra pateikiama 3.12 pav.



3.11 pav. Programos kodas po apsaugos įdiegimo

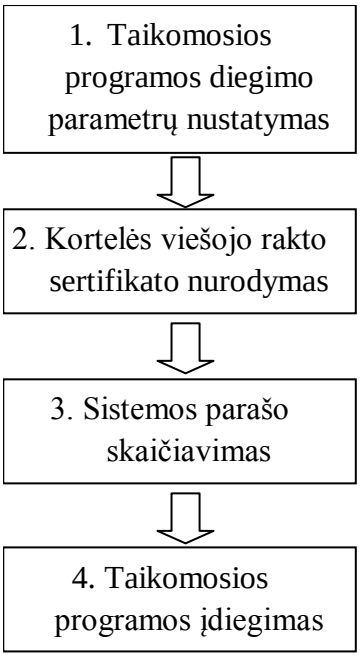


3.12 pav. Apsaugotos taikomosios programos struktūra

Šis apsaugos metodas yra patogus tuo, kad turint vieną lustinę kortelę galima naudotis neribotu skaičium apsaugotų programų, t.y. nereikia pirkti kiekvienai taikomajai programai atskiros lustinės kortelės. Trūkumas susijęs su daugybės programų naudoji yra tas, kad vienu metu lustinėje kortelėje gali būti vykdomas tik vienas modulis. Jei lustinė kortelė vykdo modulį, tai kitos programos turi laukti kol ji atsilaisvins.

3.4. Programinės įrangos diegimas

Šiame skyrelyje aprašomas taikomosios programos įdiegimo etapas. Šis etapas nuo tradicinio taikomosios programos diegimo proceso skiriasi tuo, kad jame sugeneruojamas etaloninis sistemos parašas. Sistemos parašo apskaičiavimas leidžia apsisaugoti nuo taikomųjų programų piratavimo, kuomet klientas nusipirkęs vieno kompiuterio licenciją taikomąją programą diegia keliuose kompiuteriuose. Programinės įrangos diegimo procesą galime pavaizduoti 4 žingsnių seka, kuri pateikiama 3.13 pav.



3.13 pav. Supaprastintas taikomosios programos diegimo procesas

Taikomosios programos diegimo proceso pradžia (1 žingsnis) praktiškai nesiskiria nuo tradicinio diegimo proceso. Klientas inicijuoja diegimą paleisdamas taikomosios programos įdiegėją, nurodo kur programa bus diegiama, kokius komponentus reikia diegti ir t.t. Be tradicinių nustatymų diegimo procese, klientas turi nurodyti lustinės kortelės viešojo rakto sertifikatą (2 žingsnis). Atlikus visus reikiamus nustatymus ir paleidus diegimo procesą pradedamas sistemos parašo generavimas (3 žingsnis). Sistemos parašui generuoti šiame darbe pasinaudojome kito autoriaus aprašytu algoritmu (3.14 Pav.), apie kurį plačiau galite pasiskaityti [12] [29].

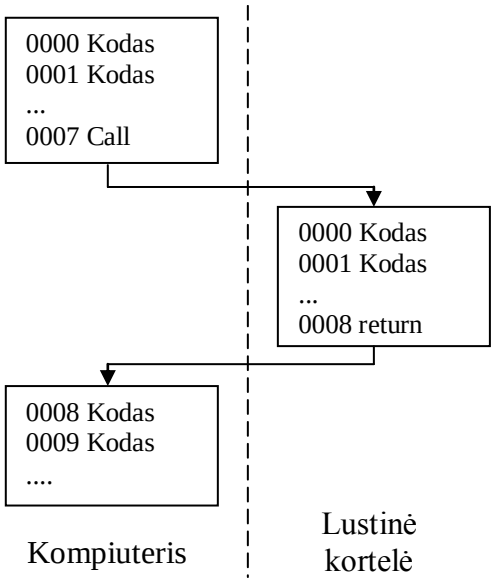
```
input: KDS, m // m komponentų sąrašas
output: SP // sistemos parašas
l = maxlength KDS // ilgiausias komponento duomuo
for j = 1 to l do
  SP (j) = KDS (1, j)
end for
for i = 2 to m do
  for j = 1 to l-1 step 2 do
    SP (j) = SP (j) XOR KDS (i, j)
    SP (j+1) = SP (j+1) OR KDS (i, j+1)
  end for
 end for
```

3.14 pav. Sistemos parašo generavimo algoritmas **[12]**

Suskaičiavus sistemos parašą, šis užšifruojamas lustinės kortelės viešuoju raktu ir saugomas kompiuteryje užšifruotoje formoje. Sekančiu žingsniu (4 žingsnis) atliekamas pačios taikomosios programos diegimas, kuris nesiskiria nuo tradicinio programinės įrangos diegimo proceso.

3.5. Programos kodo vykdymas

Šiame skyrelyje aprašomas taikomosios programos vykdymo etapas. Supaprastinta taikomosios programos vykdymo schema pateikiama 3.15 pav.

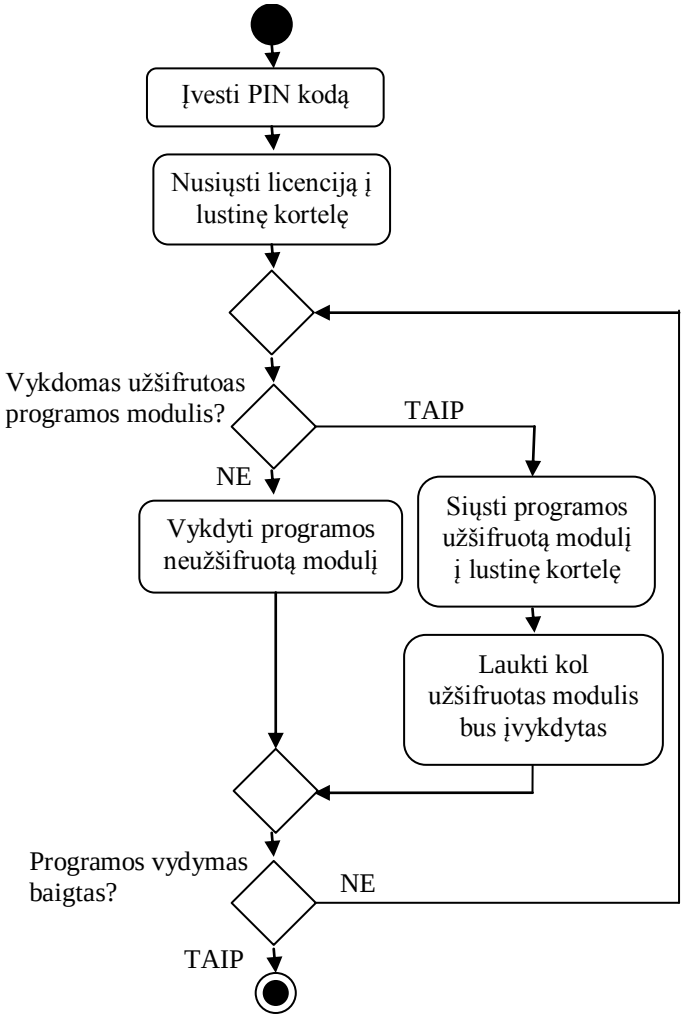


3.15 pav. Supaprastinta taikomosios programos vykdymo schema

Programos vykdymas išskaidomas į kritinio kodo vykdymą lustinėje kortelėje ir likusio kodo vykdymą kompiuteryje. Iš programos kodo vykdymo kompiuteryje veiklios diagramos (3.16 pav.) matome, jog pirmiausiai yra įvedamas PIN kodas, kuris siunčiamas į lustinę kortelę patikrinimui. Tai papildoma funkcija-apsauga, kurią įdiegėme taikomosios programos apsaugojimo metu (3.3. skyrelis). Taikant šią apsaugos priemonę, tik kortelės savininkas galės pasinaudoti jos teikiamu funkcionalumu.

Teisingai įvedus PIN kodą atliekamas licencijos siuntimas į lustinę kortelę. Lustinės kortelės turi nedaug vidinės atminties [19] [13], todėl licencija, užšifruota lustinės kortelės viešuoju raktu, saugoma kompiuteryje. Kadangi užšifruotą licenciją galima iššifruoti tik viešajam raktui atitinkančiu privačiu raktu, todėl licencija gali būt iššifruota tik lustinėje kortelėje.

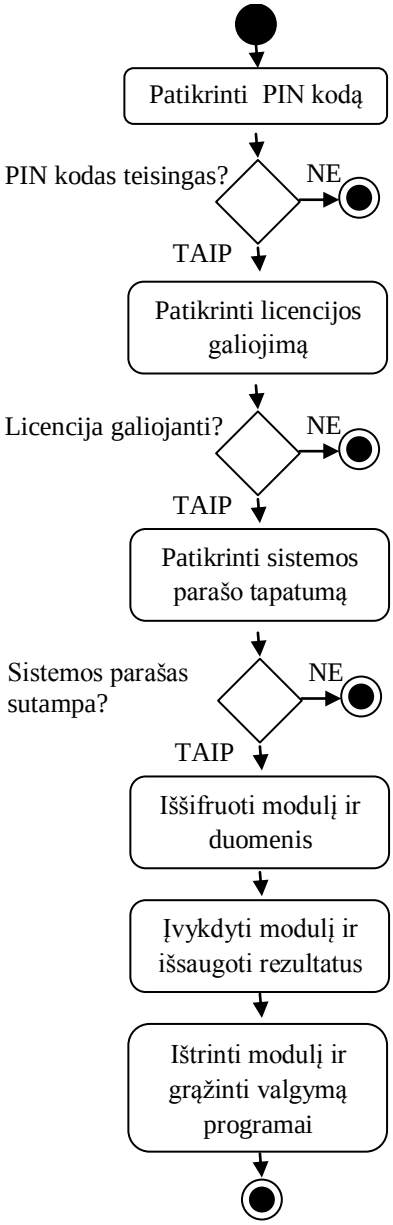
Toliau vykdant programą tikrinama, ar programos modulis yra užšifruotas. Jei programos modulis neužšifruotas, tuomet jis vykdomas kompiuteryje. Jei vykdomas programos modulis yra užšifruotas, tuomet jis siunčiamas į lustinę kurtelę. Užšifruoto modulio vykdymo lustinėje kortelėje schema pateikiama 3.17 pav. Nusiuntus užšifruotą modulį į kortelę laukiama kol jis bus įvykdytas. Jei daugiau nebėra vykdomų modulių, tuomet išsaugomi rezultatai ir uždaroma programa. Kitu atveju iš naujo tikrinama ar programos modulis yra užšifruotas, ir atitinkamai jis vykdomas kompiuteryje arba lustinėje kortelėje.



3.16 pav. Programos kodo vykdymo kompiuteryje veiklos diagrama

Lustinėje kortelėje be modulio iššifravimo ir vykdymo atliekama papildomos apsaugos funkcijos. Pirmiausiai yra patikrinama ar suvestas PIN kodas yra teisingas. Tik esant teisingam PIN kodui galima atlikti licencijos iššifravimo veiksmus ir patikrinti jos galiojimą.

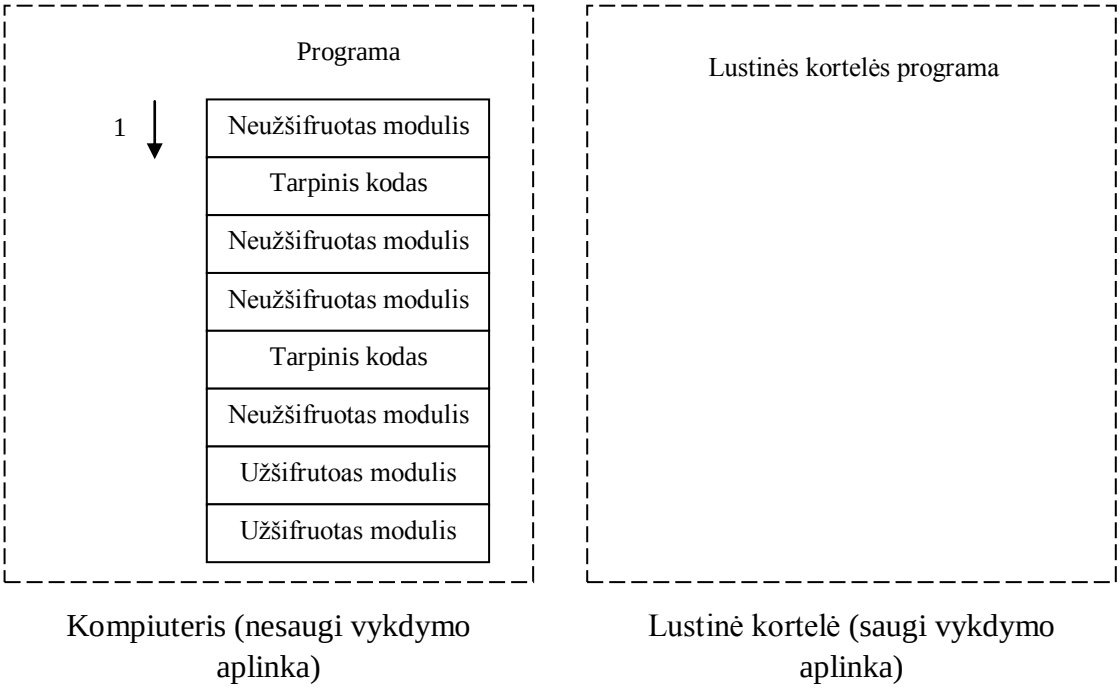
Tik turint galiojančią licenciją galima patikrinti sistemos parašą. Gavus kompiuterio komponentų duomenis sistemos parašas skaičiuojamas naudojantis 3.14 pav. pateiktu sisteminio parašo skaičiavimo algoritmu. Šis parašas palyginamas su etaloniniu parašu, kuris saugomas licencijoje. Sistemos parašui sutapus, atliekamas modulio iššifravimas ir vykdymas. Išsaugojus rezultatus modulis ištrinamas iš kortelės, kadangi kaip jau minėjome lustinės kortelės turi ribotą vidinių resursų kiekį. Po modulio ištrynimo valdymas atiduodamas kompiuteryje veikiančiai programai.



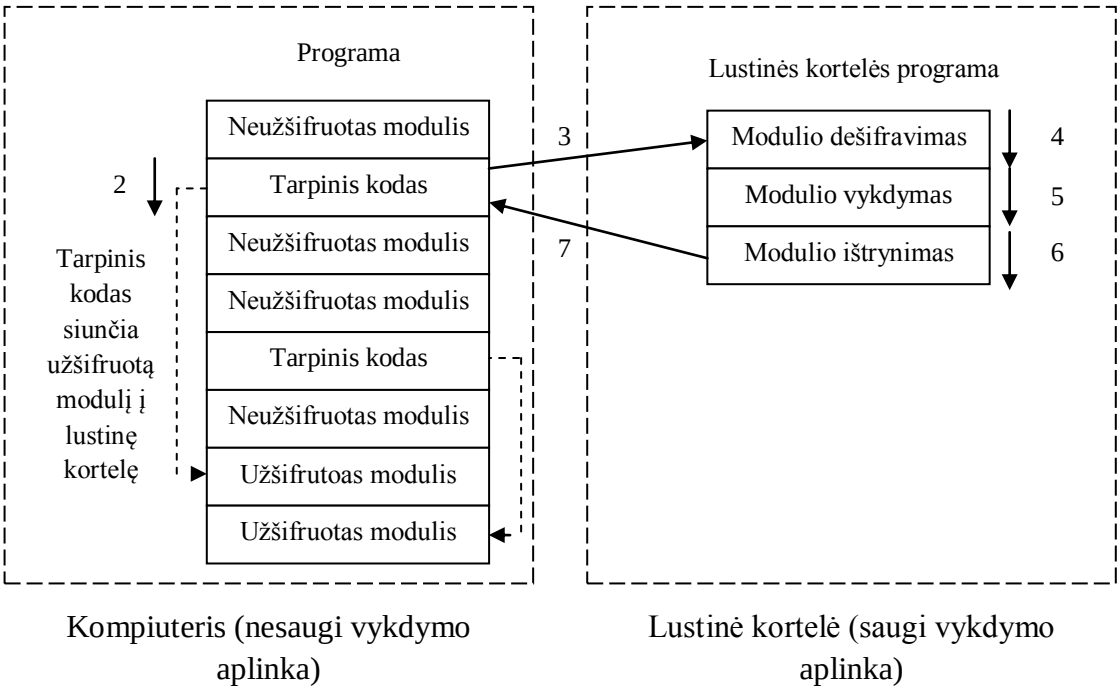
3.17 pav. Kritinio kodo vykdymo lustinėje kortelėje veiklos diagrama

Apsaugotos taikomosios programos vykdymą galime pavaizduoti šiais žingsniais:

1. taikomosios programos paleidimas ir neužšifruoto modulio vykdymas kompiuteryje (3.18 pav.).
2. Užšifruoto modulio vykdymas lustinėje kortelėje (3.19 pav.). Už užšifruoto modulio vykdymą lustinėje kortelėje ir rezultatų grąžinimą programai atsakingas tarpinis kodas. Kaip matome iš šio paveikslo, tarpinis kodas atsako tik už jam priskirtą konkretų užšifruotą modulį. Nusiuntus modulį į lustinę kortelę, ten veikianti kortelės programa modulį iššifruoja, įvykdo, išsaugo rezultatus, ištrina modulį ir atiduoda rezultatus tarpiniam kodui.
3. Sekantys taikomosios programos vykdymo žingsniai vykdomi tapačiai kaip ir 1 arba 2, priklausomai koks modulis yra vykdomas.



3.18 pav. Neužšifruoto modulio vykdymas nesaugioje aplinkoje



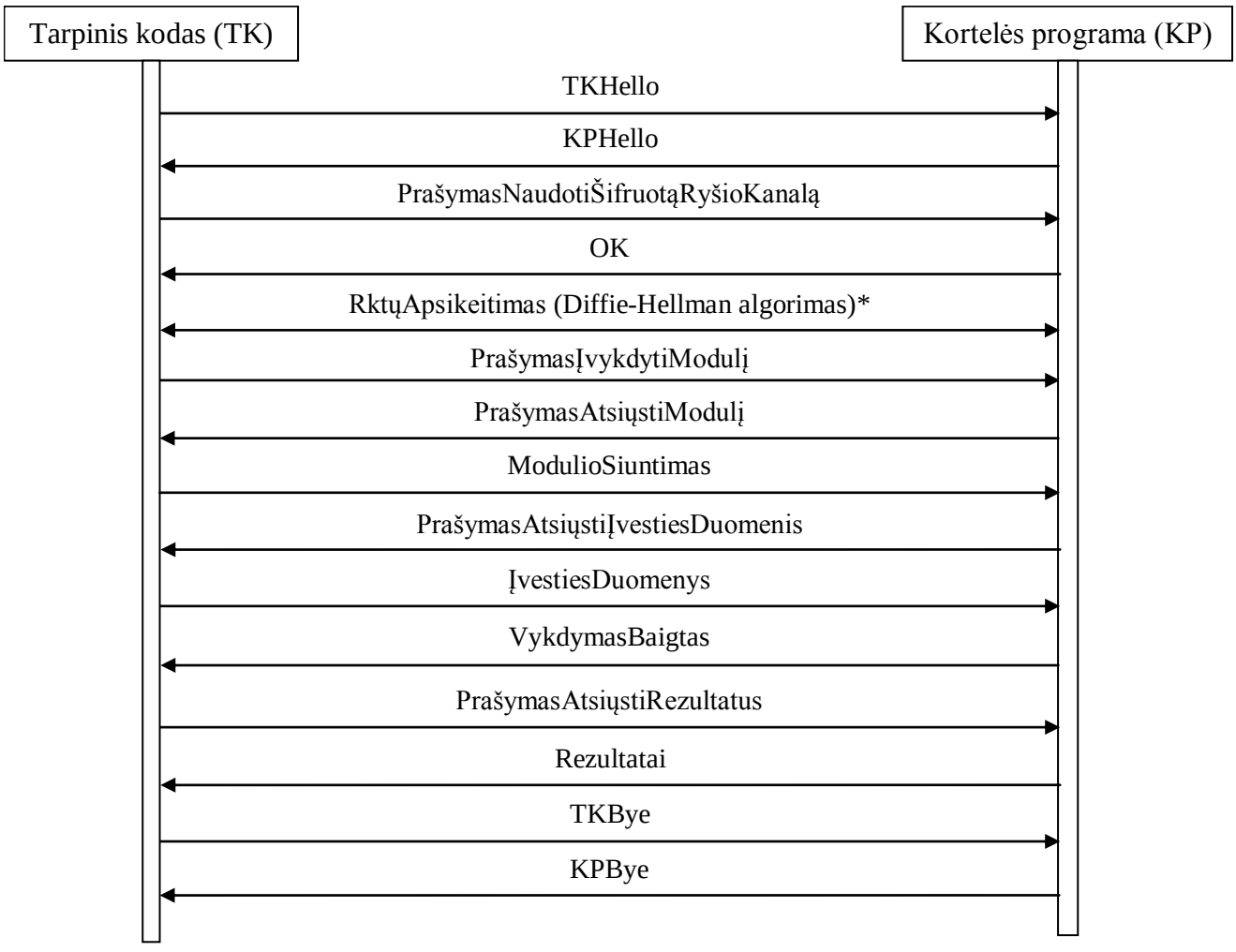
3.19 pav. Užšifruoto modulio vykdymas saugioje aplinkoje

3.6. Pranešimų apsikeitimo tarp programos ir lustinės kortelės programos protokolas

Kadangi mūsų sukurtas apsaugos nuo piratavimo metodas remiasi lustinėmis kortelėmis, kur yra vykdomi užšifruoti programos moduliai, todėl tarp apsaugotos programos ir lustinės kortelės kortelės nuolat turi būti keičiamasi įvairiais pranešimais. Iš esmės pranešimais keičiamasi tarp

tarpinio kodo, kuris atsakingas už užšifruoto modulio vykdymą kortelėje, ir lustinės kortelės programos. Pranešimų seka, kurios pagalba bendrauja šie du komponentai pateikiama 3.20 pav.

Kaip matome iš žemiau pateikto protokolo, iš pradžių abi bendraujančios pusės apsikeičia pasisveikinimo pranešimais. Sekančiu žingsniu tarpinis kodas siunčia pranešimą nurodydamas, kad visi toliau esantys pranešimai bus šifruojami *Blowfish* algoritmu (raktas 256 bitų). Jei slaptas šifravimo raktas prieš tai nebuvo sugeneruotas, arba buvo sugeneruotas, tačiau praėjo daugiau nei 20 min. po jo sugeneravimo – jis sugeneruojamas ir abi bendraujančios pusės juo apsikeičia pasinaudodamos *Diffie-Hellman* raktų apsikeitimo algoritmu [22]. Toliau keičiamasi pranešimais, kurių pagalba į lustinę kortelę nusiunčiamas užšifruotas modulis, įvesties duomenys bei po modulio įvykdymo grąžinami rezultatai. Pabaigoje abi bendraujančios pusės apsikeičia atsisveikinimo pranešimais.



3.20 pav. Pranešimų apsikeitimo protokolas

3.7. Išvados

Šiame skyrelyje pateikiamos apsaugos metodo nuo piratavimo, panaudojant lustines korteles, kūrimo išvados:

1) sumodeliuotas taikomųjų programų apsaugos metodas nuo piratavimo yra pagrįstas užšifruotų programos modulių vykdymu saugioje aplinkoje, t.y. lustinėje kortelėje;
2) kritinių programos modulių šifravimas atliekamas pasinaudojant simetriniais algoritmais dėl jų greitaveikos;
3) sumodeliuotam apsaugos metodui sukūrėme pranešimų apsikeitimo protokolą, kuris leidžia įvykdyti užšifruotą modulį lustinėje kortelėje ir grąžina rezultatus programai;
4) sumodeliuotame apsaugos metode neužšifruoti programos moduliai apsaugomi kodo supainiojimo metodu;
5) sumodeliuotame apsaugos metode įdiegėme sisteminio parašo apsaugos mechanizmą, kurio pagalba apsisaugoma nuo piratavimo atvejo, kai 2 kompiuterių vartotojai naudojasi vienu kortelių skaitytuvu ir lustine kortele;
6) taikomosios programos diegimo procesą papildėme sisteminio parašo skaičiavimo funkcija. Sugeneruotas sisteminis parašas saugomas kompiuteryje užšifruotas lustinės kortelės viešuoju raktu tol, kol jis neperkeliamas į licenciją;
7) sukurtoje apsaugos metodo koncepcijoje atvaizdavome pagrindinius procesus, kuriuos reikia įvykdyti norint apsaugoti ir pasinaudoti apsaugota taikomąją programa;
8) taikant šį apsaugos metodą viena lustinė kortelė gali būti panaudota daugiau nei 1 apsaugotos programos vykdymui (nereikia pirkti atskirų lustinių kortelių visoms apsaugotoms taikomosioms programoms);
9) vienu metu lustinėje kortelėje gali būti vykdomas tik vienas užšifruotas modulis.

4. APSAUGOS METODO NUO PIRATAVIMO, PANAUDOJANT LUSTINES KORTELES, EKSPERIMENTINIS TYRIMAS

Remiantis sumodeliuotu apsaugos nuo piratavimo metodu buvo realizuotas eksperimentinis šio metodo prototipas. Eksperimente buvo naudojama ne reali lustinė kortelė (ir jos skaitytuvas), o programinė jos realizacija. Siekiant darbe įvesti daugiau aiškumo, programinę lustinės kortelės realizaciją ir toliau vadinsime lustine kortele arba lustinės kortelės programa. Kadangi realios lustinės kortelės ir programinės jos realizacijos našumas stipriai skiriasi, todėl siekiant bent iš dalies kompensuoti šiuos skirtumus eksperimento metu apsaugota programa buvo vykdoma naudojantis dviem branduoliais (`Set Affinity -> <All Processors>`), o lustinės kortelės tik vienu branduoliu (`Set Affinity -> CPU 1`). Teoriškai toks programų vykdymo paskirstymas tarp branduolių lustinės kortelės programai turėtų suteikti dvigubai mažiau resursų nei programai vykdomai ant dviejų branduolių.

Apsaugos metodo prototipas suprogramuotas C# programavimo kalba. Kadangi .NET technologijoje radome ne visus mums reikiamus simetrinio šifravimo algoritmus, todėl šiam tikslui pasinaudojome nemokama *BouncyCastle.CryptoExt* biblioteka. Naudojantis sukurtu prototipu, apsaugos metodą vertinome greitaveikos aspektu, t.y. kiek kartų skiriasi neapsaugoto kritinio modulio vykdymas kompiuteryje nuo apsaugoto kritinio modulio vykdymo lustinėje kortelėje. Eksperimentas buvo vykdomas naudojantis šia aparatūrine įranga:

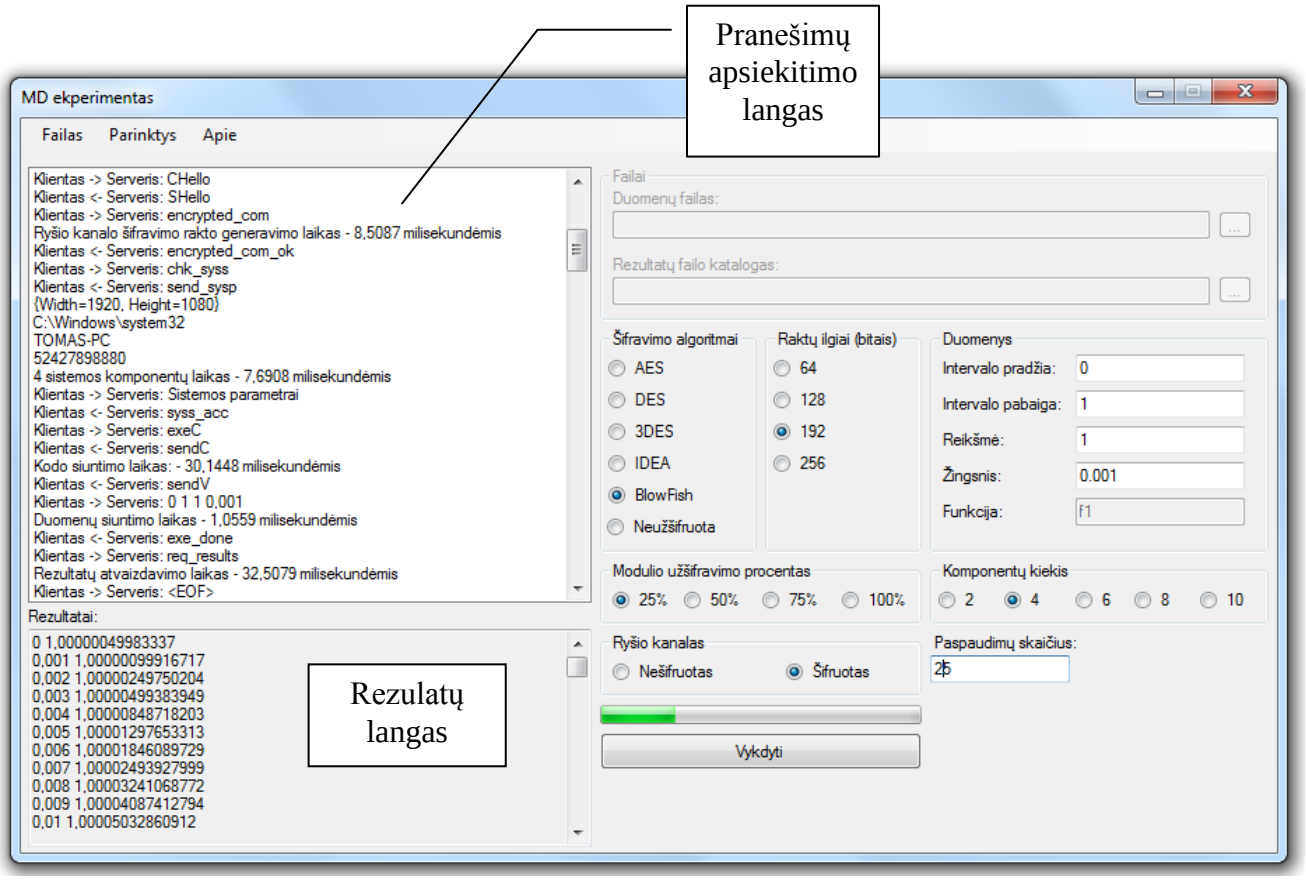
- nešiojamojo kompiuterio modelis: *Acer Extensa 5620*;
- procesorius: *Intel(R) Core(TM)2 Duo CPU T5250 @ 1.50 GHZ*;
- darbinė atmintis: *2GB*;
- operacinė sistema: *Windows 7 Professional (32 bitų) SP1*.

4.1. Grafinė realizacijos sąsaja

Šiame skyrelyje trumpai apžvelgsime grafinius programų langus. Kaip jau minėjome ankščiau, apsaugos modelį sudaro dvi programos – apsaugota programa ir lustinės kortelės programa. Apsaugota programa – valdymo programa, kurioje galime pasirinkti įvairius tyrimo parametrus. Apsaugotos programos meniu galime pasirinkti tokias parinktis:

- *Failas −> Uždaryti*. Apsaugotos programos lango uždarymas.
- *Parinktys −> Šifravimas*. Vykdomas pasirinkto modulio šifravimas.
- *Parinktys −> Iššifravimas*. Vykdomas pasirinkto modulio iššifravimas.
- *Parinktys −> Neapsaugota*. Neapsaugoto kritinio modulio vykdymas kompiuteryje.
- *Parinktys −> Apsaugota*. Apsaugoto kritinio modulio vykdymas lustinėje kortelėje.
- *Apie*. Informacija apie eksperimento sukūrimo datą, autorių ir akademinę grupę.

Toliau plačiau panagrinėsime parinkties „*Apsaugota*“ lango vaizdą (? Pav.), kadangi ši parinktis buvo dažniausiai naudojama tyrimo metu.

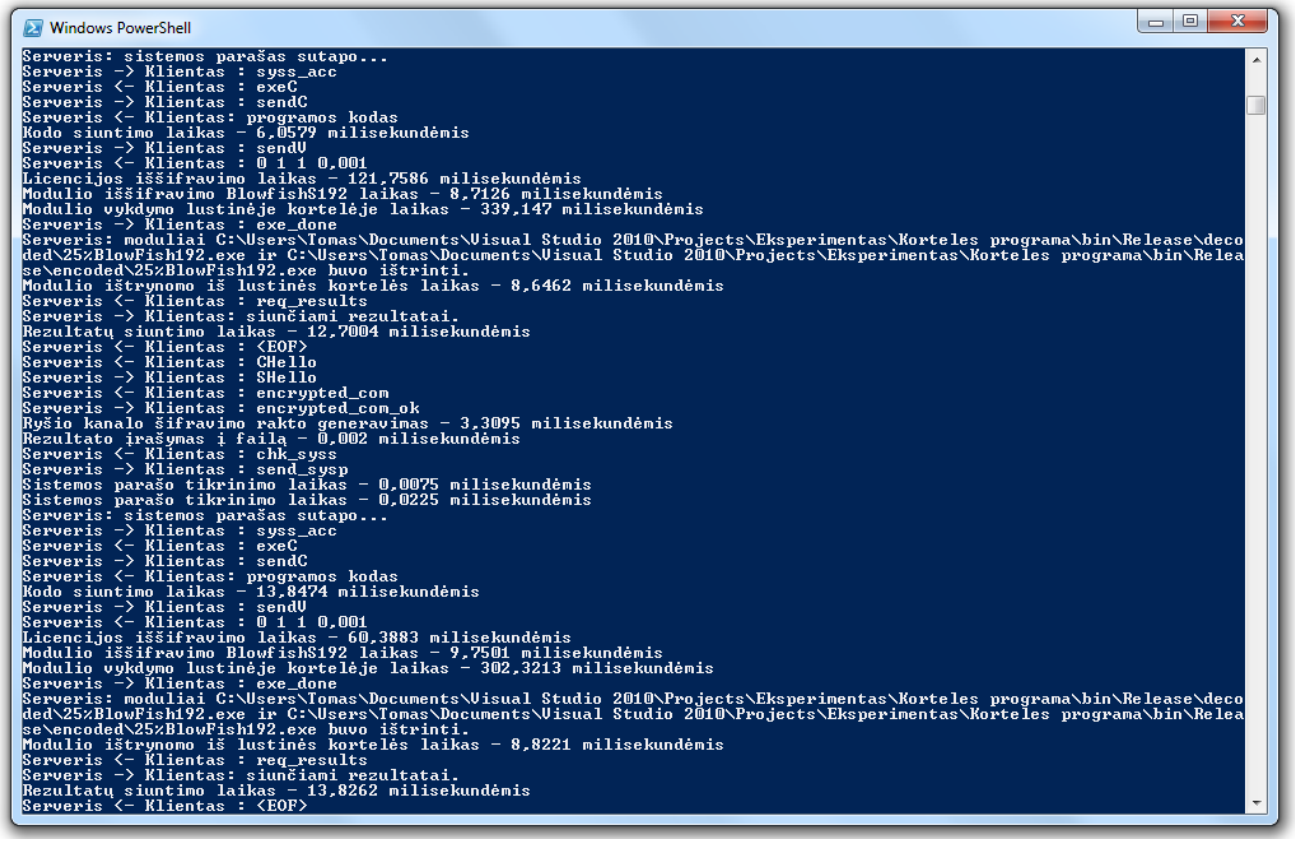


4.1 pav. Apsaugotos programos grafinė sąsaja

Kairėje pusėje esančiame pranešimų apsikeitimo lange pateikiama informacija apie vykdomus veiksmus ir jų trukmes milisekundėmis. Kad nereikėtų rankiniu būdu suvedinėti tyrimo duomenų, visos reikšmės surašomos į *apsaugota1.txt* failą. Žemiau esančiame rezultatų lange pateikiami užšifruoto modulio vykdymo rezultatai. Kadangi tyrimo metu reikėjo keisti daugybę parametrų, todėl dešinėje pusėje realizavome galimybę juos pasirinkti rankiniu būtu nekeičiant kodo. „*Šifravimo algoritmų*“, „*raktų ilgių*“ ir „*modulio užšifravimo procento*“ grupėse galime pasirinkti kokiu algoritmu, jį atitinkančiu raktu ir modulio užšifravimo procentu apsaugotą modulį norėsime vykdyti lustinėje kortelėje. „*Šifravimo algoritmų*“ grupėje pasirinkus parinktį „*Neužšifruota*“, lustinėje kortelėje vykdomas neužšifruotas modulis. „*Duomenų*“ grupėje suvedami modulio vykdymui reikalingi įvesties duomenys. Jei laukai paliekami tušti, tuomet priskiriamos numatytosios reikšmės: intervalo pradžia – 0; intervalo pabaiga – 1; reikšmė – 1; žingsnis – 0,2. Plačiau apie apie vykdomą modulį aptarsime tolimesniame skyrelyje. „*Komponentų kiekio*“ grupėje galime nustatyti iš kiek komponentų bus sudarytas sisteminis parašas. „*Ryšio kanalo*“ grupėje pasirenkame ar kanalas bus šifruotas. Pagal sumodeliuotą apsaugos metodą, kanalas turi būti šifruojamas. „*Paspaudimų*

skaičiaus“ laukelyje įvedame skaičių, kuris nusako kiek kartų reikės atlikti modulio vykdymą su tais pačiais parametrais. Paspaudus mygtuką „*Vykdyti*“ atliekamas modulio vykdymas lustinėje kortelėje su pasirinktais parametrais.

Lustinės kortelės programa – paprasta komandinės eilutės programa (4.2 pav.), kurioje pateikiama informacija apie pranešimus, po jų vykdomus veiksmus ir jų trukmes milisekundėmis. Surinkti duomenys surašomi į *apsaugota2.txt* failą.



4.2 pav. Lustinės kortelės programos grafinė sąsaja

4.2. Sistemos parašo tikrinimas

Sistemos parašo sudarymui pasinaudojome 3.14 pav. pateiktu algoritmu. Siekiant kaip galima geriau atkartoti realias tokių parašų sudarymo sąlygas, pasirinkome 10 realių sistemos komponentų:

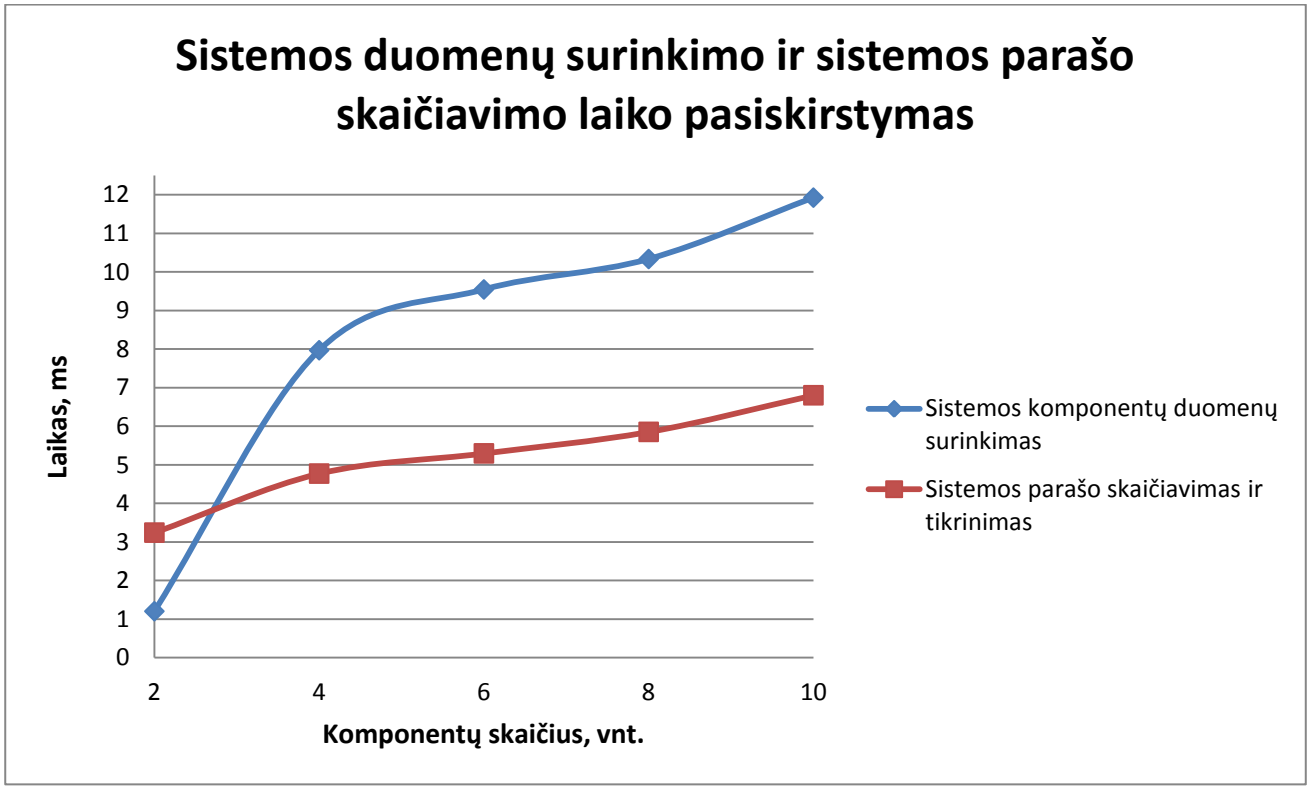
- vaizduoklio rezoliucija: *{Width=1920, Height=1080}*;
- sisteminė direktorija: *C:\Windows\system32*;
- kompiuterio pavadinimas: *TOMAS-PC*;
- C: disko talpa: *52427898880*;
- operacinės sistemos versija: *Microsoft Windows NT 6.1.7601 Service Pack 1*;
- .NET Framework versija: *4.0.30319.586*;
- vartotojo vardas: *Tomas*;
- procesoriaus duomenys: *Intel(R) Core(TM)2 Duo CPU T5250 @ 1.50 GHZ*;

- operacinės sistemos platforma: *Win32NT*;
- procesoriaus gamintojas: *GenuineIntel*.

Šiuo tyrimu siekėme nustatyti, kiek laiko užtrunka surinkti paminėtus kompiuterio duomenis, iš jų sugeneruoti sistemos parašą ir jį patikrinti su etaloniniu parašu, kuris saugomas licencijoje. Atliekant tyrimą komponentai buvo naudojami šia tvarka:

- dviejų komponentų parašą sudarė 2 pirmi elementai iš sąrašo;
- keturių komponentų parašą sudarė 4 pirmi elementai iš sąrašo;
- šešių komponentų parašą sudarė 6 pirmi elementai iš sąrašo;
- aštuonių komponentų parašą sudarė 8 pirmi elementai iš sąrašo;
- dešimties komponentų parašą sudarė visi sąraše išvardinti komponentai.

Sistemos parašo duomenų surinkimo, parašo apskaičiavimo ir patikrinimo tyrimo duomenys pateikiami priedo 7.1 lentelėje. Šiuos duomenis atitinkanti diagrama pateikiama 4.3 pav.



4.3 pav. Sistemos parašo skaičiavimas

Kaip matome iš 4.3 pav. pateiktos diagramos, sistemos duomenų surinkimas beveik visais atvejais užtrunka ilgiau nei sistemos parašo skaičiavimas ir tikrinimas. Atvejį, kai komponentų surinkimas užtruko trumpiau nei sistemos parašo skaičiavimas ir tikrinimas galime paaiškinti tuo,

kad surinkti duomenis prireikė mažiau iteracijų nei vykdant sistemos parašo skaičiavimo ir tikrinimo algoritmus.

Taip pat reikia paminėti, kad keturių komponentų surinkimas užtrunka net ~8 kartus ilgiau nei dviejų komponentų surinkimas. Tokį laiko padidėjimą galima susieti su *C:* disko talpos tikrinimu, kuriam reikia atlikti daugiau iteracijų nei kitų komponentų išgavimo metu.

Kaip matyti iš diagramos (neskaitant atvejo, kai sistemos komponentų surinkimas truko trumpiau nei sistemos parašo skaičiavimas ir tikrinimas) sistemos komponentų surinkimo laikas yra 1,7 karto ilgesnis nei sistemos parašo skaičiavimo ir tikrinimo laikas.

4.3. Modulio iššifravimas

Modulių apsaugojimas juos šifruojant yra viena iš pamatinių sumodeliuoto apsaugos metodo priemonių. Moduliui šifruoti buvo naudojami simetriniai šifravimo algoritmai, kurių sąrašas pateikiamas 4.1 lentelėje.

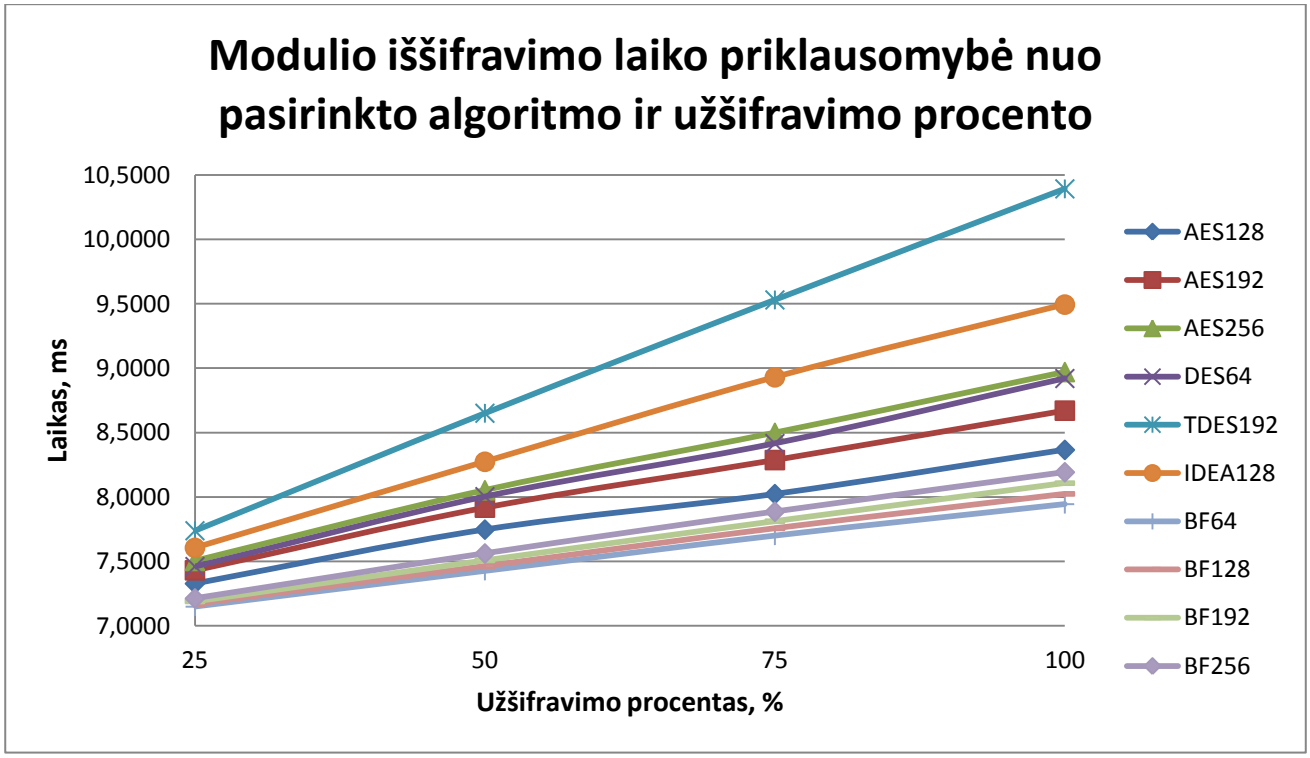
4.1 lentelė. Tyrimo metu naudoti simetriniai šifravimo algoritmai

Nr.	Pavadinimas	Rakto ilgis (bitais)
1	AES (RijnDael)	128, 192, 256
2	DES	64
3	TripleDES	192
4	IDEA	128
5	Blowfish	64, 128, 192, 256

Modulio iššifravimo duomenys, naudojant 4.1 lentelėje išvardintus algoritmus, pateikiami priedo 7.2, 7.3 lentelėse. Šių duomenų suvestinė pateikiama 4.2 lentelėje. Šiuos duomenis atitinkantis grafikas pateikiamas 4.4 pav.

4.2 lentelė. Modulio iššifravimo laikai (ms)

Nr.	Modulio pavadinimas ir rakto ilgis	Modulio užšifravimo procentas, %			
		25	50	75	100
1	AES 128	7,3285	7,7473	8,0231	8,3663
2	AES 192	7,4315	7,9180	8,2875	8,6711
3	AES 256	7,5042	8,0531	8,4990	8,9705
4	DES 64	7,4608	8,0036	8,4161	8,9204
5	TripleDES 192	7,7390	8,6508	9,5307	10,3928
6	IDEA 128	7,6057	8,2741	8,9300	9,4943
7	Blowfish 64	7,1493	7,4265	7,7005	7,9452
8	Blowfish 128	7,1685	7,4638	7,7572	8,0249
9	Blowfish 192	7,1902	7,5085	7,8123	8,1082
10	Blowfish 256	7,2132	7,5623	7,8866	8,1929



4.4 pav. Modulio iššifravimo laiko priklausomybė nuo pasirinktų parametrų

Kaip matome iš 4.4 pav. pateiktos diagramos, blogiausiomis laikinėmis charakteristikomis pasižymėjo *TripleDES* algoritmas su 192 bitų raktu. Šį algoritmą palyginome su likusiais iššifravimo algoritmais ir nustatėme, kiek procentų greičiau likę algoritmai iššifravimo užduotį atlieka už *TripleDES* algoritmą. Skaičiavimų rezultatai pateikiami 4.3 lentelėje.

4.3 lentelė. Šifravimo algoritmų greitaveikos palyginimas su *TripleDES* algoritmu (%)

Nr.	Algoritmo pavadinimas ir rakto ilgis	Modulio užšifravimo lygis, %			
		25	50	75	100
1	AES 128	5,60	11,66	18,79	24,22
2	AES 192	4,14	9,26	15,00	19,86
3	AES 256	3,13	7,42	12,14	15,85
4	DES 64	3,73	8,09	13,24	16,51
5	IDEA 128	1,75	4,55	6,73	9,46
6	Blowfish 64	8,25	16,48	23,77	30,81
7	Blowfish 128	7,96	15,90	22,86	29,51
8	Blowfish 192	7,63	15,21	22,00	28,18
9	Blowfish 256	7,29	14,39	20,85	26,85

Kaip matyti diagramos pateiktos 4.4 pav. ir iš 4.3 lentelės, geriausiomis laikinėmis charakteristikomis pasižymėjo *Blowfish* algoritmas su 64 bitų raktu. Esant 25% modulio užšifravimo lygiui, *Blowfish* algoritmas su 64 bitų raktu modulį iššifravo 8,25% greičiau, nei tai padarė *TripleDES* algoritmas su 192 bitų raktu. Esant tam pačia užšifravimo lygiui, *Blowfish* algoritmas su

192 bitų raktu modulį iššifruoja 7,63%, o AES algoritmas su tokiu pačiu rakto ilgiu 4,14% greičiau, nei tai padaro *TripleDES* algoritmas su 192 bitų raktu.

Reikia pastebėti, kad didėjant iššifruojamų duomenų kiekiui, didėja ir iššifravimo laikas. Šifruojamas modulis užėmė ~6KB, todėl esant 25% užšifravimo lygiui iššifruojamų duomenų kiekis sudarė 1,5KB, esant 50% užšifravimo lygiui – 3KB, esant 75% užšifravimo lygiui – 4,5KB, o esant 100% užšifravimo lygiui visus 6KB.

Didėjant iššifruojamų duomenų kiekiui ryškiau pasimato ir algoritmų greitaveikos skirtumai. *Blowfish* algoritmas su 192 bitų raktu, esant 25% modulio užšifravimo lygiui, modulį iššifruoja 7,63% greičiau nei tai daro *TripleDES* algoritmas su 192 bitų raktu, o esant 100% modulio užšifravimo lygiui – net 28,18% greičiau nei tai padaro *TripleDES* algoritmas.

4.4. Apsaugotos programos vykdymo laikas

Ankstesniuose skyreliuose nagrinėjome kiek laiko trunka kompiuterio komponentų duomenų surinkimas, sistemos parašo suskaičiavimas bei patikrinimas ir kiek laiko trunka apsaugoto modulio iššifravimas lustinėje kortelėje. Šiame skyriuje panagrinėsime kaip kinta apsaugotos programos vykdymo laikas, kintant užšifravimo algoritmams, jų raktų ilgiams bei modulio užšifravimo procentui.

Lutinėje kortelėje buvo vykdomas modulis, kuriame buvo suprogramuota diferencinės lygties *Y'=F(x,y)* sprendimas panaudojant *Runge-Kutta* metodą. Modulio įvesties duomenys – intervalo pradžia – 0; intervalo pabaiga – 1; reikšmė – 1; žingsnis – 0,001. Esant tokiems duomenims įvykdžius modulį buvo gaunama 1000 rezultato reikšmių.

Iš prieš tai atliktų tyrimų (4.3 skyrelis) išsiaiškinome, jog ilgiausiai iššifravimo veiksmą vykdo *TripleDES* algoritmas su 192 bitų raktu, o trumpiausiai – *Blowfish* algoritmas su 64 bitų raktu. Remiantis šiais duomenimis darome prielaidą, kad ilgiausiai apsaugota programa turėtų būti vykdoma tada, kai modulio iššifravimui bus naudojama *TripleDES* algoritmas, o trumpiausiai, kai iššifravimui bus naudojamas *Blowfish* algoritmas su 64 bitų raktu.

Apsaugotos programos vykdymo duomenys pateikiami priedo 7.2, 7.3 lentelėse. Šiuos duomenis atitinkantis grafikas pateikiamas 4.5 pav.

4.5 pav. Apsaugotos programos vykdymo laikai naudojant įvairius šifravimo algoritmus

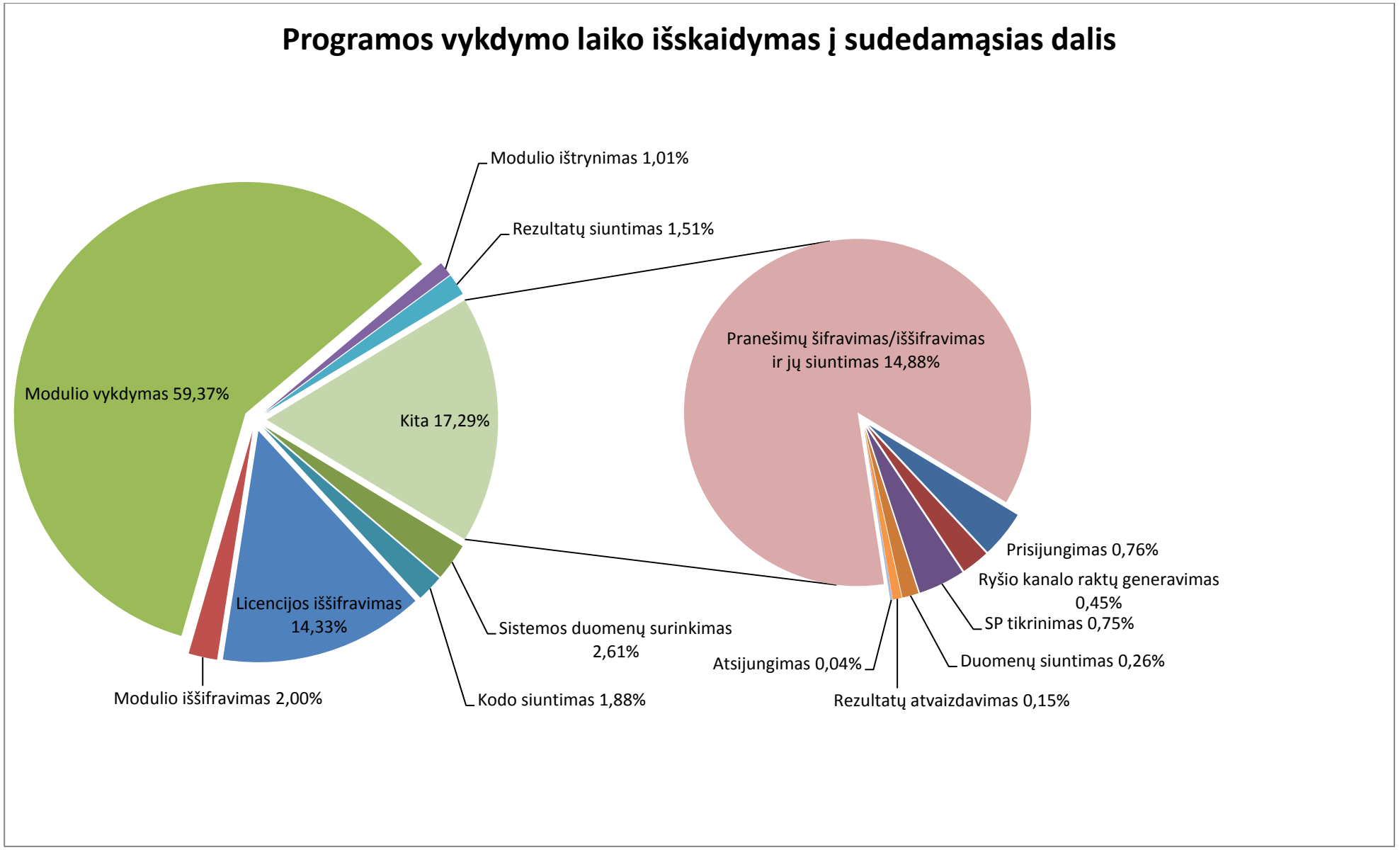
Kaip matome iš 4.5 pav. atvaizduoto grafiko, mūsų iškelta prielaida nepasitvirtino. Naudojant *TripleDES* algoritmą modulio iššifravimui programos vykdymo laikas neviršijo apsaugotos programos vykdymo laikų vidurkio. Iššifravimui naudojant *Blowfish* algoritmą su 64 bitų raktu gavome panašius apsaugotos programos vykdymo laikus, kaip ir naudojant *TripleDES* algoritmą. Ilgiausiai apsaugota programa buvo vykdoma iššifravimui naudojant AES algoritmą su 128 bitų raktu (esant 25%, 50%, 75% ir 100% modulio užšifravimo lygiams), AES algoritmą su 256 bitų raktu (esant 75% modulio užšifravimo lygiui), Blowfish algoritmą su 192 bitų raktu (esant 75% modulio užšifravimo lygiui) bei Blowfish algoritmą su 256 bitų raktu (esant 75% modulio užšifravimo lygiui).

Prieš pradedant nagrinėti kodėl mūsų prielaida nepasitvirtino, palyginkime neapsaugotos programos vykdymą kompiuteryje ir apsaugotos programos vykdymą, kai kritiniam moduliui vykdyti naudojama lustinė kortelė. Iš 4.5 pav. matome, kad neapsaugotos programos vykdymas kompiuteryje užtrunka apie 100 milisekundžių. Apsaugota programa, kuriai pritaikėme mūsų sukurtą apsaugos metodą nuo piratavimo, vykdoma kompiuteryje ir lustinėje kortelėje. Iš apsaugotos programos vykdymo laikų vidurkio (kuris yra ~400 ms) matome, kad neapsaugotos ir apsaugos programos vykdymo laikas skiriasi 4 kartus. Tai labiau teorinis palyginimas, kadangi kaip jau minėjome, šiame darbe nesinaudojome realia lustine kortele, o tik jos programine realizacija. Realios lustinės kortelės ir tyrimo aplinkos procesorių greičiai skiriasi apie 40-50 kartų.

Toliau panagrinėsime kodėl skyrelio pradžioje mūsų iškelta prielaida nepasitvirtino. Atsakymą galima rasti 4.6 pav. pateiktoje diagramoje. Kaip matyti iš šios diagramos, didžiausią dalį (59,37%) apsaugotos programos vykdymo laiko užima modulio vykdymas lustinėje kortelėje, o modulio iššifravimo laikas sudaro tik ~2% visos programos vykdymo laiko. Tam įtakos turi mažas modulio dydis. Naudojant realią lustinę kortelę modulio iššifravimo laikas išaugtų, tačiau išaugtų ir kitų veiksmų atlikimo laikas, todėl darome prielaidą, kad naudojant realią lustinę kortelę modulio iššifravimo procentas bendrame laike nepadidėtų.

Licencijos iššifravimas užima kur kas daugiau laiko (14,33%) nei modulio iššifravimas, nors duomenų iššifruojama labai mažai – vos kelios simbolių eilutės. Taip yra todėl, kad licencijos iššifravimui naudojama asimetrinis algoritmas – RSA.

Taip pat reikia paminėti, kad sistemos parašui reikiamų komponentų surinkimas sudaro 2,61% viso vykdymo laiko, o sistemos parašo suskaičiavimas ir patikrinimas tik 0,75%. Didelę apsaugotos programos vykdymo dalį (14,88%) užima pranešimų šifravimo/iššifravimo ir siuntimo veiksmai. Norint sumažinti apsaugotos programos vykdymo laiką, būtent šioje vietoje pirmiausiai reikėtų atlikti optimizavimo veiksmus. Kitos apsaugotos programos vykdymo sudedamos dalys yra nedidelės, ir užima santykinai nedaug laiko.



4.6 pav. Apsaugotos programos vykdymo laiko išskaidymas

4.5. Išvados

Atlikę programinę sumodeliuoto modelio realizaciją ir ją ištyrę nustatėme, kad

1) kompiuterio komponentų duomenų surinkimas užtrunka 1,7 karto ilgiau nei sistemos parašo skaičiavimas ir tikrinimas kartu sudėjus;
2) geriausiomis modulio iššifravimo laikinėmis charakteristikomis pasižymėjo *Blowfish* algoritmas su 64 bitų raktu, o blogiausiomis – *TripleDES* su 192 bitų raktu;
3) *Blowfish* algoritmas su 192 bitų raktu esant 25% modulio užšifravimo lygiui modulio iššifravimo veiksmą atliko 7,63% greičiau nei *TripleDES* algoritmas, o esant esant 100% modulio užšifravimo lygiui – net 28,18% greičiau;
4) didėjant iššifruojamų duomenų kiekiui stipriau išryškėja ir šifravimo algoritmų greitaveikos skirtumai;
5) apsaugotos programos vykdymo laikų vidurkis yra 4 kartus didesnis nei neapsaugotos programos. Realios lustinės kortelės ir tyrimo aplinkos procesorių greičiai skiriasi 40-50 kartų, todėl darome prielaidą, kad naudojant realią lustinę kortelę apsaugotos programos vykdymo laikas išaugtų dar labiau.
6) prielaida, jog apsaugota programa ilgiausiai turėtų būti vykdoma modulio iššifravimui naudojant *TripleDES* (192 bitų raktas), o trumpiausiai naudojant *Blowfish* (64 bitų raktas) nepasitvirtino, nes modulio iššifravimas užima tik ~2% bendro apsaugotos programos vykdymo laiko;
7) didžiausią bendro programos vykdymo laiko procentą užima: modulio vykdymas lustinėje kortelėje (59,37%), šifruotų pranešimų apsikeitimas tarp apsaugotos programos ir lustinės kortelės (14,88%) ir licencijos iššifravimas privačiuoju lustinės kortelės raktu (14,33%).

5. IŠVADOS

Šiame skyriuje pateikiamos viso darbo išvados, kurios paremtos įvade suformuluotais uždaviniais.

1) Atlikus taikomųjų programų grėsmių analizę nustatėme, kad didžiausią grėsmę programų saugai kelia atvirkštinė inžinerija. Dažniausiai atvirkštinei inžinerijai naudojamos priemonės derintuvai ir kodo ardymo priemonės.
2) Išanalizavus programinius ir aparatūrinius apsaugos metodus nuo piratavimo išsiaiškinome, kad dažniausiai naudojamos programinės priemonės yra kodo šifravimas/glaudinimas bei kodo supainiojimas, o dažniausiai naudojamos aparatūrinės priemonės – apsaugos raktai. Palyginus programines ir aparatūrines apsaugos priemones nustatėme, kad geriau nuo taikomųjų programų piratavimo apsaugo aparatūrinės apsaugos priemonės, tačiau jos yra brangios ir ne tokios lanksčios kaip programinės apsaugos priemonės, todėl naudojamos rečiau.
3) Atsižvelgiant į analizės dalyje padarytas išvadas sukūrėme kompleksinį apsaugos metodą nuo piratavimo, kuris paremtas atskirų programos modulių šifravimu simetriniu algoritmu ir šių modulių vykdymu lustinėje kortelėje. Iššifravimo raktas saugomas licencijoje, kuri užšifruota lustinės kortelės viešuoju raktu ir gali būti iššifruojama tik kortelės viduje. Taikomosios programos papildomai apsaugomos kodo supainiojimo ir sistemos parašo metodais.
4) Atlikus sumodeliuoto metodo programinę realizaciją nustatėme, kad apsaugotos programos vykdymo laikas yra 4 kartus didesnis negu neapsaugotos programos. Kadangi realios lustinės kortelės skaičiuojamieji resursai nuo tyrimo aplinkos skiriasi apie 40-50 kartų, todėl darome prielaidą, kad naudojant realią lustinę kortelę apsaugotos programos vykdymo laikas dar labiau išaugtų. Atliekant modulio iššifravimo laiko priklausomybės nuo pasirinkto šifravimo metodo, jo rakto ilgio ir modulio užšifravimo procento tyrimą nustatėme, kad blogiausiomis laikinėmis charakteristikomis pasižymėjo *TripleDES* algoritmas su 192 bitų raktu, geriausiomis – *Blowfish* su 64 bitų raktu. Tyrimo metu padaryta prielaida, jog apsaugota programa ilgiausiai bus vykdoma naudojant *TripleDES*, o trumpiausiai *Blowfish* (64 bitų raktas) algoritmus nepasitvirtino, nes modulio iššifravimas sudaro tik 2% viso apsaugotos programos vykdymo laiko. Dėl šios priežasties modulio šifravimui galima pasirinkti bet kokį saugų šifravimo algoritmą su ilgiausiu raktu. Didžiausią bendro programos vykdymo laiko procentą užima modulio vykdymas lustinėje kortelėje (59,37%).

6. LITERATŪRA

[1] „Eighth annual BSA global software 2010 piracy study“, Business Software Alliance, 2011.

[2] „Shadow market 2011 BSA global software piracy study. Ninth edition.“, Business Software Alliance, 2012.

[3] „Piracy impact study in brief: The econimic benefits of reducing sowtware piracy“, Business Software Alliance, 2010.

[4] „NCTE Advice Sheet - Software Licencing“, National Centre for Technology in Education, 2011.

[5] „Software License Types“, ServiceDesk, [Interaktyvus]. [Žiūrėta 2013-04-15]. Prieiga per internetą: http://www.manageengine.com/products/service-desk/help/adminguide/configurations/software/software-license-type.html..

[6] „Identify and avoid piracy : Types of piracy“, Adobe, [Interaktyvus]. [Žiūrėta 2013-03-19], Prieiga per internetą: http://www.adobe.com/antipiracy/identify-and-avoid-piracy.html..

[7] S. Parker, „Types of Software Piracy“, [Interaktyvus]. [Žiūrėta 2013-05-11]. Prieiga per internetą: http://www.ehow.com/list_5911080_types-software-piracy.html..

[8] Katzenbeisser, S., Schrittwieser S., „Code Obfuscation against Static and Dynamic“, pp. 1-16.

[9] Birrer B., Raines R., Baldwin R., Mullins B., Bennington R. R., „Program Fragmentation as a Metamorphic Protection// Third International Symposium of Information Assurance and Security“, 2007, p. 369-374.

[10] Borello J. M., Me L.Code, „Obfuscation techniques for metamorphic viruses // Journal in Computer Virology“, 2008.

[11] P. Červen, „Crackproof Your Software – The Best Ways to Protect Your Software Against Crackers“, p. 170, 2001.

[12] N. Jusas, „Programų apsaugos metodas, naudojant sistemos parašą“, Kaunas, 2012.

[13] Serge Chaumette, Olivier Ly, Renaud Tabary, „Automated secure software protection through program externalization on memory-limited secure devices“.

[14] F. Veseli, „HIKOS - Highly Secure, Intelligent Software. Master work“, Gjøvik, 2001.

[15] Antonio Mana, Javier Lopez, Juan J. Ortega, Ernesto Pimentel, Jose M. Troya, „Aframework for secure execution of software“, Malaga, 2004.

[16] P. Weber, „The anti-piracy Copyright Alert System: Is the Napster era finally dead?“, *The Week,* 2013.

[17] J. Cappaert, „Code Obfuscation Techniques“, Arenberg, 2012.

[18] M. Popa, „Techniques of Program Code Obfuscation for Secure Software“, *Journal of Mobile, Embedded and Distributed Systems,* p. 15, 2011.

[19] Algimantas Venčkauskas, Egidijus Kazanavičius, "Informacinių technologijų saugos metodai. Mokomiji knyga", Kaunas, KTU, 2010.

[20] A. Liutkevicius, A. Vrubliauskas, E. Kazanavicius, „Assessment of Dongle-based Software Copy Protection Combined with Additional Protection Methods“, *Sistemų inžinerija, kompiuterinės technologijos,* p. 6, 2011.

[21] M. Lenza, „E. parašo taikymas autentifikavimui ir šifravimui video paskaitų sistemoje. Magistrinis darbas“, Kaunas, 2010.

[22] J. Matačiūnas, „Vartotojo prieigos duomenų saugojimo lustinėse kortelėse metodo sukūrimas ir tyrimas. Magistrinis darbas“, Kaunas, 2011.

[23] S. R. Adavalli, „Smart Card Solution: Highly secured Java Card Technology“, Auckland.

[24] J. Matačiūnas, „Vartotojo prieigos duomenų saugojimo lustinėse kortelėse metodo sukūrimas ir

tyrimas. magistro darbas,“ Kaunas, 2011.

[25] „Chip Card & Security ICs SLE 66CLX360PE(M) Family“, Infineon.

[26] „Standart ISO/IEC 7816“, [Interaktyvus]. [Žiūrėta 2013-04-15]. Prieiga per internetą: http://www.smartcardbasics.com/smart-card-standards.html .

[27] „Standart SO/IEC 14443“, [Interaktyvus]. [Žiūrėta 2013-05-15]. Prieiga per internetą:: http://wg8.de/sd1.html. [Kreiptasi 1 1 2013].

[28] M. Witteman, „Advances in smart card security“, 2002.

[29] M. G., „Guarding against identity theft,“ *Hakin9,* 2011.

7. PRIEDAI

Šiame skyriuje pateikiami sumodeliuoto apsaugos metodo tyrimo rezultatai.

7.1. priedas. Tyrimo duomenys

7.1 lentelė. Sistemos parašo apsaugos tyrimo duomenys

Matavimo nr.	Sistemos komponentų surinnkimo laikas (ms)					Sistemos parašo skaičiavimo ir tikrinimo laikas (ms)				
	Komponentų skaičius									
	2	4	6	8	10	2	4	6	8	10
1	0,7336	6,4295	6,9468	7,7551	9,0197	0,9649	3,0637	4,1074	4,1445	5,2016
2	0,7391	6,4671	7,2329	8,1972	9,0731	1,0237	3,4938	4,1129	4,5784	5,3106
3	0,7397	6,5047	7,2671	8,2095	9,2730	1,5136	3,5415	4,1307	4,5997	5,3798
4	0,7445	6,6444	7,3150	8,2464	9,7383	1,6040	3,8960	4,2020	4,6153	5,5534
5	0,7452	6,7997	7,3212	8,2683	10,3248	1,6067	3,9411	4,3811	4,7768	5,5782
6	0,7692	6,8079	7,3294	8,3806	11,1556	1,8127	4,0410	4,4872	5,1306	5,5782
7	0,8328	6,8647	7,6257	8,4148	11,1666	1,8525	4,1047	4,5832	5,1970	5,6994
8	0,8506	6,9722	7,6593	8,4894	11,2035	1,9093	4,1129	4,5998	5,2237	5,7812
9	0,8773	7,1330	8,7932	8,5079	11,2884	1,9709	4,2751	4,6386	5,2313	5,8584
10	1,0771	7,3123	8,8083	8,6488	11,3260	2,9121	4,3845	4,6563	5,2839	5,9356
11	1,0840	7,5990	8,8877	8,9315	11,3828	3,5072	4,6563	4,7544	5,2914	6,0511
12	1,0867	7,7202	9,0129	10,4576	11,4040	3,5255	4,7281	4,9437	5,3134	6,7288
13	1,1230	8,0507	9,0307	10,6335	11,4102	3,5982	4,9457	5,1400	5,3715	6,7411
14	1,1319	8,1527	9,0444	10,9982	11,4725	3,6263	5,0506	5,2982	5,3750	7,1237
15	1,1531	8,1794	9,1422	11,0215	11,5040	3,6651	5,1212	5,4371	5,4085	7,2571
16	1,1729	8,2252	9,1689	11,0646	11,6675	4,0026	5,1284	5,4790	5,4667	7,3707
17	1,1825	8,3703	9,1778	11,0988	12,4230	4,2267	5,1592	5,6069	5,8361	7,5866
18	1,1955	8,5277	9,2209	11,2528	12,4655	4,2534	5,3344	5,8097	6,0196	7,9000
19	1,3132	8,5852	9,2387	11,3616	12,4928	4,4058	5,5160	5,8847	6,7035	7,9038
20	1,3146	8,5982	9,2798	11,4143	12,5059	4,7155	5,5462	6,0441	6,7788	7,9439
21	1,3297	8,7966	9,5371	11,5375	12,7816	4,7637	5,7334	6,2342	7,0135	7,9864
22	1,4268	8,9349	9,7835	11,6463	12,9917	4,7680	5,7791	6,5328	7,9267	7,9914
23	1,4467	9,7766	10,2646	12,1212	13,6816	4,8818	5,8443	6,6111	7,9563	8,1506
24	1,8121	10,6088	20,3403	15,7401	13,9916	4,9306	5,9148	6,8336	8,1101	8,5252
25	4,2053	11,1460	21,1705	15,9297	22,3585	4,9943	5,9866	7,8697	8,9655	8,9033
Vidurkis:	1,2035	7,9683	9,5440	10,3331	11,9241	3,2414	4,7719	5,2951	5,8527	6,8016

7.2 lentelė. Kompiuteryje atliktų veiksmų laikų vidurkiai

Nr.	Neapsaugotos programos vykdymas	Algoritmo pavadinimas ir rakto ilgis	Prisijungimas	Ryšio kanalo šifravimo rakto generavimas	10 komponentų duomenų identifikavimas	Duomenų siuntimas	Rezultatų atvaizdavimas	Atsijungimas	Apsaugotos programos vykdymas	Moduliu siuntimas
	Laikas, ms		Laikas, ms							
1	127,5311	25AES128	2,8755	3,8736	9,3937	0,9660	0,2818	0,1571	487,4809	7,9185
2	113,2977	50AES128	3,3287	4,2841	10,2689	1,0517	0,4664	0,1490	461,9394	8,8262
3	84,6853	75AES128	3,1661	5,7098	10,9025	1,1442	0,5204	0,1631	502,6220	8,5892
4	98,5039	100AES128	3,0575	4,8450	9,6014	1,2134	0,4327	0,1454	476,1632	7,5023
5	92,8860	25AES192	2,8766	1,8753	10,4883	0,9912	0,5112	0,1442	389,8297	6,4595
6	98,7072	50AES192	2,8512	1,8391	10,3084	0,9272	0,5155	0,1433	382,7158	5,4958
7	86,9075	75AES192	2,8141	1,2879	10,7688	0,9900	0,6022	0,1454	384,4022	5,1179
8	96,1799	100AES192	2,8394	1,3213	10,2782	0,9816	0,2603	0,1438	385,5041	5,5951
9	99,3895	25AES256	3,1103	1,2909	10,8848	1,0567	0,7492	0,1504	381,3985	6,5034
10	90,4285	50AES256	3,0459	1,5980	10,2257	1,0074	0,4896	0,1466	396,1579	8,7851
11	88,5835	75AES256	3,1255	1,1574	11,2323	1,3532	0,5821	0,1873	437,3381	7,2118
12	88,3508	100AES256	3,2840	1,2923	9,6917	0,9943	0,6464	0,1509	408,8853	5,0452
13	98,9481	25DES64	2,8157	1,6198	10,9525	0,9554	0,8375	0,1464	389,0796	6,4910
14	85,7790	50DES64	3,8107	1,3566	10,7844	1,0107	0,6459	0,1780	385,9588	6,5495
15	87,6904	75DES64	2,8739	1,4192	10,3112	1,0362	0,6567	0,1472	382,3253	6,1924
16	95,6426	100DES64	3,0267	1,1818	10,4019	1,1247	0,6615	0,1712	370,7489	6,0720
17	85,7092	25TDES192	3,8115	1,6640	10,8512	1,1374	0,7330	0,1451	390,4934	6,7594
18	112,9624	50TDES192	2,8032	1,8491	9,7092	0,9827	0,7928	0,1456	388,7729	6,2475
19	86,1909	75TDES192	2,8752	1,8608	9,5882	0,9812	0,6973	0,1478	384,5864	6,2586
20	94,4416	100TDES192	3,8931	1,3088	10,2398	0,9537	0,8465	0,1540	404,5813	5,3488
21		25IDEA128	2,9147	1,3785	9,7333	1,1573	0,2695	0,1466	381,2263	6,9144
22		50IDEA128	2,8359	1,2180	10,5641	0,9645	0,6684	0,1461	395,5192	6,5022
23		75IDEA128	2,8531	1,3728	10,5955	0,9949	0,5378	0,1500	383,4180	5,5146
24		100IDEA128	3,3164	2,0679	11,2884	1,1562	0,7276	0,1427	388,4643	5,5223

Nr.	Neapsaugotos programos vykdymas	Algoritmo pavadinimas ir rakto ilgis	Prisijungimas	Ryšio kanalo šifravimo rakto generavimas	10 komponentų duomenų identifikavimas	Duomenų siuntimas	Rezultatų atvaizdavimas	Atsijungimas	Apsaugotos programos vykdymas	Moduliu siuntimas
	Laikas, ms		Laikas, ms							
25		25BF64	2,7790	1,1985	9,7577	0,9295	0,7289	0,1444	392,0522	6,3509
26		50BF64	3,2850	1,1562	11,1267	0,9629	0,5400	0,1432	382,5791	6,4800
27		75BF64	2,8416	1,8410	10,4170	1,2354	0,7573	0,1484	404,9505	5,8093
28		100BF64	3,0571	1,2435	10,8986	0,9350	0,6630	0,1605	379,5649	5,1449
29		25BF128	2,8600	1,3799	11,2481	1,2575	0,7224	0,1470	399,9534	6,5290
30		50BF128	2,8336	1,5190	10,7728	0,9701	0,5423	0,1444	393,7814	6,5202
31		75BF128	3,5800	1,3590	10,9981	0,9935	0,5980	0,1457	384,9369	5,7412
32		100BF128	2,8462	1,5726	10,5278	0,9816	0,6237	0,1510	402,6154	5,1756
33		25BF192	2,8550	1,6013	10,7628	1,1983	0,6690	0,1456	399,3255	6,8728
34		50BF192	2,8280	1,9909	10,5429	0,9866	0,6113	0,1525	393,1709	6,4174
35		75BF192	2,8816	1,1500	11,3675	1,2196	0,7922	0,1455	421,5964	6,6911
36		100BF192	2,8243	1,0458	10,3002	1,0327	0,8316	0,1448	383,8969	5,5078
37		25BF256	3,1753	1,6524	10,8829	1,2713	0,3802	0,1449	398,6526	9,2192
38		50BF256	3,4170	2,0102	10,0189	1,0236	0,3806	0,1599	410,4951	6,5088
39		75BF256	2,9295	1,5919	9,9761	1,4470	0,2760	0,1430	433,7598	6,0249
40		100BF256	3,2390	1,4171	11,1692	0,9869	0,2735	0,1456	388,2194	4,8376
V.:	95,6408		3,0609	1,8100	10,4958	1,0641	0,5881	0,1503	402,7290	6,4313

7.3 lentelė. Lustinėje kortelėje atliktų veiksmų laikų vidurkiai

Nr.	Algoritmo pavadinimas ir rakto ilgis	SP skaičiavimas ir tikrinimas	Licencijos iššifravimas	Modulio iššifravimas	Modulio vykdymas	Modulio ištrynimas	Rezuktatų siuntimas
		Laikas, ms					
1	25AES128	13,6651	56,1718	7,3285	260,0629	8,0638	10,2099
2	50AES128	10,9561	56,9622	7,7473	236,6272	9,3906	7,0103
3	75AES128	13,8299	61,7905	8,0231	248,5243	9,6770	10,8161
4	100AES128	11,5657	55,8252	8,3663	243,5368	9,8956	9,1879
5	25AES192	1,9253	57,2267	7,4315	237,7507	2,6788	7,1136
6	50AES192	1,6831	56,3804	7,9180	232,9438	3,0820	5,7110
7	75AES192	2,1679	57,5992	8,2875	230,3990	3,1671	6,1338
8	100AES192	1,8419	58,5866	8,6711	231,8193	2,7657	5,9857
9	25AES256	1,8488	57,1148	7,5042	227,3185	3,0288	7,3240
10	50AES256	1,7955	56,6775	8,0531	243,1583	3,2360	4,1368
11	75AES256	1,1952	62,8056	8,4990	274,3289	3,2267	3,1475
12	100AES256	2,0379	55,3164	8,9705	260,1757	3,3108	6,0720
13	25DES64	2,4944	56,2289	7,4608	235,0437	3,2542	5,0023
14	50DES64	1,8955	57,5284	8,0036	229,8303	3,4436	5,4719
15	75DES64	1,9534	56,6098	8,4161	231,1476	3,9267	8,2377
16	100DES64	1,0346	56,7891	8,9204	222,1850	2,9397	2,8879
17	25TDES192	1,9447	56,4077	7,7390	237,9510	3,4044	9,2806
18	50TDES192	1,6996	58,6889	8,6508	233,2991	3,8821	5,5054
19	75TDES192	2,0936	57,4396	9,5307	226,8735	3,7378	4,9863
20	100TDES192	2,0464	58,6996	10,3928	243,0699	5,0381	8,4343
21	25IDEA128	1,9627	56,5266	7,6057	227,6735	3,2955	3,8079
22	50IDEA128	2,1484	57,3421	8,2741	241,3334	3,3897	6,5439
23	75IDEA128	2,0364	57,6011	8,9300	223,5366	3,6539	5,4224
24	100IDEA128	2,2318	56,8785	9,4943	231,3621	3,3256	3,9917
25	25BF64	2,0521	57,7914	7,1493	239,1603	3,4656	6,0518

Nr.	Algoritmo pavadinimas ir rakto ilgis	SP skaičiavimas ir tikrinimas	Licencijos iššifravimas	Modulio iššifravimas	Modulio vykdymas	Modulio ištrynimas	Rezuktatų siuntimas
		Laikas, ms					
26	50BF64	1,9027	57,6756	7,4265	226,9677	3,3146	5,4980
27	75BF64	2,0650	58,4275	7,7005	246,2335	3,3736	4,3305
28	100BF64	1,7341	56,8704	7,9452	228,9567	3,2601	4,0495
29	25BF128	1,8560	58,7850	7,1685	238,2142	4,2561	8,7660
30	50BF128	2,1994	57,0320	7,4638	233,2069	3,8077	5,5428
31	75BF128	2,1983	57,8914	7,7572	227,7944	3,6375	3,2971
32	100BF128	2,1239	58,1213	8,0249	247,3025	3,4085	4,8832
33	25BF192	2,0678	58,6534	7,1902	238,8663	3,5460	5,7820
34	50BF192	2,0097	58,0684	7,5085	232,2922	3,9777	5,4140
35	75BF192	2,0188	60,3403	7,8123	260,0133	4,0384	6,0605
36	100BF192	2,7585	56,8859	8,1082	227,9546	3,6654	5,3678
37	25BF256	1,9441	57,7644	7,2132	239,7573	3,7051	6,6747
38	50BF256	2,1955	58,2973	7,5623	252,7805	3,5052	3,7061
39	75BF256	2,1799	58,5449	7,8866	280,0094	3,0083	9,6973
40	100BF256	1,6985	57,7146	8,1929	234,9220	3,5720	6,4861
Vidurkis:		3,0265	57,7015	8,0582	239,1096	4,0589	6,1007

7.2. priedas. Kompaktinis diskas