KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
VERSLO INFORMATIKOS KATEDRA

Tomas Valinčius

Programinės įrangos ir duomenų saugumas: grėsmės ir jų valdymas, šifravimo algoritmai

Magistro darbas

Darbo vadovas
doc. dr. V. Pilkauskas
2008-05-10

Kaunas, 2008

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
VERSLO INFORMATIKOS KATEDRA

Tomas Valinčius

Programinės įrangos ir duomenų saugumas: grėsmės ir jų valdymas, šifravimo algoritmai

Magistro darbas

Recenzentas
doc. dr. T. Blažauskas
2008-05

Vadovas
doc. dr. V. Pilkauskas
2008-05-10

Atliko
IFM-2/2 gr. stud.
Tomas Valinčius
2008-05-10

Kaunas, 2008

TURINYS

1 ĮVADAS

Ar saugumas yra problema? Šiandien programų inžinerijoje gerai žinomos priemonės, metodai kokybiškai, saugiai programinei įrangai kurti. Vis dėlto spaudimas kuo greičiau į rinką paleisti naują programinę įrangą ar jos versiją daugeliu atvejų yra svarbesnis nei reikalavimai saugumui ar potencialių grėsmių studija.

Šio darbo tikslas – išnagrinėti būdus darbo kompiuteriu keliamoms grėsmėms valdyti:

- Aptarti, pažinti grėsmes kompiuterinių sistemų saugumui;
- Suprasti, kas sukelia šias grėsmes, nagrinėjant programinės įrangos kūrimo procesą;
- Nustatyti būdus, kurie gali sumažinti ar eliminuoti grėsmes.

Darbe pirmiausiai aptariamos techninės priežastys, lemiančios saugumo spragų programinėje įrangoje pasirodymą; piktavališkų programų tipai bei jų daroma žala. Toliau aptarti šiuo metu naudojami metodai programinės įrangos saugumui ir kokybei užtikrinti, suformuluoti pagrindiniai saugios programinės įrangos kūrimo principai.

Kadangi duomenys yra viena iš trijų kompiuterinės sistemos sudedamųjų dalių (programinė įranga, techninė įranga, duomenys), svarbią dalį darbe užima informacijos slaptumo, konfidencialumo užtikrinimo problema. Aptarti pagrindiniai dalykai, užtikrinantys asmens privatumą darbo kompiuteriu metu; pasiūlyti būdai, kaip tinkamai saugoti privačius/slaptus duomenis; paminėtos teisinės priemonės šioje srityje. Taip pat aptarti saugaus darbo internete principai.

Saugumui užtikrinti labai svarbūs organizaciniai veiksniai. Šiame darbe aptartos organizacinės priemonės duomenų bei programinės įrangos saugumui užtikrinti, saugumo politika, grėsmių analizės priemonės.

Darbe taip pat išnagrinėti šifravimo ir dešifravimo algoritmai – pagrindinės techninės priemonės daugeliui su saugumu susijusių problemų spręsti. Aptarti ne tik šiuo metu naudojami, bet ir patys pirmieji šifravimo algoritmai, algoritmų tipai, paskirtis, vertinimo kriterijai.

Su saugumo problemomis susidurta projektuojant ir realizuojant nuotolinio mokymosi ir žinių testavimo sistemą „Cool Test Tool“.

2 PROGRAMINĖS ĮRANGOS IR DUOMENŲ SAUGUMAS

2.1 Nuotolinio mokymosi ir žinių testavimo sistema „Cool Test Tool“ (CTT)

Nuotolinio mokymosi ir žinių testavimo sistema „Cool Test Tool“ (toliau CTT) buvo sukurta magistro studijų metu, 2008-aisiais metais.

Sistema sukurta tam, kad automatizuoti ir palengvinti žmogaus žinių testavimo darbą. Programa labiausiai tinkama mokymo įstaigoms, pvz., mokykloms, universitetams, taip pat didelėms įmonėms, kurioms svarbu nuolat kelti ir tikrinti darbuotojų kvalifikaciją. Sistema puikiai tinka darbuotojų paieška bei atranka užsiimančioms įstaigoms.

Pagrindinės CTT sistemos funkcijos:

- Mokymosi medžiagos saugojimas ir pateikimas,
- Žinių patikrinimo testų kūrimas ir administravimas,
- Sukurtų testų vykdymas ir žinių vertinimas,
- Mokymosi rezultatų vertinimas.

Sistemą sudaro penkios dalys: testų vykdytojo programa, testų administratoriaus programa, dvi serveryje veikiančios ir vartotojų sąsajos programas aptarnaujančios programos (Windows servisai), serverio programas valdanti administratoriaus programa.

Kliento (testų vykdytojo) programa suteikia galimybę naudotis daugialype mokymosi medžiaga (garsinė bei vaizdinė medžiaga, paveikslėliai ir kt.), leidžia vartotojams bendrauti tarpusavyje, dalintis informacija, atlikti žinių patikrinimo testus, matyti mokymosi rezultatus bei testų laikymo tvarkaraščius. Administratoriaus programa leidžia kurti daugialypę mokymosi medžiagą bei žinių patikrinimo testus, valdyti kliento programos vartotojus, matyti bei koreguoti jų mokymosi rezultatus, skirti testus, sudaryti jų laikymo tvarkaraščius.

Programinė įranga sukurta kliento ir serverio architektūros principu.

2.2 CTT ir su saugumu susijusios problemos

CTT sistema realizuota kliento ir serverio architektūros principu. Kadangi tai yra paskirstyta sistema, pagrindinė realizavimo problema yra saugaus ryšio tarp programų užtikrinimas. Pradinėje CTT sistemos versijoje programos tarpusavyje bendrauja nuotolinių metodų kvietimo pagalba (angl. Remote Procedure Call, sutr. RPC). Interneto ryšiu keliaujantys duomenys yra atviri, t.y. nėra užšifruoti ar kaip nors kitaip paslėpti. Taigi pagrindinės grėsmės, kylančios dėl nesaugaus ryšio, yra šios:

- Duomenys gali būti perimti. Interneto ryšiu kliento programos serveriui siunčia vartotojų prisijungimo duomenis, kurie gali būti perimti. Perėmus prisijungimo

duomenis, prie sistemos gali prisijungti pašalinis asmuo ar kita sistema ir atlikti nesankcionuotus veiksmus.

- Duomenys gali būti nukreipti, dėl ko serveryje veikiančios programos gali tapti neprieinamomis.
- Duomenys gali būti modifikuoti / suklastoti. Testų vykdytojo programa serveriui siunčia laikomų žinių patikrinimo testų rezultatus, kurie gali būti perimti ir pakeisti.

CTT sistemos duomenų bazėje yra saugomi vartotojų (dėstytojų, studentų / mokinių) duomenys. Nors tokie duomenys kaip vardas, pavardė nėra laikomi „jautriais“ duomenimis, tačiau, kartu saugant telefono numerį ar elektroninio pašto adresą, galima nesunkiai identifikuoti asmenis. Pavogus ar kitaip perėmus šiuos duomenis, jie gali būti susieti su kitų sistemų duomenų bazėmis, jais gali būti pasinaudota piktavaliais tikslais. Be asmenis apibūdinančių duomenų sistemoje saugomi mokinių / studentų mokymosi rezultatai. Šie duomenys labai dažnai laikomi privačiais duomenimis ir daugelis asmenų nenori jų atskleisti viešai. CTT sistemos duomenų bazėje duomenys saugomi atviru formatu, specialios saugumo priemonės nėra numatytos. Taigi sistemoje egzistuoja privačių asmens duomenų saugojimo ir valdymo problema.

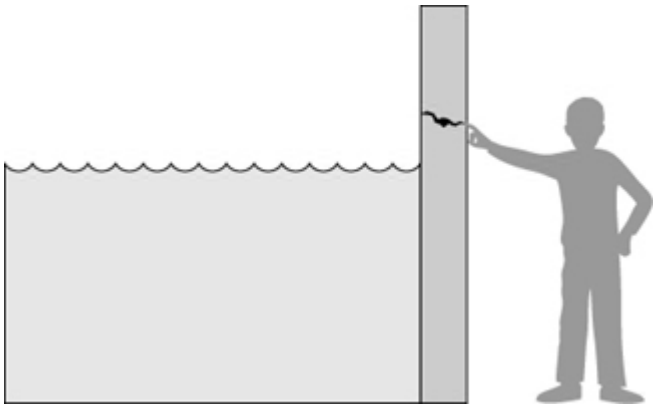
CTT sistemos realizavimo laikotarpiu buvo susidurta ir su organizacinėmis problemomis. Nors sistemą realizavimo tik du asmenys, buvo susidurta su darbų organizavimo, derinimo problemomis. Programinės įrangos kūrimo principai, stilius, kokybės valdymo priemonės nebuvo bendrai sutarti. Taigi CTT sistemą sudarančios atskiros programos neturi vieningų priemonių priežiūrai, palaikymui, o dėl to sumažėjo visos CTT sistemos kaip vieno produkto kokybė. Organizacinės priemonės kompiuterinių sistemų saugumui valdyti yra šio magistro darbo tyrimo objektas.

Realizavus sistemą ir pristačius ją potencialiems pirkėjams, taip pat buvo susidurta su pasitikėjimo sistema problema. Kadangi sistema dar nėra naudojama praktikoje, potencialūs jos vartotojai negali įvertinti realiai teikiamos sistemos naudos. Taip pat yra bijoma, jog sistemoje gali būti realizuotos slaptos funkcijos / metodai (angl. trapdoor, backdoor), kuriais naudojantis testus atliekantys asmenys galėtų sukčiauti. Piktavalio, slapto kodo realizavimo problema taip pat nagrinėjama šio darbo tyrimo dalyje.

2.3 Su saugumu susiję terminai

Kompiuterinė sistema sudaryta iš trijų pagrindinių komponentų: techninės įrangos, programinės įrangos ir duomenų. Pažeidžiamumu (vulnerability) darbe vadinama silpna vieta kompiuterinėje sistemoje (paprastai architektūroje ar realizacijoje), kuria gali būti pasinaudota, kad sukelti žalą. Grėsme (threat) kompiuterinei sistemai gali būti tiek žmogus, tiek kita kompiuterinė sistema. Sistemos ataka (attack) vadiname procesą, kada žmogus ar kita sistema pasinaudoja silpna vieta. Kontrolės priemonėmis/įrankiais vadinami objektai, saugantys nuo atakos. Šiais objektais gali būti tam tikros veiklos, techninės priemonės, procedūros, kurios pašalina silpnas sistemos vietas arba jas sustiprina. Ryšys tarp grėsmių, priemonių ir pažeidžiamų vietų gali būti apibūdintas taip: grėsmė yra eliminuojama/blokuojama kontroliuojant silpnąsias sistemos vietas.

Skirtumas tarp grėsmės ir pažeidžiamumo pavaizduotas paveikslėlyje 1. Paveikslėlio kairėje pavaizduotas vanduo yra grėsmė žmogui dešinėje (vanduo gali sugriauti sieną arba pakilti iki jos aukščio ir užtvindyti žmogų). Šiuo metu žmogus yra saugus, tačiau nuo vandens saugančioje sienoje matomas plyšys – pažeidžiamumas, pro kurį gali pratekėti vanduo šiam pakilus. Žmogus ant plyšio uždėjęs pirštą, t.y. kontroliuoja galimybę vandeniui pratekėti.



Paveikslėlis 1. Grėsmės, pažeidžiamumai ir kontrolės priemonės.

Grėsmės kompiuterinėms sistemoms gali būti suskirstytos į keturis tipus:

- Perėmimas (interception). Neleistinas prisijungimas prie sistemos. Įsibrovėliu gali būti žmogus, programa ar kita sistema.
- Pertraukimas (interruption). Negalėjimas pasinaudoti saugomu objektu dėl jo praradimo, neprieinamumo ar kt. Pvz., failas gali būti ištrintas, techninė įranga sugadinta.
- Pakeitimas (modification). Nesankcionuotas duomenų ar kito objekto pakeitimas.

- Klastojimas (fabrication). Neautorizuoti asmenys gali sufabrikuoti, suklastoti kompiuterinės sistemos naudojamus duomenis. Pvz., įsibrovėlis gali įterpti melagingus įrašus duomenų bazėje, atlikti negalimus, draudžiamus sandorius ar operacijas.

Šiame darbe bus dažnai naudojamos sąvokos „saugus“, „saugumas“. Saugia kompiuterine sistema vadinama tokia sistema, kuri yra:

- Prieinama (availability). Prieinama reiškia, kad asmuo, turintis teisę naudotis sistema, turi galėti ja naudotis jam suteiktu/paskirtu laiku. Ši teisė neturi būti apribota ar atimta trečiųjų šalių.
- Konfidenciali (confidentiality). Naudotis sistema turi galėti tik tokią teisę turintys asmenys (naudotis – skaityti, matyti, spausdinti informaciją ir pan.).
- Integrali / vientisa (integrity). Integralumas reiškia, kad objektai gali būti modifikuoti tik tokią teisę turinčių asmenų ir tik iš anksto numatytais/apibrėžtais būdais (modifikacija apima rašymą, keitimą, trynimą). Sąvoka „integralus“ gali turėti skirtingą paaiškinimą skirtingose sistemose. Integrali sistema gali reikšti apibrėžta (precise), tiksli (precise), nuosekli (consistent), naudojama (usable) sistema ir pan.

Sukurti saugią kompiuterinę sistemą/aplinką – tai pasiekti šių trijų išvardintų savybių.

2.4 Programinės įrangos saugumas

Kompiuterinių sistemų saugumas labai priklauso nuo programinės įrangos (toliau PĮ) saugumo. Šiame skyriuje aptariamos techninės priežastys, įtakojančios PĮ saugumą (tipinės programavimo klaidos, piktavališkos programos ir jų tipai), ir būdai, kaip valdyti programų klaidas dar sistemų kūrimo metu.

Klaidos programų kode vadinamos defektais (fault). Defektai yra PĮ gedimų (failure) priežastys. Gedimais paprastai vadiname tokias PĮ klaidas, kurios matomos vartotojui. Pvz., vartotojas, norintis surikiuoti įvestus skaičius didėjimo tvarka, mato, kad skaičių masyvas surikiuotas neteisingai, t.y. mato gedimą. Šio gedimo priežastimi gali būti įvairūs defektai, pvz., skaičių palyginime nedalyvauja paskutinis masyvo elementas, neteisingai nurodyta palyginimo operacija ir pan.

2.4.1 PĮ klaidos ir jų tipai

Landwehr et al. [LAN94] programų defektus siūlo skirstyti į dvi grupes: tyčinius ir netyčinius. Tyčiniai defektai gali būti padaryti siekiant iš anksto apgalvotų piktavališkų arba nepiktavališkų tikslų. Netyčiniai defektai gali būti skirstomi į šias kategorijas:

- Vartotojų prieigos teisių nesužiūrėjimas,
- Nekontroliuojamas ar nepilnai kontroliuojamas duomenų valdymas,
- Neteisingi programos vykdymo keliai,
- Netinkama subjektų autorizacija,
- Ribinių sąlygų pažeidimai,
- Kitos loginės klaidos.

Kiti autoriai, pvz., Tsipenyuk et al. [TSI05], OWASP project [OWA06] siūlo panašius klaidų grupavimo būdus.

Programuotojai, kaip ir kiti žmonės, savo darbo metu daro klaidų. Dauguma jų nėra piktavališkos. Vis dėlto tam tikros klaidos programinėje įrangoje dažnai kartojasi. Programuotojai turėtų žinoti tipinių klaidų atvejus ir būdus, kaip jų išvengti. Šiame skyrelyje aptariami trys tipinių klaidų atvejai, kodėl jie svarbūs saugumui ir kaip jų išvengti.

Buferio perpildymas (buffer overflow). Buferis – tai vieta kompiuterio atmintyje, kurioje laikomi programos duomenys. Kadangi atminties kiekis yra ribotas, buferio dydis taip pat yra ribotas ir programuotojai turėtų nustatyti maksimalų galimą buferio dydį. Sakykime, kad turime tokį kodą, kuris aprašo dešimties elementų masyvą:

```
char sample[10];
```

Kompiliatorius masyvui išskiria 10 baitų atmintyje, t.y. po vieną baitą kiekvienam elementui (nuo sample[0] iki sample[9]). Toliau užrašytame kode kiekvienam masyvo elementui priskiriama reikšmė A ir bandoma priskirti reikšmę B vienuoliktam elementui, kuris yra už masyvo ribų:

```
for (int i = 0; i <= 9; i++)
    sample[i] = 'A';
sample[10] = 'B';
```

Paprastai kompiliatorius tokių klaidų negali nustatyti programos kompiliavimo metu. Programos vykdymo metu ši atmintimi dalinasi su operacine sistema, kitomis programomis ir jų duomenimis. Reikšmės B priskyrimas gali paveikti:

- Kitus programos duomenis. Simbolis B gali perrašyti egzistuojančio kintamojo reikšmę (arba užimti dar nenaudojamos atminties vietą) ir taip paveikti programos darbą.
- Programos kodą. Simbolis B gali būti įrašytas į programos komandų sritį. Jeigu simbolis įrašomas ant jau įvykdytų instrukcijų, kurios nebebus vykdomos, problemos nebus, tačiau, bandant vykdyti „sugadintas“ instrukcijas, programos darbas sutriks, rezultatai bus nenuspėjami.
- Operacinės sistemos duomenis.
- Operacinės sistemos kodą.

Piktavalis asmuo, suprantantis, kokias pasekmes gali turėti buferio perpildymas, gali pasinaudoti programos klaida. Pvz., ketvirtuoju atveju galima taip parinkti duomenis, kad jų reikšmė operacinės sistemos kodo srityje būtų korektiškos, vykdomos instrukcijos. Kadangi sistemos teisės paprastai didesnės už jos vykdomų programų, įsibrovėlis gali priversti sistemą įvykdyti piktavališkas komandas ir taip perimti jos valdymą. Buferio perpildymas išsamiai aprašytas Mudge knygoje [MUD95].

Nepilna kontrolė – tai situacija, kada programa negali kontroliuoti jai perduodamų duomenų. Paprastai ši situacija aktuali interneto sistemoms, kur duomenys perduodami per adreso eilutę. Serveris negali atskirti, ar užklausą atsiuntė naršyklė, ar ji buvo specialiai modifikuota vartotojo, siekiančio paveikti sistemą. Taigi labai svarbu tikrinti ne tik interneto programoms perduodamų duomenų tipą, bet ir reikšmių ribas.

Klaidos dėl pasenusių patikrinimo rezultatų (time-of-check to time-of-use). Tai tokia problema, kuri kyla dėl esančio laiko tarpo tarp veiksmo legalumo patikrinimo ir paties veiksmo. Pvz., vartotojas A gali ištrinti savo sukurtus failus, tačiau negali trinti vartotojo B sukurtų failų. Vartotojas gali nurodyti vykdyti komandą trinti savo failą „F.txt“. Programa pirmiausiai patikrina, ar failo „F.txt“ autorius yra A. Jeigu taip, failas ištrinamas. Kadangi kompiuteryje vienu metu gali veikti daug programų, po failo autoriaus patikrinimo procesorius gali laikinai nutraukti programos vykdymą ir pradėti/pratęsti kitos programos vykdymą. Kai failą trinančios programos vykdymas bus pratęstas, failo autoriaus patikrinimo

rezultatas jau gali nebegalioti, pvz., vartotojas galėjo spėti pakeisti failą „F.txt“ vartotojo B failu ir programa jį ištrins vadovaudamasi prieš tai atlikto patikrinimo rezultatu. Šią problemą galima spręsti keliais būdais:

- Neleisti naudotis duomenimis kitiems procesams, kol komanda nebus pilnai įvykdyta (duomenų bazių transakcijos);
- Naudoti maišos algoritmus, kad nustatyti, ar duomenys nebuvo pakeisti;
- Veiksmus atlikti su duomenų kopijomis, kurių vartotojas negali pasiekti.

2.4.2 Virusai ir kitas piktavalis kodas

Virusai – tai programos, kurios parašytos su piktavališkais ketinimais, siekiant sutrikdyti sistemų darbą, sugadinti duomenis ir pan. Sukurta programinė įranga gali ne tik atlikti specifikacijoje numatytus, bet ir piktavalius veiksmus, pvz., slapta rinkti privačius duomenis. Su pasitikėjimo programine įranga problema susidurta ir realizavus CTT sistemą.

Šiame skyrelyje nurodyti piktavalių programų tipai ir jų daroma žala.

Virusas – tai programa, kuri gali atkartoti savo piktavalį kodą kitose – neužkrėstose – programose modifikuodama jų kodą. Virusai skirstomi į nuolatinius (resident) ir laikinus. Nuolatiniai virusai glūdi atmintyje ir veikia kaip atskiros programos. Laikini virusai veikia tik tada, kada veikia apkrėstos programos. Kol veikia užkrėstoji programa, virusas ieško kitų programų ir jas modifikuoja.

Trojos arklys – tai programa, kuri be savo pagrindinės paskirties atlieka ir piktavališkus, vartotojui nematomus veiksmus. Pvz., programa, kuri reikalauja vartotojo prisijungimo duomenų, gali juos panaudoti ne tik vartotojui identifikuoti, bet ir išsaugoti juos su ketinimu vėliau pasinaudoti.

Loginė bomba (logic bomb) – tai programa, kuri nustoja veikti įvykdžius tam tikrą sąlygą; laiko bomba (time bomb) – loginė bomba, kurios vykdymo pabaigos sąlyga yra laikas.

Slaptas įėjimo/prisijungimo kelias (trapdoor, backdoor). Slapti būdai prisijungti prie sistemos gali būti sukurti siekiant tiek piktavališkų, tiek nepiktavališkų tikslų. Pvz., labai dažnai prisijungimas prie kompiuterio BIOS programos galimas ne tik naudojant administratoriaus sukurtą slaptažodį, bet ir pagrindinės plokštės gamintojo slaptą slaptažodį. Toks priėjimas sukurtas ne piktavaliais tikslais, o siekiant užtikrinti, kad net ir administratoriui pamiršus slaptažodį, išliktų priėjimas prie pagrindinės kompiuterio

programos. Pvz., prie AMI pagrindinių plokščių BIOS programos galima prisijungti naudojant slaptažodžius „A.M.I.“, „AAAMMMIII“ ir kitus.

Slaptas priėjimas gali būti sukurtas ir siekiant vėliau juo pasinaudoti šnipinėjimo, kenkimo tikslais. Slaptos prieigos gali būti sukurtos tam, kad būtų galima atlikti tokius veiksmus, kuriuos tipiniu atveju programa draustų.

Kirminu (worm) vadinama programa, kuri platina savo kopijas tinklu. Pirmieji kirmino sąvoką apibūdino Shock ir Hupp [SHO82]. Pagrindinis skirtumas nuo viruso yra tas, kad kirminas plinta tinklais, o virusas gali būti platinamas bet kokioje duomenų laikmenoje. Be to, kirminas platina save kaip atskirai veikiančią programą, o virusas paprastai plinta platinant juo užkrėstas programas.

Triušis (rabbit) – tai kirminas ar virusas, kuris platina save su tikslu „užgrobti“ kompiuterio resursus, pvz., daryti savo kopijas tol, kol diske nebeliks vietos arba atlikti beprasmius veiksmus, kad apkrauti procesorių.

Skirtumas tarp visų aukščiau išvardintų sąvokų nėra didelis, todėl terminai dažnai maišomi. Sąvoka „virusas“ dažnai naudojama kalbant apie bet kurį iš šių piktavalių programų tipą. Be to, virusas kartu gali būti ir laiko bomba, jeigu savo komandas vykdo po tam tikro programos-šeimininkės darbo laiko, ir kt.

Visos iš aptartų piktavalių programų nėra pavojingos tol, kol nėra vykdomos. Paprastai programoms pradėti vykdyti reikalingas žmogaus įsikišimas. Pvz., virusas gali būti aktyvuotas paleidus specialiai sukurtą tam tikros programos diegimo sistemą arba atsiųstas kartu su el. laišku, tikinančiu gavėją atsidaryti prisegtuką.

Laimei, virusai (piktavalės programos) negali būti visiškai nematomi. Viruso kodas turi būti kažkur saugomas, jis turi būti atmintyje, norint jį vykdyti. Antivirusinės programos stebi vykdomas komandas ir gali atpažinti virusus pagal jų kodą. Būtina pabrėžti, kad antivirusinė programa naudinga tik nuolat atnaujinant jos atpažįstamų virusų duomenų bazę.

2.4.3 Saugios PĮ kūrimo principai

Balfanz [BAL04] teigia, kad labai svarbu laikytis žemiau išvardintų punktų, norint užtikrinti, kad kuriamos sistemos bus saugios ir naudojamos:

- Negalima paveikti/modifikuoti jau naudojamų sistemų, t.y. negalima valdyti, taisyti vartotojui atiduotos sistemos.

- Įrankiai – ne sprendimas („Tools aren‘t solutions“)
- „Think locally; act locally“

Pagrindinis tikslas siekiant sistemų kokybės ir saugumo – projektuoti sistemas ir rašyti kodą nedideliais vienetais (units). Komponentas/modulis, kuris gali atlikti funkcijas nepriklausomai nuo kitų vienetų, yra lengviau palaikomas: greičiau randamos klaidos, pakeitimai neįtakoja kitų sistemos dalių. PĮ kūrimas iš savarankiškų vienetų ne tik gerina bendrą produkto kokybę, bet ir jo saugumą. Kuriant programinį vienetą, reikėtų siekti šių tikslų:

- Komponentas turėtų atlikti tik vieną konkrečią funkciją;
- Komponento dydis turėtų būti toks, kad jo turinį ir struktūrą būtų galima greitai perprasti;
- Komponentas turėtų būti nesudėtingas, kad jo paskirtis būtų aiški, struktūra – suprantama;
- Komponentas turėtų būti nepriklausomas arba kuo galima mažiau priklausomas nuo kitų modulių.

Žiūrint iš saugumo pusės, programų kūrimas moduliais labai padidina tikimybę, kad PĮ realizacija yra teisinga. Komponentų kūrimas yra naudingas tuo, kad:

- Juos yra lengviau pakeisti (jeigu komponentas atlieka vieną funkciją, jis gali būti lengvai pakeistas kitu – našesniu, saugesniu);
- Juos lengviau suprasti, „skaityti“;
- Juos lengva pakartotinai panaudoti kuriant kitas sistemas;
- Juos lengviau taisyti;
- Juos lengviau testuoti (komponentai, turintys apibrėžtus įėjimus ir išėjimus gali būti testuojami atskirai).

Vidiniai komponentų duomenys, metodai turi būti neprieinami, t.y. komponentas turi būti sukurtas taip, kad naudotis juo būtų galima tik iš anksto apibrėžtais ir dokumentuotais būdais. Paslėpta komponento realizacija apsaugo jį nuo netinkamų, piktavalių modifikacijų. Pavieniai vienetai gali būti atskirai išanalizuoti ir pripažinti saugiais.

Kad surasti galimas sistemos grėsmes (taip pat ir kitas klaidas), Pfleeger [PFL01] rekomenduoja į darbo procesą įtraukti šias veiklas:

- Nuolatinę peržiūrą,
- Rizikų analizę ir prognozavimą,
- Testavimą,
- Statinę kodo analizę,
- Produkto versijų kontroliavimą,
- Klaidų analizę.

Daugelis tyrėjų įrodė, kad peržiūra yra labai veiksminga. Pvz., Jones [JON91] apibendrino daugelio projektų informaciją ir sudarė lentelę (žr. lentelę 1), kaip peržiūra ir testavimas įtakoja surastų klaidų skaičių.

Lentelė 1. Peržiūros ir testavimo įtaka surastų klaidų skaičiui.

Veikla	**Surastų defektų skaičius (tūkstančiui kodo eilučių)**
Reikalavimų peržiūra	2,5
Sistemos modelio (architektūros) peržiūra	5,0
Kodo inspektavimas (planavimas, individualus pasiruošimas, sprendimų dokumentavimas, perdarymas, pakartotinė peržiūra ir t.t.)	10,0
Integravimo testavimas	3,0
Priėmimo testavimas	2,0

Gera saugių sistemų kūrimo praktika laikoma:

- Tolerancija klaidoms;
- Nuosekli, pastovi klaidų apdorojimo politika (pvz., sistemos būsenų saugojimas ir galimybė atstatyti paskutinę korektišką būseną);
- Projektavimo šablonų (design patterns) naudojimas;
- Projektavimo metodų ir pasirinktų sprendimų aprašymas. Dokumentavimas informuoja, kodėl sistema buvo suprojektuota taip, o ne kitaip. Ši informacija ypač naudinga vystant sistemą, taip pat projektavimo metodų istorija padeda rasti šablonus, kurie labiausiai tinka vienoje ar kitoje situacijoje. Tokie šablonai gali būti sėkmingai pakartotinai panaudoti naujų sistemų kūrimo metu.

Programinėje įrangoje pasirodžiusias klaidas galima apdoroti trimis būdais:

- Pakartojimu. Sistemos būsena atstatoma į prieš klaidą buvusią būseną ir veiksmas kartojamas naudojant kitą strategiją.
- Taisymu. Sistemos būsena atstatoma į prieš klaidą buvusią būseną, keičiami funkcijos vykdymo parametrai ir funkcija vykdoma iš naujo naudojant tą pačią strategiją.
- Pranešimu. Sistemos būsena atstatoma į prieš klaidą buvusią būseną, klaida fiksuojama įvertinimui ir funkcija nebekartojama.

Pagrindinės veiklos, užtikrinančios PĮ saugumą ir kokybę yra šios:

- Prognozavimas,
- PĮ versijų kontrolė,
- Mokymasis iš klaidų,
- Kokybės standartų naudojimas.

Kiekvienas iš punktų trumpai apibūdintas toliau.

Prognozavimas. Kad galėtumėte spręsti iškilusias problemas, pirmiausiai reikia numatyti galimas rizikas. Numačius galimas nepageidautinas situacijas, galima imtis veiksmų šioms situacijoms išvengti ar jų sukeltiems neigiamiems padariniams sušvelninti. Saugumo požiūriu rizikos analizė yra ypač svarbi, todėl kad nepageidaujami veiksniai beveik visada sukelia neigiamų padarinių.

PĮ versijų kontrolė. Kuriant programinę įrangą, svarbu žinoti, kas atlieka kodo modifikacijas ir kuriuose programos vienetuose. Versijų kontrolė yra procesas, kuris kontroliuoja PĮ pakeitimus jos kūrimo ir palaikymo metu. Versijų kontrolės sistemos atskiria naują ir seną kodą ir taip padeda užtikrinti, kad pakeitimų metu nebuvo įvelta naujų klaidų, tiek atsitiktinai, tiek norint specialiai pakenkti.

Mokymasis iš klaidų. Lengviausiai sustiprinti PĮ saugumą galima mokantis iš klaidų. Projektuojant sistemas rekomenduojama dokumentuoti pasirinktus sprendimus, ne tik tai, ką ir kodėl nusprendėme daryti, bet ir tai, ko ir kodėl nusprendėme nedaryti. Eksploatuojant sistemą, pasirodžiusios klaidos gali padėti suprasti, kurie projektavimo sprendimai lėmė jų atsiradimą. Iškilusių problemų dokumentavimas gali padėti ne tik mums, bet ir kitiems komandoms nariams išvengti anksčiau pasirodžiusių klaidų.

Kokybės standartų naudojimas. Norint pagerinti, užtikrinti kuriamų programų kokybę, buvo sukurti standartai, aprašantys, kaip valdyti projektus. Programų inžinerijos institutas sukūrė CMM modelį (Capability Maturity Model), vertinantį organizacijas [PAU93]. Tarptautinė standartų organizacija (International Standards Organization, sutr. ISO) sukūrė panašų įmonės procesų valdymo standartą ISO 9001 [ISO06]. Saugių sistemų kūrimo procesus standartizavo JAV Nacionalinio saugumo agentūra (U.S. National Security Agency) sukurdama SSE CMM modelį (System Security Engineering CMM) [CMM01].

Svarbu pabrėžti, kad visi šie standartai nagrinėja ne organizacijų kuriamus produktus, o tai, kaip tie produktai kuriami. Standartų įdiegimas gali padėti užtikrinti kuriamų produktų kokybę, programinės įrangos saugumą.

2.4.4 PĮ testavimas

Pagrindinis PĮ testavimo tikslas – užtikrinti produkto kokybę. Sistemos testuojamos norint pašalinti klaidas, taip pat užtikrinti, kad sistemos bus tolerantiškos nepastebėtoms klaidoms. Testavimas ypač svarbus sistemoms, reikalaujančioms ypatingo saugumo. Kiekviena nepastebėta klaida gali neigiamai paveikti ne tik verslą, bet ir žmogaus gyvybę.

Testavimas paprastai vykdomas keliais etapais. Pirmiausiai testuojami smulkiausi programos vienetai – klasės (unit testing, module testing arba component testing). Integravimo testavimo (integration testing) metu tikrinama, ar sistemos komponentai teisingai naudojami, ar veikia kartu. Kai apjungti komponentai veikia tinkamai, sistema testuojama funkciniu požiūriu, t.y. tikrinama, ar sistema atlieka reikalavimuose numatytas funkcijas. Testavimo rezultatas – pilnai funkcionuojanti sistema. Paskutinis testavimo etapas – galutinio vartotojo įvertinimas (priėmimo testavimas, acceptance testing). Vartotojas įvertina, ar sukurtas produktas atitinka jo lūkesčius. Programinės įrangos testavimas yra labai plati tema, šiame darbe jis tik paminėtas kaip viena svarbiausių priemonių PĮ kokybei ir saugumui užtikrinti. Svarbu paminėti, kad saugumo požiūriu labai vertinamas nepriklausomų testuotojų atliktas produkto testavimas. Toks testavimas paprastai apsaugo nuo mėginimų paslėpti piktavalį kodą, neleidžia kontroliuoti testinių atvejų aibės. Nepriklausomas testavimas labai padidina tikimybę, kad bus surastos paslėptos programos savybės.

Saugumo testavime unikalus yra vadinamasis įsibrovimo testavimas (penetration testing). Šio tipo testavimo metu testuotojai stengiasi priversti sistemą suklysti, t.y. tikrinama ne tai, ar sistema vykdo numatytas funkcijas, o tai, ar sistema nevykdo nenumatytų funkcijų. Pavyzdžiui, reikalavimuose numatyta, kad sistema tam tikrą funkciją gali įvykdyti tik įvedus teisingą slaptažodį. Tipinio testavimo metu bus laikoma, kad sistema veikia tinkamai, jeigu,

įvedus slaptažodį, sistema įvykdys aprašytąją funkciją. Įsibrovimo testavimo tikslas – parodyti, kad aprašytoji funkcija nebus vykdoma visais kitais atvejais.

2.5 Duomenų saugumas

2.5.1 Duomenų bazių saugumas

Duomenų baze vadinama duomenų aibė ir taisyklių, kaip duomenys susieti tarpusavyje, rinkinys. Taisyklėmis vartotojas apibrėžia loginį duomenų formatą. Duomenys saugomi failuose, tačiau tiksli jų fizinė struktūra vartotojui yra nežinoma ir nesvarbi. Duomenų bazės administratorius yra asmuo, aprašantis duomenų valdymo taisykles ir valdantis duomenų prieinamumo teises. Vartotojas duomenų bazėje saugomus duomenis pasiekia per duomenų bazės valdymo sistemą (Database Management System, sutr. DBMS). Su duomenų bazės valdymo sistema (toliau DBVS) vartotojas sąveikauja naudodamas komandas, kurios grąžina, įterpia, trina duomenis. Komandų sudarymo taisyklės yra griežtai apibrėžtos DBVS.

Reikalavimai duomenų bazių saugumui labai skiriasi nuo reikalavimų kitų tipų programinei įrangai. Pagrindiniai reikalavimai duomenų bazių saugumui yra šie:

- Fizinis duomenų bazės vientisumas. Duomenų bazėje laikomi duomenys turi būti atsparūs fizinėms problemoms, pvz., elektros energijos praradimui. Duomenis turi būti galima atstatyti net ir po katastrofiško sugadinimo.
- Loginis duomenų vientisumas. Duomenų bazės struktūra turi būti visada išlaikyta.
- Elementų vientisumas. Bet kurių duomenų tikslumas turi būti baigtinis. DBVS turi užtikrinti, kad duomenų elementai yra apibrėžto tipo ir jų reikšmės yra nurodytame intervale.
- Apskaitos kontrolė. Turi būti galima susekti, kas ir kokiais duomenimis naudojosi, juos pakeitė.
- Duomenų bazės prieigos kontrolė. Vartotojas turi galėti skaityti, keisti, trinti duomenis tik tuo atveju, jeigu jam suteikta atitinkama teisė.
- Vartotojų autentifikavimas. Duomenų bazės valdymo sistema turi turėti vartotojų autentifikavimo mechanizmą, kad galėtų nustatyti duomenų prieigos teises ir sekti duomenų pakeitimus.
- Prieinamumas. Vartotojas turi galėti pasiekti/modifikuoti duomenis, jeigu turi tokią teisę.

Viena didžiausių problemų duomenų bazių valdymo sistemoms – nesėkmė duomenų atnaujinimo metu. Vieno elemento atnaujinimo metu dalis saugomos reikšmės gali būti atnaujinta, dalis ne. Net jeigu tokias klaidas būtų galima lengvai pastebėti, didesnė problema iškyla viso įrašo (kortežo) atnaujinimo metu. Kadangi dalis elementų gali būti pilnai atnaujinti, tokias klaidas pastebėti praktiškai neįmanoma. Duomenų atnaujinimo problemai spręsti Lampson ir Sturgis [LAM76] pasiūlė dviejų žingsnių metodą, kurį naudoja dauguma DBVS. Pirmo žingsnio metu DBVS surenka visus reikalingus duomenis bei resursus atnaujinimui atlikti, pvz., sukuria fiktyvius įrašus, atidaro atitinkamus failus, uždraudžia prieigą prie duomenų kitiems procesams ir kt. Šis žingsnis reikalui esant gali būti kartojamas, nes jo metu jokie ilgalaikiai pakeitimai duomenų bazėje neatliekami. Jeigu pirmojo žingsnio metu sistemoje įvyksta klaida, duomenims nepadaroma jokia žala. Antrojo žingsnio metu atliekami ilgalaikiai pakeitimai. Jokie pirmojo žingsnio veiksmai nebegali būti kartojami, tačiau antrojo žingsnio veiksmai gali būti kartojami daug kartų, nes jie priklauso tik nuo pirmojo žingsnio metu surinktos informacijos ir tuo metu buvusios duomenų būsenos. Jeigu sistemoje įvyksta klaida antrojo žingsnio metu, duomenys gali būti nepilni ar sugadinti, tačiau jie gali būti ištaisyti pakartojus antrojo žingsnio veiksmus. Įvykdžius abu žingsnius, duomenys yra pilni ir korektiški.

Dažniausiai duomenų bazės yra naudojamos daugelio vartotojų vienu metu. Lygiagretaus duomenų naudojimo problema sprendžiama realizuojant duomenų įrašymo, atnaujinimo ir trynimo veiksmus kaip atominius (nedalomus) veiksmus. Pvz., vienas vartotojas gali atnaujinti tam tikro elemento reikšmę, o kitas vartotojas gali tuo metu skaityti elemento reikšmę. Perskaityti duomenys gali būti tik iš dalies atnaujinti, jeigu skaitymas vyko duomenų atnaujinimo metu. Taigi duomenų atnaujinimas, įrašymas ir trynimas privalo būti atliekamas tik užtikrinus, kad kiti procesai negalės pasiekti informacijos, kol ši nebus pilnai atnaujinta, įrašyta ar pašalinta.

2.5.2 Privačių asmens duomenų saugumas

Kas yra privatumas? Privatumu paprastai vadinama galimybė kontroliuoti, kas turi informaciją apie jus, jūsų ryšius ar veiklą. Kitais žodžiais tariant, jūs patys galite rinktis, kas ir kokią informaciją apie jus gali žinoti. Jeigu asmuo prašo jūsų telefono numerio, jūs sprendžiate, ar duoti jį, ar ne. Taigi privatumu laikoma tai, kas susiję su jumis ir ką tik jūs pats galite kontroliuoti.

Visgi realiame pasaulyje jūs neturite pilnos galimybės kontroliuoti tą informaciją, kurią laikote privačia. Davus kam nors savo telefono numerį, jūsų galimybės toliau kontroliuoti jo

slaptumą labai sumažėja, nes tai jau priklauso ir nuo asmens, kuriam davėte numerį. Viskas, ką jūs galite padaryti, tai paprašyti numerį saugoti ir niekam kitam jo neskelbti, tačiau jūs negalite to kontroliuoti. Ši problema labai aktuali ir kompiuterių saugume. Bet kas, kas turi prieigą prie tam tikro objekto, gali jį perduoti, kopijuoti, modifikuoti be savininko leidimo.

Kas yra privati informacija priklauso nuo paties asmens. Kas vienam atrodo privatu, kitam gali pasirodyti nesvarbu. Paprastai žmonės privačia laiko šią informaciją:

- Finansinius duomenis, kreditinių kortelių, bankų duomenis;
- Sveikatos būklę, vartojamus vaistus, genetinius polinkius;
- Balsavimo rezultatus, narystę tam tikrose organizacijose;
- Išpažįstamą religiją, seksualinį polinkį;
- Pirštų atspaudus, fizinius duomenis;
- Dienoraščius, laiškus, pokalbius telefonu;
- Ryšius su teisininkais, finansininkais, dvasininkais;
- Mokymosi rezultatus, darbo rodiklius;
- Skaitymo, naršymo internete įpročius;
- Neteisėtas veiklas, kriminalinius įrašus.

Visa ši informacija laikoma privačia individualaus asmens požiūriu. Labai įvairi informacija gali būti laikoma privačia asmenų grupių, organizacijų, įmonių ir t.t. Pvz., įmonės gali saugoti produktų gamybos technologiją, klientų sąrašus, finansinius duomenis. Tokios įstaigos kaip mokykla, ligoninė gali saugoti mokinių, pacientų, donorų duomenis, valstybinės įstaigos – piliečių sumokamų mokesčių dalį ir t.t.

Rezgui [REZ03] išskiria šiuos pagrindinius dalykus, kurie turėtų užtikrinti privatumą darbo kompiuteriu metu (daugiausiai naršymo internete metu):

- Duomenų surinkimas. Duomenys turėtų būti surenkami tik vartotojui žinant apie tai ir su tuo sutinkant.
- Duomenų naudojimas. Duomenys turėtų būti naudojami tik iš anksto nurodytais tikslais.
- Duomenų saugojimas. Duomenys turi būti saugomi tik tol, kol jie reikalingi.
- Duomenų atskleidimas. Duomenys gali būti atskleidžiami tik įgaliotiems asmenims.
- Duomenų saugumas. Privatūs duomenys turi būti tinkamai saugomi.
- Prieigos kontrolė. Bet koks duomenų naudojimas turi būti kontroliuojamas.

- Stebėjimas. Visi duomenų naudojimo, modifikavimo atvejai turi būti fiksuojami.
- Politikos pasikeitimas. Nauja, mažesnį saugumą deklaruojanti politika negali būti taikoma jau surinktiems duomenims.

Bet kuris iš mūsų greičiausiai sutiks atskleisti savo adresą norėdamas įsigyti tam tikrą daiktą internetu, kadangi jį būtina žinoti, norint pristatyti prekes. Taip pat gali reikėti atskleisti savo kreditinės kortelės ar kitus duomenis. Pardavėjas turi galimybę atpažinti asmenį pagal surinktus duomenis ir pasinaudoti šia informacija, pvz., pradėti siūlyti panašias prekes į tas, kurias pirkote.

Informaciją, paliktą internete (blog'uose, komentaruose ir pan.), labai sunku suvaldyti. Net ir ištrynus ją iš pradinio šaltinio, gali likti daugybė informacijos pėdsakų: kiti asmenys galėjo nukopijuoti informaciją, ją įsiminti, patalpinti kituose informacijos šaltiniuose ir pan. Labai svarbu rūpintis tuo, ką rašome internete, nes pašalinti informaciją gali būti praktiškai neįmanoma – jūs prarandate galimybę kontroliuoti informacijos panaudojimą. Privačios informacijos kontrolė gali būti prarasta net jums to ir nežinant, pvz., įstaigos, saugančios jūsų privačius duomenis, išpuolio metu gali būti nutekinti jūsų duomenis. Įstaiga greičiausiai nepasirūpins informuoti jūsų apie išpuolį, juo labiau nekompensuos jums padarytą žalą.

Paprastai privačių duomenų naudojimo sąlygos sulygstamos ar nurodomos saugumo politikoje, tam tikruose pareiškimuose (svetainėse jie randami puslapių apačioje) ir pan.

Duomenų naudojimo politika laikoma teisinga, jeigu ji atitinka šiuos punktus:

- Duomenys surenkami teisėtai ir sąžiningai.
- Duomenys tiesiogiai susiję su jų surinkimo tikslu.
- Duomenų surinkimo tikslai turi būti aiškiai nurodyti. Nebereikalingi duomenys turi būti pašalinti.
- Informacijos naudojimas kitais tikslais nei nurodyta galimas tik vartotojui leidus.
- Privati informacija yra saugoma nuo praradimo, sugadinimo, nutekėjimo.
- Privačių duomenų savininkas gali sužinoti, kaip saugomi ir naudojami jo duomenys.
- Duomenų savininkas gali pakeisti, pašalinti savo duomenis.
- Yra numatyti politikos laikymosi kontroliavimo būdai.

Turn ir Ware [TUR75] siūlo keturis būdus, kaip apsaugoti saugomus duomenis:

- Visada saugoti kuo galima mažiau duomenų.
- Sumažinti duomenų jautrumą keičiant duomenis arba įveliant sunkiai pastebimų klaidų.
- Nesaugoti asmenis identifikuojančių duomenų (asmens kodų, socialinio draudimo pažymėjimų numerių, adresų ir pan.).
- Šifruoti duomenis.

2.5.2.1 Europos Privatumo Direktyva (European Privacy Directive)

1981 m. Europos Taryba (tuo metu 46-ias Europos valstybes vienijantis organas) priėmė konvenciją[1], deklaruojančią asmens duomenų naudojimą. 1995 m. Europos Sąjunga priėmė direktyvą 95/46/EC, dažnai vadinamą Europos Privatumo Direktyva (European Privacy Directive), reikalaujančią, kad būtų laikomasi individualių asmenų privatumo ir kad duomenys apie juos būtų:

- Apdorojami sąžiningai ir teisingai.
- Surenkami tiksliai nurodytais ir tik teisėtais tikslais. Duomenys negali būti apdorojami siekiant nenumatytų tikslų, nebent imamasi specialių priemonių, garantuojančių privatumą.
- Tiesiogiai susiję su surinkimo tikslais ir neperteklinai.
- Tikslūs ir, kur reikia, patys naujausi. Privaloma imtis priemonių, kad užtikrinti, jog netikslūs ar nepilni duomenys bus ištaisyti arba pašalinti.
- Laikomi tokioje formoje, kuri leidžia identifikuoti subjektus tik tiek laiko, kiek reikia nurodytiems tikslams pasiekti.

Be to, subjektai turi teisę pasiekti apie juos surinktus duomenis, ištaisyti netikslią ar nepilną informaciją. Direktyvoje taip pat nurodyta, kad:

- „Jautrių duomenų“ surinkimui ir apdorojimui turi būti taikomi griežtesni apribojimai. Jautriais duomenimis Europos Sąjungos duomenų apsaugos direktyva vadina tokius duomenis, kurie nusako rasę, etninę kilmę, politines pažiūras, religinius, filosofinius, etinius įsitikinimus, sveikatos būklę, seksualinį gyvenimą ir kita.
- Draudžiama perduoti duomenis tretiesiems asmenims be savininko sutikimo.

[1] Konvencija – tarptautinė sutartis, reguliuojanti tam tikros srities tarpvalstybinius santykius.

- Subjektai, apdorojantys surinktus duomenys, turi būti atskaitingi ir prižiūrimi nepriklausomų subjektų. Pvz., vyriausybė turi būti atskaitinga departamentui, kuris kontroliuoja duomenų valdymo teisėtumą ir yra nepriklausomas nuo duomenis apdorojančių subjektų. Europos Sąjungos duomenų apsaugos direktyvoje nurodyta, kad nepriklausomas prižiūrėtojas turi turėti galią audituoti duomenis surenkančius ir apdorojančius subjektus, tirti individualius skundus bei taikyti sankcijas[2] susitarimų ir įstatymų laikymuisi užtikrinti.

Skirtingi įstatymai skirtingose teisinėse sistemose gali nesutapti, prieštarauti vienas kitam. Pvz., santykiai tarp Europos Sąjungos ir JAV kurį laiką buvo įtempti dėl Europos Sąjungos draudimo dalintis duomenimis su kompanijomis ir vyriausybėmis tų valstybių, kuriose privatumo įstatymai ne tokie griežti kaip Europos Sąjungoje. Visgi susitarimas buvo pasiektas sutikus vadovautis tam tikrais „saugaus bendradarbiavimo“ principais net ir neatitinkant Europos teisės.

Po 2001 m. rugsėjo 11-osios atakų JAV buvo priimtas įstatymas, įpareigojantis fotografuoti oro uostų keleivius, su tikslu atpažinti galimus teroristus. Informaciją apie kiekvieną keleivį įpareigotos rinkti avialinijų kompanijos. Šioje informacijoje nurodomi ne tik keleivio vardas, pavardė, skrydžio duomenys, bet ir telefono numeris, kreditinės kortelės duomenys, adresas ir kt. Kadangi europiečiai sudaro didžiąją dalį į JAV atvykstančių lankytojų, amerikiečiai pareikalavo Europos avialinijų bendrovių pateikti atvykstančiųjų duomenis per 15 minučių nuo lėktuvo pakilimo. Kadangi Europos Sąjungos direktyvos draudžia perduoti informaciją ar ją naudoti kitais tikslais nei ji buvo surinkta, JAV reikalavimas akivaizdžiai pažeidžia direktyvą. 2004 m. Europos Taryba leido oro transporto bendrovėms suteikti šiuos duomenis valstijoms. Vis dėl to, prieštaraujant Europos Parlamentui, 2006 m. gegužės 30 d. Aukščiausiasis Europos Teismas (European Court of Justice) nustatė, kad Europos Taryba neturėjo įgaliojimų sudaryti tokį susitarimą su JAV. Kadangi avialinijų kompanijos duomenis rinko komerciniais tikslais, Europos Sąjunga negalėjo leisti naudoti šių duomenų kitais tikslais, net ir siekiant užtikrinti JAV saugumą [CLA06]. Tačiau JAV, negavusi reikalaujamų duomenų, galėjo tiesiog uždrausti šalyje leistis nebendradarbiaujančių avialinijų lėktuvams. Vis dar laukiama galutinio privatumo principų susikirtimo problemos sprendimo.

[2] Sankcija – poveikio ar kontrolės priemonė (gali būti ekonominės, administracinės ir kitos sankcijos); nustatyta pažeidimo pasekmė: bauda, civilinė atsakomybė ir pan.

2.5.2.2 Saugumas internete

Mulligan [MUL99] nurodo keletą priežasčių, kodėl žmonės siekia anonimiškumo internete. Svarbiausia tai, kad internete galima nebijoti diskriminacijos, dėl kurios realiame pasaulyje būtų sunku įsidarbinti, pirkti paslaugas ar pan. Anonimiškumas taip pat gali sukelti ir problemų, pvz., kaip anonimas gali sumokėti už tam tikras prekes? Trečioji šalis, pvz., nekilnojamo turto agentas ar teisininkas, galėtų vykdyti pirkimą ir išlaikyti tikrojo pirkėjo anonimiškumą, tačiau tada trečioji šalis turi žinoti pirkėjo duomenis. Chaum [CHA81, CHA85] tyrinėjo šią problemą ir sugalvojo tam tikrus būdus, kaip pirkti/parduoti neatskleidžiant pardavėjui pirkėjo tapatybės.

Kartais pilnas anonimiškumas nėra pageidaujamas. Pvz., asmuo vieną kartą užsisakęs gėlių nenori atskleisti savo tapatybės, tačiau kitą kartą pirkdamas gėles nori nurodyti, kad pageidauja gauti tokios pačios spalvos gėlių kaip ir paskutinio užsakymo metu. Tokia situacija vadinama pseudo-anonimiškumu, kada specialūs identifikatoriai gali būti naudojami susieti konkretiems įrašams (užsakymams) tarpusavyje, tačiau negali būti panaudoti tikrajai kliento tapatybei nustatyti.

Identifikatoriai. Dauguma žmonių jau turi keletą juos identifikuojančių duomenų. Banke asmuo gali būti atpažintas kaip sąskaitos, kurios numeris 123456, savininkas, transporto priemonių registravimo biure asmuo gali būti atpažintas kaip vairuotojo pažymėjimo, kurio numeris 234567, savininkas ir t.t. Šie skaičiai tarnauja kaip asmens tapatybės duomenys. Tikrasis asmens vardas ar pavardė nebūtinai yra saugomi kartu su identifikatoriais, todėl susieti konkretų asmenį su identifikatoriumi gali būti sunku. Identifikatoriai tampa jautriais duomenimis, jeigu jie gali būti susieti su kitose duomenų bazėse saugomais duomenimis.

Mokėjimai internetu. Mokėdami internetu, vartotojai paprastai nurodo kreditinės kortelės duomenis. Vartotojas negali būti tikras, kad šiuos duomenis pardavėjas saugos ir nepasinaudos jais neteisėtoms bankinėms transakcijoms atlikti. Šiuo metu plačiau naudojama sistema mokėjimams internetu atlikti yra PayPal (aukcionų svetainės eBay kontroliuojama įmonė). Registruoti vartotojai nurodo PayPal savo kreditinių kortelių duomenis ir įgyja identifikatorių sistemoje (šiuo metu identifikatorius yra el. pašto adresas). Pirkimo metu pirkėjas nurodo pardavėjo identifikatorių, o PayPal perveda pinigus. PayPal sistema negarantuoja tokios klientų-pirkėjų teisių apsaugos, kokią suteikia įprastos bankinės sistemos, tačiau užtikrina privačių duomenų saugumą, nes kreditinių kortelių duomenys žinomi tik PayPal.

Registracija svetainėse. Registracija interneto svetainėse šiuo metu yra įprastas dalykas. Dažniausiai registracija yra nemokama, jūs tiesiog pasirenkate savo identifikatorių ir slaptažodį. Populiariose svetainėse, tokiose kaip Yahoo ar MSN, registracijos privalumai yra abejotini: pritaikymas naršymo įpročiams, specialus turinys ir kt. Iš tiesų, svetainės nori išgauti vartotojų demografinius[3] duomenis, kuriuos galėtų parduoti prekybininkams ar reklamuotojams.

Dažnam asmeniui sunku prisiminti daugelyje svetainių naudojamus identifikatorius, todėl labai dažnai identifikatoriumi reikalaujama nurodyti el. pašto adresą. Daugelyje svetainių kaip raktą naudojant tą patį el. pašto adresą, galima apjungti skirtingų svetainių duomenų bazes. Be to, dažnai el. pašto adrese galima įžiūrėti tikrąjį asmens vardą ir pavardę, kas leidžia surinktus duomenis susieti su konkrečiu asmeniu.

Sausainėliai (cookies). HTTP protokolas praktiškai negali saugoti būsenų, t.y. naršyklė paprastai atvaizduoja vaizdą nepriklausomai nuo to, kas vyko anksčiau. Sausainėlis – tai konkrečios svetainės sukurtas tekstinis failas, saugomas kliento kompiuteryje. Vartotojui apsilankius svetainėje, naršyklė automatiškai perduoda šį failą. Svetainė gali sukurti neribotą kiekį sausainėlių ir kiekviename iš jų saugoti ne daugiau kaip 4096 baitus duomenų. Sausainėliai negali sutrikdyti kompiuterio darbo, jie nėra pavojingi. Visgi juose gali būti saugoma privati informacija: perskaityti straipsniai, peržiūrėti reklaminiai skelbimai, paieškos laukeliuose naudotos frazės ir pan. Kitaip tariant, sausainėliuose gali būti kaupiama įvairi konkrečios svetainės naršymo statistika (kiekviena svetainė gali matyti tik savo sukurtus sausainėlius). Kokie duomenys ir kaip jie saugomi pačiame sausainėlyje priklauso tik nuo svetainės. Naršymo statistika paprastai naudojama reklamai parinkti ir rodyti. Problema privatumui iškyla tada, kada svetainės valdytojai dalinasi sausainėliuose surinkta naršymo statistika su trečiomis šalimis ir galbūt net atskleidžia lankytojo tapatybę (jeigu turi tokius duomenis). Visgi tokia statistika gali būti netiksli, kadangi sausainėliai saugomi kompiuteryje, o šiuo gali naudotis keletas vartotojų. Taip pat lankytojas gali naršyti svetainę naudodamasis keliomis naršyklėmis, prisijungęs skirtingais identifikatoriais ar iš skirtingų kompiuterių.

Klavišų paspaudimus registruojančios (keystroke loggers) ir šnipinėjančios programos (spyware). Klavišų paspaudimo registruotojas yra programa, sauganti kiekvieno paspausto klaviatūros klavišo informaciją. Tokios programos gali labai paveikti privatumą

[3] Demografija – mokslas, tiriantis gyventojų (vartotojų) sudėtį, kiekybinius ir kokybinius jos struktūros kitimus tam tikromis sąlygomis.

nustatydamos įvedamus slaptažodžius, bankų sąskaitų numerius ir t.t. „Šnipinėjančios programos“ – tai bendresnis terminas kenkėjiškoms programoms apibūdinti, įtraukiantis ir klavišų paspaudimus registruojančias programas. Šnipinėjančios programos slapta registruoja vartotojo veiklą, tačiau nebūtinai kiekvieno klavišo paspaudimo lygiu. Pvz., programa gali fiksuoti lankomų svetainių adresus, išsiųsti surinktus duomenis apdorojimui ir priklausomai nuo jų vartotojas gali būti apipiltais specialiai jam skirtais reklaminiais laiškais, skambučiais ir pan. Reklamos tikslais šnipinėjanti programa dar vadinama reklaminiu kenkėju (adware).

Kompiuteryje slapta veikiantys šnipai gali labai paveikti jo saugumą. Gali būti pavogta vartotojo tapatybė ar atlikti kiti nusikalstami veiksmai. Be to, šnipai gali labai apkrauti centrinio procesoriaus darbą, trukdyti veikti legalioms programoms ir pan.

Pirkimas internetu. Perkant prekes ar paslaugas internetu, pardavėjas turi galimybę sukčiauti nurodydamas pirkinio kainą. Jums matoma kaina gali būti skirta tik jums, t.y. kitam pirkėjui gali būti pasiūlyta kita kaina. Turow [TUR05] atlikta studija rodo, kada kainų diskriminacija paprastai atsiranda, kai pardavėjas surenka vis daugiau informacijos apie pirkėją. Net ir tokia populiari svetainė kaip amazon.com naudojosi vartotojų pirkimo statistika kainai nustatyti: naujiems lankytojams buvo siūloma 40% nuolaida, o sugrįžtantiems – 30% nuolaida. Akivaizdu, kad prisivilioti naują klientą yra sunkiau, negu išlaikyti lojalų klientą.

Turow atliktos 1500 JAV suaugusiųjų apklausos, susijusios su kainomis internete ir pirkimo kausimais, svarbesni pastebėjimai:

- 50% apklaustųjų manė, kad negali valdyti pardavėjų surinktų duomenų apie juos.
- 38% apklaustųjų manė, kad yra teisėta tuo pačiu metu skirtingiems vartotojams taikyti skirtingas kainas.
- 36% apklaustųjų manė, kad prekybos centrai turi teisę parduoti klientų pirkimo įpročius apibūdinančius duomenis.
- 29% apklaustųjų manė, kad video punktams nėra draudžiama parduoti informaciją, kokius video įrašus klientas išsinuomojo.

Elektroninio pašto saugumas. Elektroninis paštas abstrakčiai yra komunikavimas taškas-į-tašką. Jeigu siuntėjas A siunčia el. laišką gavėjui B, siuntėjo kompiuteris užmezga ryšį su gavėjo kompiuteriu ir pranešimas perduodamas SMTP (simple mail transfer protocol) pagalba. Visgi laiško siuntimo metu gavėjo kompiuteris gali būti išjungtas. Tokiu atveju el. laiškas išsaugomas POP (post office protocol) serveryje. Kai gavėjas prisijungs prie tinklo, pranešimas bus parsiųstas iš serverio. Bendravimo taškas-į-tašką būdu pranešimas yra

privatus, o serveryje saugomam laiškui kyla grėsmė būti atskleistam. Elektroniniams laiškams apsaugoti naudojamos dvi pagrindinės sistemos: PGP ir S/MIME. Šie produktai apsaugo pranešimus juos užšifruodami (naudojamas RSA šifravimo algoritmas).

Elektroninių laiškų siuntėjams patikrinti praktiškai nėra galimybių, t.y. SMTP protokolas netikrina, ar siuntėjo nurodytas el. pašto adresas egzistuoja, ar apskritai yra teisingai nurodytas. Tai labai palengvina elektroninio pašto šiukšlių (spam) siuntimą, nes tikrąjį laiško siuntėją nustatyti praktiškai neįmanoma.

Duomenų vagystė „phishing“ (angl. terminas phishing kilęs nuo žodžių password fishing – slaptažodžių žvejyba) – tai tokia sukčiavimo forma prieš organizacijas ar privačius asmenis, kai pasinaudojant nepageidaujamomis elektroninio pašto žinutėmis ar falsifikuotais internetiniais tinklalapiais siekiama išgauti prisijungimo prie informacinių sistemų slaptažodžius ar kitus konfidencialius duomenis. Dažniausiai tokio pobūdžio atakos būna nukreiptos prieš bankų klientus, siekiant sužinoti jų prisijungimo prie elektroninės bankininkystės sistemų slaptažodžius ar kreditinių kortelių duomenis. Vėliau tokiu būdu gauta informacija gali būti panaudota pasipelnymo tikslais vykdant nusikalstamas veikas: neteisėtus prisijungimus prie informacinių sistemų, pinigų vagystes iš sąskaitų ar elektroninėje erdvėje atsiskaitant už prekes svetimomis kortelėmis. Suklastotos svetainės (spoofed) gali būti identiškoms tikrosioms savo išvaizda, tačiau jų internetinis adresas skiriasi, pvz., vietoje tikrojo adreso vbankas.seb.lt gali būti naudojamas adresas vbankaas.seb.lt. Tokio tipo sukčiavimui valdyti praktiškai nėra priemonių, todėl vartotojo žinojimas yra geriausia apsaugos priemonė.

2.6 Šifravimo ir dešifravimo algoritmai

Kripto[4] *sistema* vadinama sistema, kurią sudaro aibė taisyklių, kaip užšifruoti informaciją ir kaip dešifruoti *šifrą*[5]. Šifravimo ir dešifravimo algoritmai dažnai naudoja slaptą raktą. Taigi šifras priklauso nuo originalios informacijos, šifravimo algoritmo ir rakto. Šią priklausomybę galime užrašyti taip:

$$C = E(K, P)$$

Čia C yra šifras, P – šifruojama informacija, E – aibė šifravimo taisyklių, K – šifravimo raktas.

[4] Kripto – pirmasis sudurtinių žodžių dėmuo, rodantis antruoju dėmeniu reiškiamos sąvokos sąsają su slaptumu.
[5] Šifras – kodas, kurio elementų reikšmė ir jo naudojimo taisyklės žinomos ribotam asmenų skaičiui; vartojamas informacijai apsaugoti.

Užkoduotai informacijai dešifruoti gali būti naudojamas tas pats raktas kaip ir informacijos šifravimo metu, tačiau nebūtinai. Algoritmai, kurie informacijai šifruoti ir dešifruoti naudoja tą patį raktą, vadinami simetriškais (symmetric) arba slapto rakto ir yra užrašomi taip:

$$P = D(K, E(K, P))$$

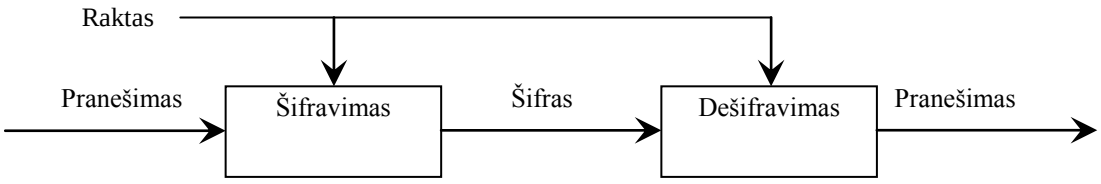
Čia *D* yra dešifravimo algoritmas.

Algoritmai, kurie informacijai šifruoti naudoją vieną raktą, o dešifruoti – kitą, vadinami asimetriškais (asymmetric) arba viešo rakto ir yra užrašomi tokia priklausomybe:

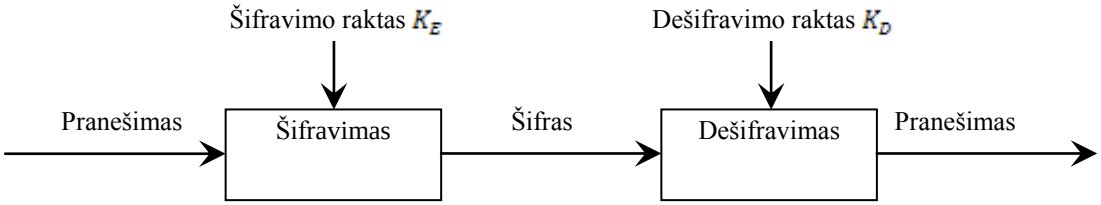
$$P = D(K_D, E(K_E, P))$$

Čia K_E yra šifravimo raktas, o K_D – dešifravimo raktas. Šie raktai yra priklausomi vienas nuo kito, todėl generuojami vienu metu.

Simetriškų ir asimetriškų algoritmų skirtumai pavaizduoti paveikslėlyje 2.



(a) Simetrinė kripto sistema



(b) Asimetrinė kripto sistema

Paveikslėlis 2. Simetrinė ir asimetrinė kripto sistemos.

Informacijos šifravimo algoritmai leidžia apsisaugoti nuo informacijos nutekėjimo. Net jeigu informaciją ir perimtų tretieji asmenys, jie negalėtų suprasti jos turinio, nes nežinotų slapto rakto.

Informacijai užšifruoti raktas gali būti ir nenaudojamas. Tokiu algoritmu užrašytą tekstą suprasti gali tik algoritmą žinantys asmenys. Tokie algoritmai paprastai paremti sutartinių

ženklų, skaičių naudojimu, raidžių praleidimu ar jų pakeitimu kitomis raidėmis, žodžių rašymu atbuline tvarka, akrostichais[6].

Kripto analizės tikslas – „nulaužti“ šifravimo algoritmą, t.y. dešifruoti šifrą nežinant dešifravimo algoritmo ar slapto rakto. Kripto analizės metu šifre ieškoma tam tikrų simbolių kombinacijų, pasikartojimų ir pan. Tokiu būdu stengiamasi išvesti dešifravimo formules, nustatyti šifravimo raktus. Kripto analitiko naudojamais informacijos šaltiniais gali būti užkoduoti pranešimai; perimti neužšifruoti pranešimai; informacijos elementai, kurie įtariama yra užkoduoti pranešime; matematiniai ir statistiniai įrankiai bei technikos ir kita.

Šifravimo algoritmas yra laikomas „nulaužiamu“, kai per duotą laiką kripto analitikas gali nustatyti dešifravimo algoritmą. Nors teoriškai bet kuris algoritmas gali būti laikomas „nulaužiamu“, tačiau stengtis tą padaryti gali būti nepraktiška dėl reikalingo milžiniško laiko resursų kiekio. Taigi rizikinga teigti, jog algoritmas yra saugus dėl to, kad jo negalima „nulaužti“ šiandienos technologinėmis priemonėmis, nes, bėgant laikui ir tobulėjant technologijoms, tai gali būti padaryta.

2.6.1 Sukeitimo algoritmai

2.6.1.1 Cezario algoritmas

Julijaus Cezario algoritmas turi didelę reikšmę šifravimo algoritmų istorijoje. Šis algoritmas yra labai nesudėtingas ir pagrįstas tuo, kad kiekviena raidė pranešime yra užrašoma kita raide, esančia per tris simbolius abėcėlėje nuo tikrosios raidės.

Pilnas Cezario šifras atrodė taip:

Tekstas: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Šifras: d e f g h i j k l m n o p q r s t u v w x y z a b c

Šis algoritmas yra labai paprastas ir įsimintinas, tačiau Julijaus Cezario laikais jis toks ir turėjo būti. Sudėtingi algoritmai netiko, nes tokius būtų reikėję užrašyti, o popieriuje saugomus algoritmų aprašymus būtų galima pavogti ar atimti iš pasiuntinių, kurie kartu su pranešimais būtų turėję perduoti ir dešifravimo algoritmų aprašus.

Sukeitimo algoritmus galima sustiprinti naudojant raktą. Paprastai pirmiausiai užrašoma abėcėlė, o po ja – slaptas raktas. Po likusiomis abėcėlėje raidėmis kitos raidės užrašomos tam

[6] Akrostichas – eiliuoto kūrinio eilučių pirmosios raidės, kurios, skaitant iš viršaus į apačią, sudaro žodį ar posakį.

tikra lengvai įsimenama tvarka. Kadangi viena abėcėlės raidė gali būti pakeista tik viena raide, rakte negali būti pasikartojančių simbolių. Toliau esančiame pavyzdyje matomas šifras, naudojantis raktą „katinas“. Raktas užrašytas kaip „katins“, nes antroji raidė „a“ kartojasi. Raidės po rakto užrašytos iš eilės pagal abėcėlę, praleidžiant panaudotąsias rakte.

Tekstas: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Šifras: k a t i n s b c d e f g h j l m o p q r u v w x y z

2.6.1.2 Vižinieriaus lentelė (Viginère Tableau)

Vižinieriaus algoritmas pagrįstas tuo, kad pranešimui užkoduoti naudojamas raktas yra tokio ilgio, kokio yra pranešimas. Koduojamas pranešimas po vieną simbolį rašomas virš rakto, o po to šie du simboliai pakeičiami kitu pagal Vižinieriaus lentelę (paveikslėlis 3). Kad pranešimo gavėjas perskaitytų pranešimą, jis turi žinoti raktą ir turėti Vižinieriaus lentelę.

Šio algoritmo problema yra ta, kad reikalingas labai ilgas raktas. Nors sugeneruoti didelį raktą nėra sunku, tačiau jį saugoti, perduoti yra problemiška.

	0					5					10					15					20					25	
	a	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*k*	*l*	*m*	*n*	*o*	*p*	*q*	*r*	*s*	*t*	*u*	*v*	*w*	*x*	*y*	*z*	π
A	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	0
B	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	1
C	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	2
D	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	3
E	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	4
F	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	5
G	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	6
H	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	7
I	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	8
J	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	9
K	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	10
L	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	11
M	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	12
N	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	13
O	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	14
P	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	15
Q	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	16
R	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	17
S	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	18
T	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	19
U	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	20
V	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	21
W	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	22
X	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	23
Y	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	24
Z	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	25

Paveikslėlis 3. Vižinieriaus lentelė.

2.6.1.3 Vernamo šifras (Vernam Cipher)

Vernamo šifras pagrįstas tuo, kad kiekviena pranešimo raidė yra apjungiama su vis kitu skaičiumi, gautu atsitiktine tvarka. Jeigu kiekvienam pranešimui užkoduoti naudosime vis kitą atsitiktinių skaičių seką, šifras bus atsparus leksikografinei analizei, nes pranešime pasikartojantys simboliai bus keičiami skirtingais simboliais priklausomai nuo atsitiktinio

skaičiaus. Šio šifravimo modelio procesas parodytas paveikslėlyje 4. Šifravimo proceso metu pranešimo raidės paverčiamos skaičiais pagal jų vietą abėcėlėje, t.y. A – 0, B – 1, ..., Ž – 31.

Tekstas:	K	A	T	I	N	A	S
Skaitinė reikšmė:	16	0	25	12	19	0	23
+ Atsitiktinis skč.:	13	67	23	47	62	39	52
= Suma:	29	67	48	59	81	39	75
= mod 32:	29	3	16	27	17	7	11
Šifras:	v	c	k	ų	l	ę	h

Paveikslėlis 4. Vernamo šifras.

2.6.1.4 Sukeitimo algoritmų kripto analizė

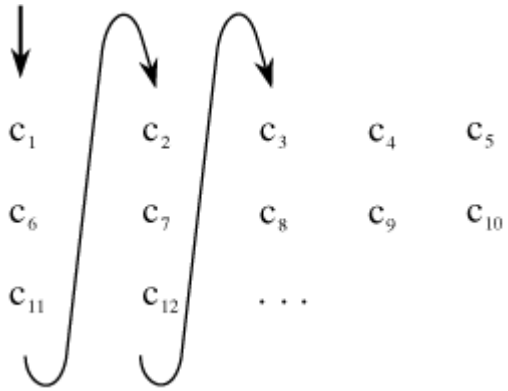
Sukeitimo algoritmų „nulaužimas“ priklauso nuo jų sudėtingumo. Sudėtingiausias tokio tipo algoritmo šifras gaunamas tada, kada viena raidė keičiama kita atsitiktine abėcėlės raide, t.y. nesilaikant jokių apibrėžtų taisyklių. Tokiu būdu galima gauti $32!$ skirtingus šifrus. Toks šifrų skaičius gaunamas laikant, kad abėcėlėje yra 32 raidės. Raidei A pakeisti kita raide turime 32 variantus, raidei B – 31 variantą (visos raidės, išskyrus tą raidę, kuria pakeitėme A), raidei C – 30 variantų (visos raidės, išskyrus tas raides, kuriomis pakeitėme A ir B) ir t.t. Tokiu būdu gauname $32 * 31 * 30 * \dots * 2 * 1 = 32!$ skirtingų būdų užšifruoti pranešimą. Kripto analitikas galėtų išbandyti visus variantus, tačiau, net ir per vieną mikrosekundę patikrinant po vieną variantą (100000 variantų per sekundę), visų variantų perrinkimas užimtų tūkstančius metų. Taigi perstatymo algoritmai paprastai analizuojami leksikografiškai, t.y. atsižvelgiama į tai, kurios raidės dažniausiai/rečiausiai naudojamos kalboje, ieškoma trumpų žodžių ir pan. Turint pakankamai užšifruoto teksto, kripto analitikas, pasitelkęs kompiuterines analizės programas, perstatymo algoritmu užšifruotą tekstą galėtų įveikti per mažiau nei vieną valandą. Visgi, turint trumpą pranešimą, iššifruoti jį gali būti labai sunku.

Vernamo šifras yra atsparus leksikografinei analizei, tačiau tik tuo atveju, jeigu kiekvieną kartą naudojamas vis naujas raktas (atsitiktinių skaičių seka). Taigi šį algoritmą naudoti praktikoje yra nepatogu, iškyla raktų saugojimo ir perdavimo problemos.

2.6.2 Perstatymo algoritmai

Sukeitimo algoritmų tikslas – užšifruoti pranešimą taip, kad būtų sunku nustatyti, kaip pranešimas ir raktas buvo apjungti, kad gauti šifrą. Perstatymo algoritmai pagrįsti tuo, kad informaciniai simboliai yra sukeičiami vietomis tam tikra tvarka.

Perstatymo stulpeliais algoritmo esmė – užrašyti tekstą eilutėmis po *n* simbolių, o po to perrašyti jį stulpeliais. Paveikslėlyje 6 pateiktas pavyzdys, kuriame tekstas užrašomas eilutėmis po 5 simbolius, o po to perrašomas stulpeliais. Šifravimo šiuo algoritmu laikas priklauso nuo pranešimo dydžio. Taip pat šifras gali būti baigtas tik tada, kada bus užrašytas visas pranešimas.



Paveikslėlis 5. Perstatymas stulpeliais.

T A I Y R
A P E R S
T A T Y M
O P V Z .

t a t o a p a p i e
t v y r y z r s m .

Paveikslėlis 6. Perstatymas stulpeliais.

Perstatymo algoritmai nekeičia pranešimo simbolių reikšmės, todėl kripto analizės metu daugiausiai reikia remtis žmogiška nuojauta. Pirmas žingsnis analizuojant šifrą – nustatyti raidžių pasikartojimo dažnius. Jeigu nustatoma, kad šifre esančių simbolių pasikartojimo dažniai sutampa su tikraisiais naudojamos kalbos simbolių pasikartojimo dažniais, galima manyti, jog šifravimui buvo panaudotas perstatymo algoritmas. Kitas žingsnis – suskaidyti šifrą stulpeliais. Šio proceso metu reikia labai nuodugniai nagrinėti šifro dalis. Pradinė dalis šifro yra lyginama su likusia šifro dalimi slenkant lyginamąją dalį po vieną simbolį dešinėn. Pavyzdžiui, jeigu pirmus septynis šifro simbolius nuo c_1 iki c_8 paimsime lyginamąja dalimi, ją toliau lyginsime su šifru nuo c_2 iki c_9, nuo c_3 iki c_{10} ir t.t. Po to lyginamąja dalimi imame vis daugiau simbolių ir kartojame palyginimus. Šis procesas pavaizduotas paveikslėlyje 7. Palyginimo metu ieškoma dviejų greta esančių simbolių, kurie kalboje turi prasmę. Pvz., anglų kalboje tokiais deriniais yra „EN“, „TH“, „IN“, „OR“, „ON“ ir kt. Suradus šifro dalį, galimai sutinkančią su lyginamąja dalimi, galima praplėsti paiešką. Jeigu surasta galimai tinkanti šifro dalis yra nuo c_i iki c_{i+k}, kur k yra lyginamosios dalies ilgis, galima manyti, kad ir šifro pozicijose nuo c_{i+k} iki c_{i+2k} galime rasti atitikimų su lyginamąja dalimi.

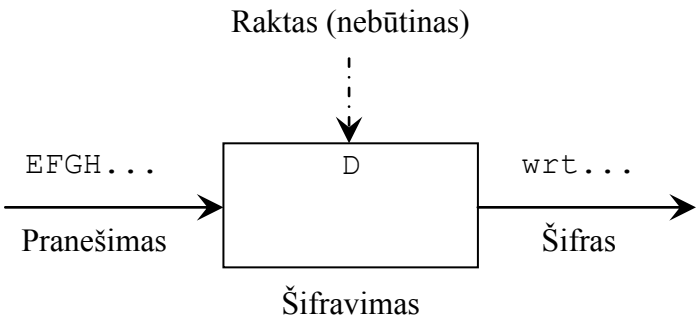
Nors perstatymo algoritmas atrodo nesudėtingas, tačiau dešifruoti juo užkoduotą pranešimą gali būti daug sunkiau nei sukeitimo algoritmų atveju.

```
t a t o a p
a p i e t v y r y z r s m . . .

  t a t o a p
a p i e t v y r y z r s m . . .

    t a t o a p
a p i e t v y r y z r s m . . .

      t a t o a p
a p i e t v y r y z r s m . . .
```

Paveikslėlis 7. Perstatymas stulpeliais. Digramų[7] paieška.

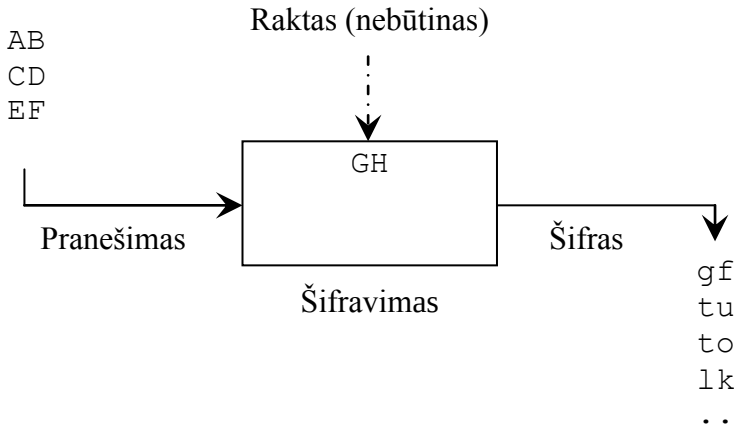
2.6.3 Srautiniai ir blokiniai algoritmai

Šifravimo algoritmai priklausomai nuo vienu metu apdorojamos informacijos kiekio taip pat skirstomi į srautinius (stream) ir blokinius (block). Visi aukščiau aprašyti algoritmai, išskyrus perstatymo stulpeliais algoritmą, vadinami srautiniais, nes simboliai vienas po kito keičiami kitu – užšifruotu – simboliu. Kaip gaunamas šifras priklauso tik nuo rakto, koduojamo simbolio ir algoritmo, bet nepriklauso nuo prieš ar už koduojamo simbolių esančių simbolių (paveikslėlis 8). Blokiniais algoritmais vadinami tokie, kurie informaciją užkoduoja blokais (paveikslėlis 9). Perstatymo stulpeliais algoritmas yra blokinis, nes, norint juo užkoduoti informaciją, reikia turėti iš karto visą pranešimą – informacijos bloką (visas pranešimas laikomas vienu bloku). Srautinių ir blokinių algoritmų privalumai bei trūkumai pateikti lentelėje 2.



Paveikslėlis 8. Srautiniai algoritmai.

[7] Digrama (gr. gramma – raidė) – dviejų raidžių junginys.



Paveikslėlis 9. Blokiniai algoritmai.

Lentelė 2. Srautinių ir blokinių algoritmų palyginimas.

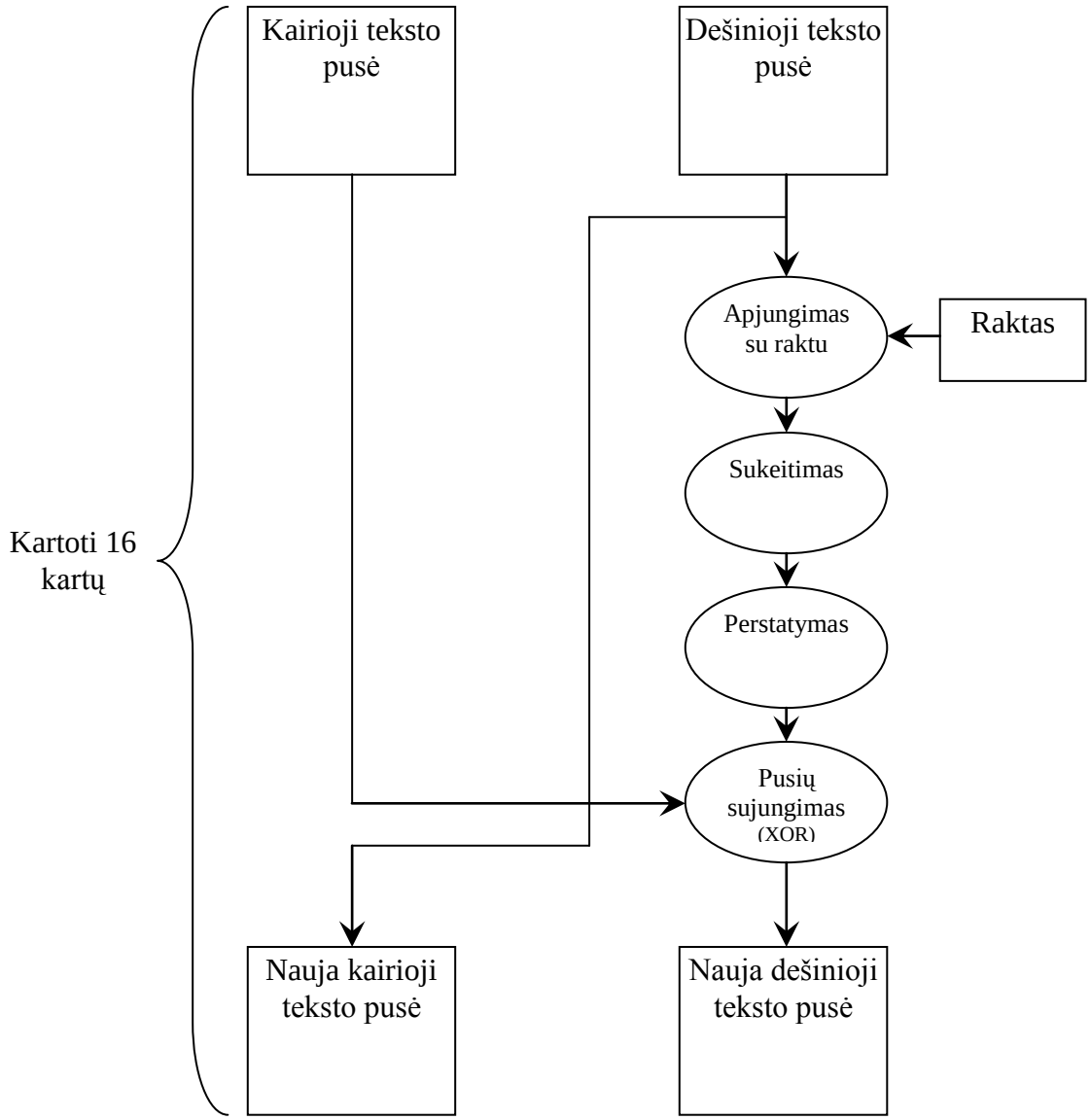
	Srautiniai algoritmai	**Blokiniai algoritmai**
Privalumai	Atsparūs klaidoms. Kiekvienas simbolis koduojamas atskirai, klaida šifravimo proceso metu paveikia tik koduojamą simbolį. Greiti. Laikas, reikalingas vienam simboliui užkoduoti, priklauso tik nuo algoritmo, t.y. nereikia laukti, kol bus gautas/perskaitytas tam tikras kiekis informacijos.	Vienas užšifruoto pranešimo blokas priklauso nuo kelių originalaus pranešimo simbolių – algoritmus sunkiau „nulaužti“. Atsparūs šifro modifikacijoms. Įterpus tam tikrą melagingos informacijos kiekį į šifrą, sugadinami kiti užšifruotos informacijos blokai, o tai lengvai pastebima dešifravimo proceso metu.
Trūkumai	Kadangi kiekvienas užšifruotas pranešimo simbolis priklauso tik nuo vieno originalaus pranešimo simbolio, algoritmus lengviau „nulaužti“. Neatsparūs šifro modifikacijoms. Į perimtą šifrą galima įterpti anksčiau perimtų šifrų dalis ir taip gauti modifikuotą/melagingą pranešimą.	Lėti. Kodavimas gali būti pradėtas tik gavus tam tikrą duomenų kiekį. Neatsparūs klaidoms. Klaida šifravimo proceso metu gali sugadinti visą duomenų bloką.

2.6.4 Data Encryption Standard (DES)

DES (DEA, DEA1). DES algoritmas pagrįstas sudėtinga sukeitimo ir perstatymo algoritmų kombinacija. Šifravimo algoritmas naudoja 64 bitų raktą. Iš tiesų raktu gali būti tik 56 bitų skaičius, 8 papildomi bitai naudojami kaip kontrolinė suma ir neįtakoja šifravimo algoritmo.

Šifruojant DES algoritmu, informacija apdorojama kartojant sukeitimo ir perstatymo algoritmus 16 kartų (paveikslėlis 10). Dešifruoti informaciją, kuriai daug kartų buvo pritaikyti

minėtieji algoritmai, praktiškai neįmanoma. Kadangi DES algoritmas operuoja 64 bitų skaičiais, jis buvo tinkamas įgyvendinti kompiuteriuose ir techninėje įrangoje.



Paveikslėlis 10. DES algoritmas.

Double DES. Laikui bėgant, 56 bitų raktas pasirodė per trumpas saugumui užtikrinti, tačiau DES algoritmas yra sukurtas taip, kad tegali naudoti fiksuotą 56 bitų raktą. Buvo pasiūlyta sustiprinti DES algoritmą naudojant du slaptus raktus, t.y. vieną kartą informaciją užšifruoti naudojant pirmąjį raktą, o po to gautą šifrą dar kartą užkoduoti naudojant antrąjį raktą:

$$C = E(k_2, E(k_1, P))$$

Čia C yra šifras, P – šifruojama informacija, E – DES algoritmas, k_1 ir k_2 – šifravimo raktai.

Teoriškai, tokiu būdu sudėtingumas dešifruoti informaciją turėtų padidėti du kartus, nes parinkti du raktus yra sudėtingiau negu vieną. Visgi tai netiesa. Merkle ir Helman [MER81] įrodė, kad dešifruoti Double DES algoritmu užkoduotą informaciją reikia tiek pastangų, kiek reikėtų dešifruoti DES šifrą, gautą naudojant 57 bitų raktą. Taigi Double DES algoritmas praktiškai nepadidino DES algoritmo saugumo.

Triple DES. Kadangi dviejų raktų naudojimas saugumo nepadidino, buvo sukurta dar viena DES algoritmo atmaina – Triple DES. Yra du Triple DES algoritmo variantai. Vienas iš jų naudoja tris raktus ir yra vertinamas kaip 112 bitų rakto algoritmo atitikmuo:

$$C = E(k_3, E(k_2, E(k_1, P)))$$

Čia *C* yra šifras, *P* – šifruojama informacija, *E* – DES algoritmas, k_1, k_2 ir k_3 – šifravimo raktai.

Kitas variantas naudoja du raktus ir nuo Double DES algoritmo skiriasi tuo, kad pirmiausiai informacija užkoduojama pirmuoju raktu, po to dešifruojama antruoju raktu ir vėl užšifruojama pirmuoju:

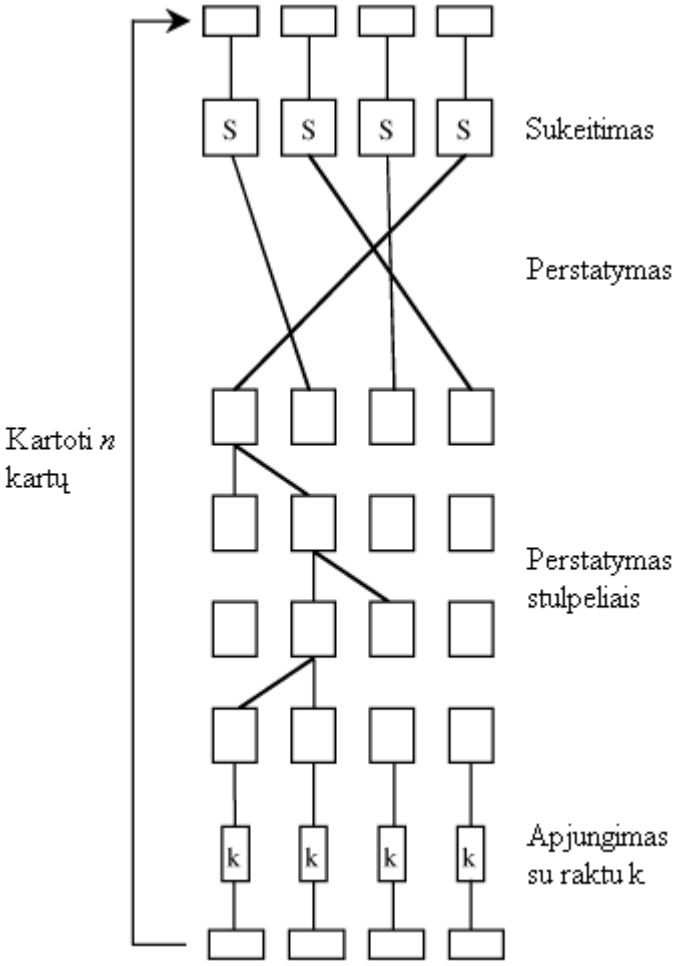
$$C = E(k_1, D(k_2, E(k_1, P)))$$

Šio algoritmo saugumas vertinamas kaip 80 bitų raktą naudojančio algoritmo atitikmuo.

1997 metais DES algoritmas buvo „nulaužtas“ – specialiai sukurta mašina, kurią sudarė 3500 lygiagrečiai veikiančių procesorių, surado DES algoritmu užkoduotos informacijos dešifravimo raktą per 4 mėnesius. 1998 metais raktui surasti prireikė 4 dienų.

2.6.5 Advanced Encryption Standard (AES, Rijndael)

AES algoritmas pradėtas kurti 1999 metais ir pradėtas naudoti 2001 metais. Naujo saugaus šifravimo algoritmo poreikis atsirado „nulaužus“ DES algoritmą. AES algoritmo kūrėjai yra danų kriptografai Vincent Rijmen ir Joan Daemen. AES algoritmas kaip ir DES pakartotinai atlieka tiek sukeitimo, tiek perstatymo algoritmus (paveikslėlis 11):

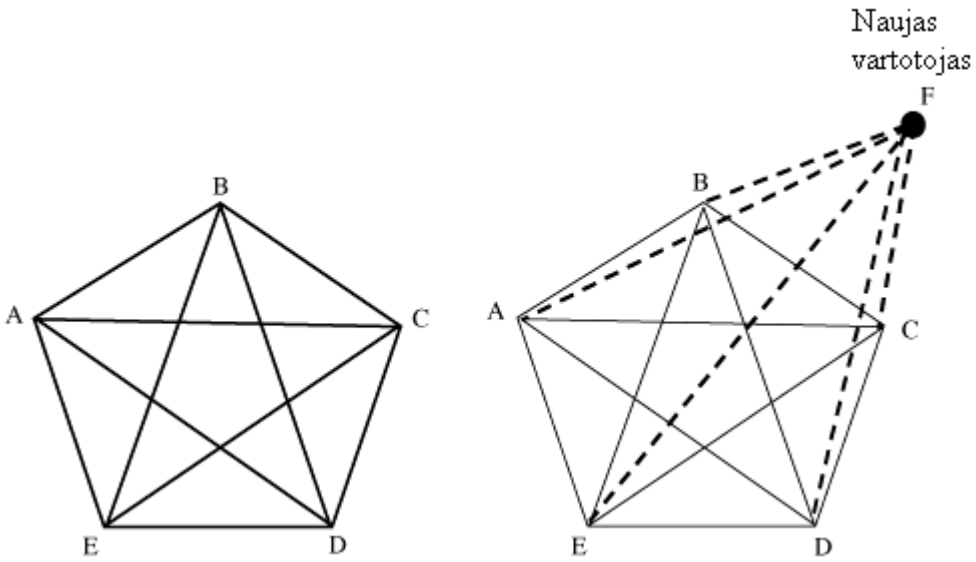


Paveikslėlis 11. AES algoritmas.

Šifravimo proceso metu naudojamų pakartojimų skaičius nėra fiksuotas ir gali būti padidintas prireikus (DES – 16 ciklų). AES algoritmo naudojamas raktas yra minimalaus 128 bitų ilgio. Rakto dydis nėra fiksuotas, todėl algoritmas gali būti saugus dar ilgą laiką. Šiuo metu naudojami ir 2048 bitų raktai.

2.6.6 Viešo rakto algoritmas RSA (Rivest-Shamir-Adelman)

Kodėl reikalingi vieši raktai? Slapto rakto problema yra ta, kad, norint užtikrinti slaptą bendravimą tarp vartotojų, kiekvienai porai reikia turėti atskirą raktą. Pvz., jeigu jūs norite slapta bendrauti su 4 asmenimis, jums reikės įsiminti/saugoti 4 skirtingus raktus. N asmenų grupei tarpusavyje slapta bendrauti reikės $n \times (n-1)/2$ raktų (paveikslėlyje 12 pavaizduota, kiek raktų reikia 5 asmenų bendravimui, kiek naujų raktų reikės atsiradus šeštam vartotojui).



Paveikslėlis 12. Raktų skaičiaus augimas.

Viešo rakto algoritmai pagrįsti tuo, kad kiekvienam asmeniui pakanka turėti du raktus: vieną viešą (public key), kurį galima platinti laisvai, ir kitą slaptą (private key). Šifravimo algoritmas ypatingas tuo, kad informacija, užšifruota vienu raktu, gali būti dešifruota tik naudojant kitą raktą:

$$P = D(k_{priv}, E(k_{pub}, P))$$

IR

$$P = D(k_{pub}, E(k_{priv}, P))$$

Čia P yra šifruojama informacija, E – šifravimo algoritmas, D – dešifravimo algoritmas, k_{priv} – privatus raktas ir k_{pub} – viešas raktas.

Taigi vartotojui A (žr. paveikslėlį 12) vartotojai B, C, D,E gali siųsti pranešimus, užkoduotus vartotojo A viešu raktu. Pranešimą dešifruoti galės tik A, nes tik jis žino antrąjį – slaptąjį – raktą.

RSA sistema buvo sukurta 1978 metais ir iki dabar nebuvo rasta jokių rimtų algoritmo spragų, todėl pasitikėjimas algoritmo saugumu tik auga. Viešo rakto sistemos vidutiniškai yra 10000 kartų lėtesnės už slapto rakto sistemas.

2.6.7 Maišos algoritmai (hash, checksum, message digest)

Maišos funkcijų paskirtis – užtikrinti, kad gauta informacija nebuvo pakeista. Maišos funkcijos rezultatas priklauso nuo koduojamos informacijos turinio. Pakeitus turinį, keičiasi ir maišos funkcijos rezultatas (kontrolinė suma). Jeigu gautos informacijos kontrolinė suma

nesutampa su lauktąja, vadinasi, informacija buvo pakeista. Maišos algoritmai yra vienkrypčiai, t.y. surasti kontrolinę informacijos sumą yra nesunku, tačiau nustatyti, kokia buvo informacija, praktiškai neįmanoma. Pvz., suskaičiuoti x^3 yra nesunku, tačiau surasti $\sqrt[3]{x}$ yra sudėtinga.

Plačiai žinomi ir naudojami maišos algoritmai yra MD4, MD5 ir SHA (SHS). Kripto analizės metu nustatyta, kad surasti informacijai, kurios kontrolinė suma būtų lygi užduotajai, vidutiniškai reikia 2^{63} žingsnių. Tai nereiškia, kad maišos algoritmai nėra saugūs, tiesiog tai įmanoma padaryti turint pakankamai techninių ir laiko resursų.

2.6.8 Šifravimo algoritmų vertinimo kriterijai

1949 metais Claude Shannon pasiūlė keletą charakteristikų, apibūdinančių gerą šifravimo algoritmą:

Informacijai užšifruoti reikalingas resursų kiekis turi būti tiesiogiai proporcingas reikalingam saugumui.

- Slaptam raktui ir šifruojamai informacijai neturi būti taikomi jokie apribojimai. Pvz., reikalavimas, kad šifruojamame tekste būtų vienodai raidžių A ir E, algoritmą paverstų beverčiu.
- Raktų sudarymo ir šifravimo algoritmai turi būti nesudėtingi (algoritmas su daugybe sudėtingų taisyklių gali būti panaudotas neteisingai arba pamirštas).
- Šifravimo metu suklydus tam tikroje vietoje neturi būti prarasta visa tolesnė informacija.
- Užšifruoto pranešimo dydis neturi būti didesnis už originalaus pranešimo dydį.

Šifravimo sistema laikoma patikima, jeigu ji:

- Pagrįsta matematika,
- Išanalizuota kompetentingų ekspertų ir paskelbta kaip saugi,
- Yra patikrinta laiko.

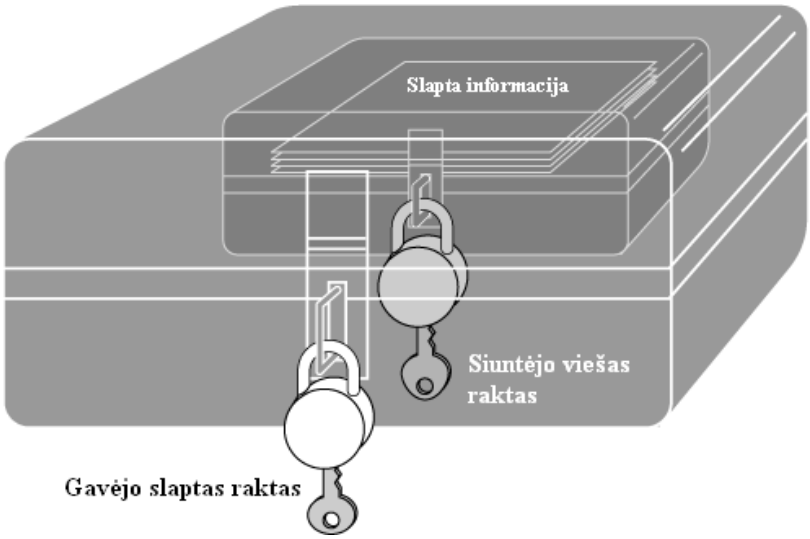
Šiuo metu plačiai naudojamos šifravimo sistemos yra šios:

- DES (Data Encryption Standard)
- RSA (Rivest, Shamir, Adelman – kūrėjų vardai)
- AES (Advanced Encryption Standard)

2.6.9 Raktų apsikeitimo problema

Kad pradėti saugų bendravimą (duomenų perdavimą) tarp dviejų kompiuterių, abi pusės turi naudoti tą patį šifravimo raktą. Viena iš bendraujančių pusių turi sugeneruoti ir perduoti raktą kitai pusei. Raktų apsikeitimui naudojama viešo rakto sistema. Laikykime, kad tiek siuntėjas, tiek gavėjas turi po du raktus: k_{priv-S}, k_{pub-S} (siuntėjo raktai, privatus ir viešas) ir k_{priv-G}, k_{pub-G} (gavėjo raktai, privatus ir viešas). Lengviausias būdas užmegzti saugų ryšį yra siuntėjui pasirinkti slaptą raktą K, užkoduoti jį savo privačiu raktu ir perduoti gavėjui: $E(k_{priv-S}, K)$. Tada gavėjas, pasinaudodamas viešuoju siuntėjo raktu, galėtų dešifruoti pranešimą ir perskaityti raktą K. Visgi šis sprendimas yra netinkamas, nes bet kas, perėmęs pranešimą, galėtų jį dešifruoti pasinaudodamas viešai skelbiamu siuntėjo raktu. Siuntėjas galėtų užkoduoti raktą pasinaudodamas gavėjo viešu raktu $E(k_{pub-G}, K)$, tada tik gavėjas galėtų dešifruoti pranešimą, tačiau jis negalėtų būti tikras, jog pranešimas tikrai atkeliavo iš konkretaus siuntėjo. Taigi rakto apsikeitimo problema yra sprendžiama taip: siuntėjas parenka slaptą raktą ir užkoduoja jį savo privačiu raktu, gautas šifras dar kartą užkoduojamas gavėjo viešu raktu ir tada išsiunčiamas. Pranešimą dešifruoti galės tik gavėjas, pasinaudodamas savo privačiu raktu. Gautam šifrui dešifruoti reikės siuntėjo viešojo rakto. Jeigu informacija sėkmingai dešifruojama, vadinasi, ji buvo užkoduota siuntėjo privačiu raktu, kas rodo, jog pranešimas tikrai atėjo iš konkretaus siuntėjo:

$$E(k_{pub-G}, E(k_{priv-S}, K))$$



Paveikslėlis 13. Apsikeitimas slaptu raktu.

3 NUOTOLINIO MOKYMOSI IR ŽINIŲ TESTAVIMO SISTEMA „COOL TEST TOOL“

Nuotolinio mokymosi ir žinių testavimo sistema „Cool Test Tool“ (CTT) buvo sukurta magistro studijų metu, 2008-aisiais metais.

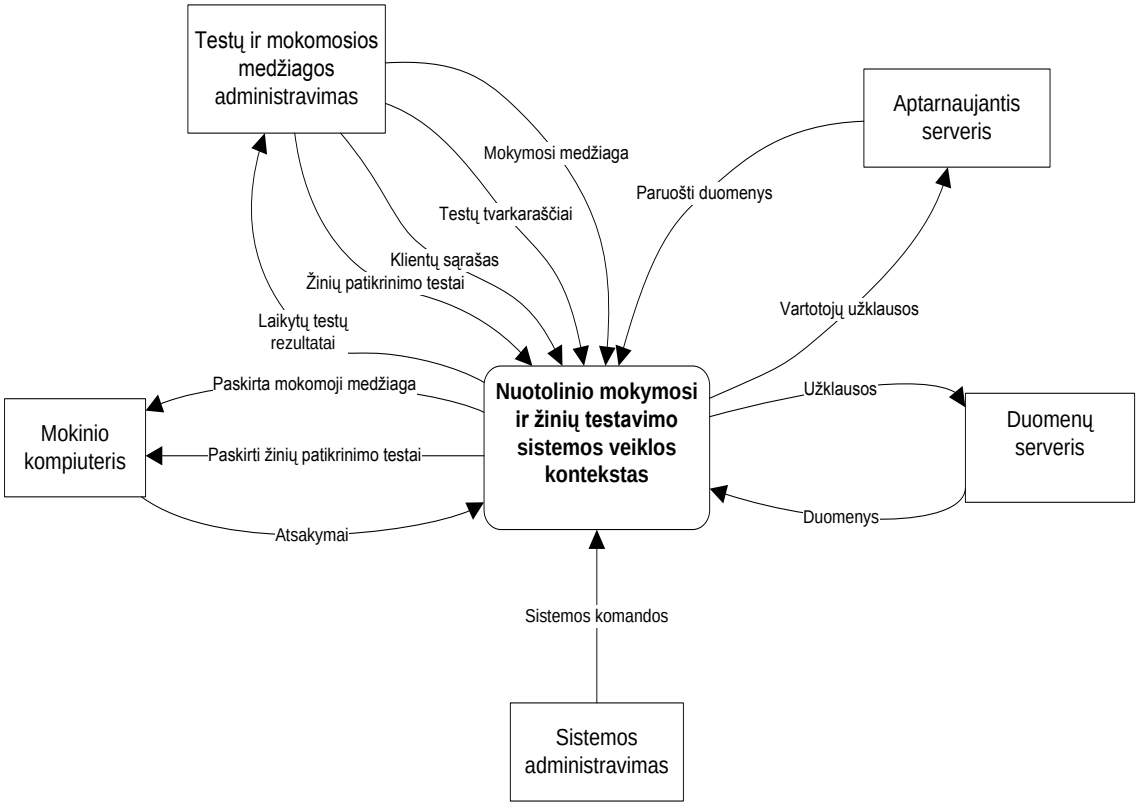
Sistema sukurta tam, kad automatizuoti ir palengvinti žmogaus žinių testavimo darbą. Programa labiausiai tinkama mokymo įstaigoms, pvz., mokykloms, universitetams, taip pat didelėms įmonėms, kurioms svarbu nuolat kelti ir tikrinti darbuotojų kvalifikaciją. Sistema puikiai tinka darbuotojų paieška bei atranka užsiimančioms įstaigoms.

Pagrindinės CTT sistemos funkcijos:

- Mokymosi medžiagos saugojimas ir pateikimas,
- Žinių patikrinimo testų kūrimas ir administravimas,
- Sukurtų testų vykdymas ir žinių vertinimas,
- Mokymosi rezultatų vertinimas.

Sistema suprojektuota naudojantis objektinio projektavimo metodologija ir vieninga modeliavimo kalba UML (Unified Modelling Language). Sistema sumodeliuota, aprašyta ir dokumentuota naudojantis UML CASE įrankiais.

3.1 Sistemos veiklos kontekstas



Paveikslėlis 14. CTT sistemos veiklos kontekstas.

3.2 Sistemos ribos ir funkcijos

Priklausomai nuo vartotojo tipo, sistema atlieka skirtingas funkcijas.

Testų vykdytojo funkcijos. Testų vykdytoju gali būti studentas, moksleivis ar bet koks kitas asmuo, kurio žinios yra tikrinamos arba kuris naudojasi serveryje patalpinta mokymosi medžiaga.

- Peržiūrėti mokymosi medžiagą. Prisijungęs vartotojas gali peržiūrėti mokymosi medžiagą, atsisiųsti reikiamą informaciją įvairių bylų pavidale ar tiesiog naudotis informacija (skaityti, žiūrėti, klausytis).
- Bendrauti su kitais vartotojais. Prisijungęs vartotojas gali dalyvauti diskusijose, užduoti klausimus, atsakyti į juos ir pan.
- Vykdyti testą. Vartotojas atsakinėja į užduodamus klausimus pasirinkdamas vieną ar kelis teisingus atsakymus arba įvesdamas reikiamą frazę. Testo vykdymui yra skirtas laikas, per kurį vartotojas turi spėti atsakyti į klausimus. Testo pabaigoje parodomas testo įvertinimas.
- Keisti savo duomenis. Vartotojas gali keisti savo prisijungimo prie sistemos duomenis: prisijungimo vardą ir slaptažodį.

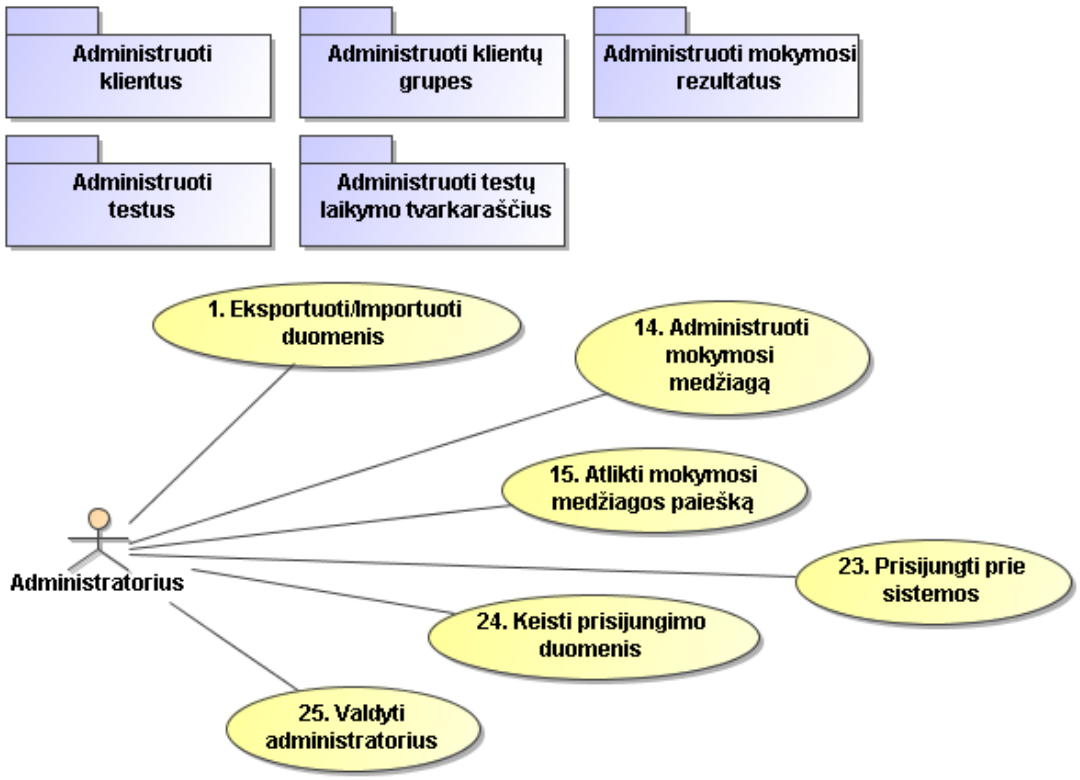
Testų administratoriaus funkcijos. Testų administratoriumi laikomas asmuo, administruojantis testus. Jis gali kurti, redaguoti testus; priskirti juos tam tikroms vartotojų grupėms laikymui; kurti, talpinti bei administruoti mokymosi medžiagą.

- Kurti testą. Testų administratorius sukuria klausimus, atsakymus, pažymi teisingus variantus. Sukurtą testą priskiria tam tikrai vartotojų grupei ar atskirai konkretiems vartotojams, kurie laikys testą.
- Redaguoti testą. Galimybė pakeisti jau sukurto testo duomenis: klausimus, atsakymus, testo parametrus.
- Peržiūrėti laikytus testus. Galimybė peržiūrėti informaciją, kaip vartotojas laikė testą, t.y. pateiktų klausimų sąrašą, pasirinktus atsakymų variantus, sistemos apskaičiuotus įvertinimus.
- Administruoti testų laikymo leidimus. Galimybė konkrečiam testo vykdytojui arba jų grupei suteikti/atimti teisę laikyti testą(-us). Taip pat galimybė nurodyti/keisti tam tikrus testo laikymo parametrus: laikotarpį, galimų laikymų skaičių.
- Keisti laikytų testų įvertinimus.

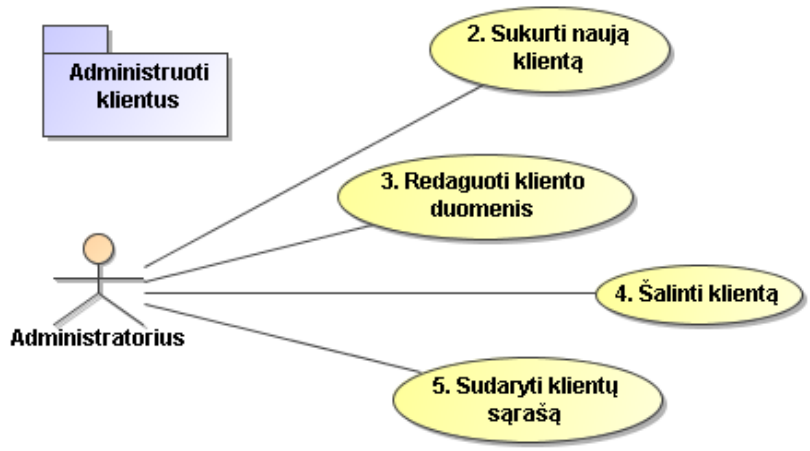
- Administruoti mokymosi medžiagą. Testų administratorius gali kurti/redaguoti mokymosi medžiagą, ją talpinti, valdyti prieinamumo teises.

Serverio administratoriaus funkcijos. Serverio administratoriumi laikomas asmuo, atsakingas už serverio programų darbą. Jis taip pat gali atlikti visus testų administratoriaus veiksmus, administruoti sistemos vartotojus: testų vykdytojus, testų administratorius. Šis vartotojas yra atsakingas už prisijungimo duomenų prie duomenų bazės ir ftp serverio administravimą, atsarginių duomenų kopijų darymą ir pan.

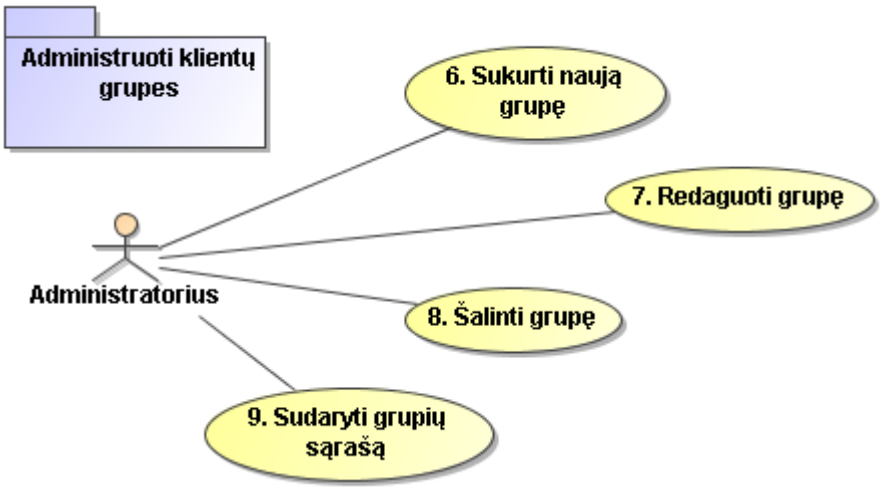
- Sistemos vartotojų administravimas. Galimybė kurti, redaguoti bei šalinti sistemos vartotojus.
- Serverio programų administravimas. Vartotojus aptarnaujančių Windows servisų valdymas: paleidimas, sustabdymas, šalinimas. Prisijungimo duomenų prie ftp serverio ir duomenų bazės valdymas. Atsarginių duomenų kopijų darymas.



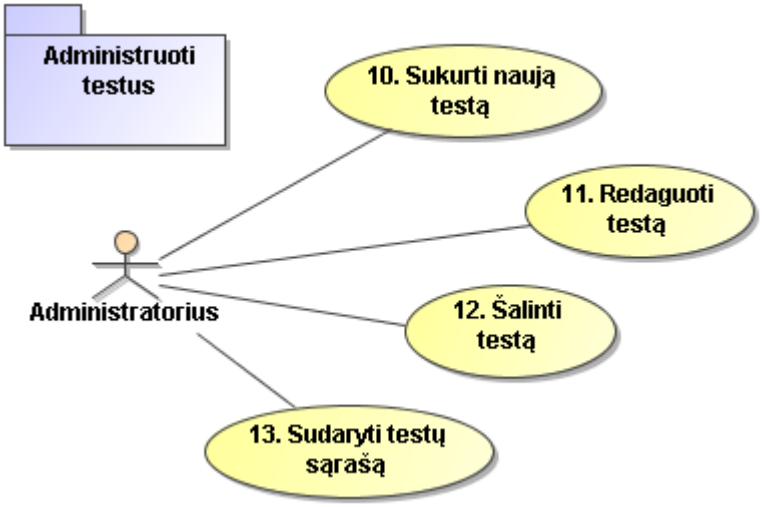
Paveikslėlis 15. CTT sistemos panaudojimo atvejai.



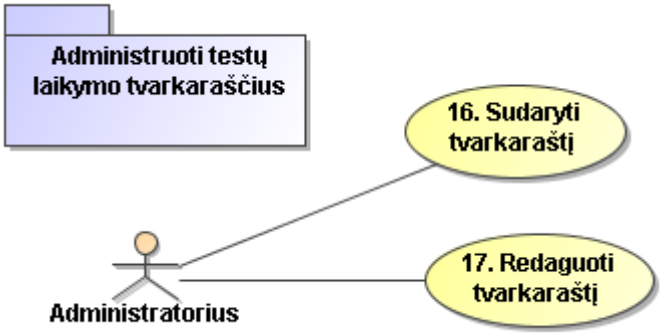
Paveikslėlis 16. CTT sistemos panaudojimo atvejai.



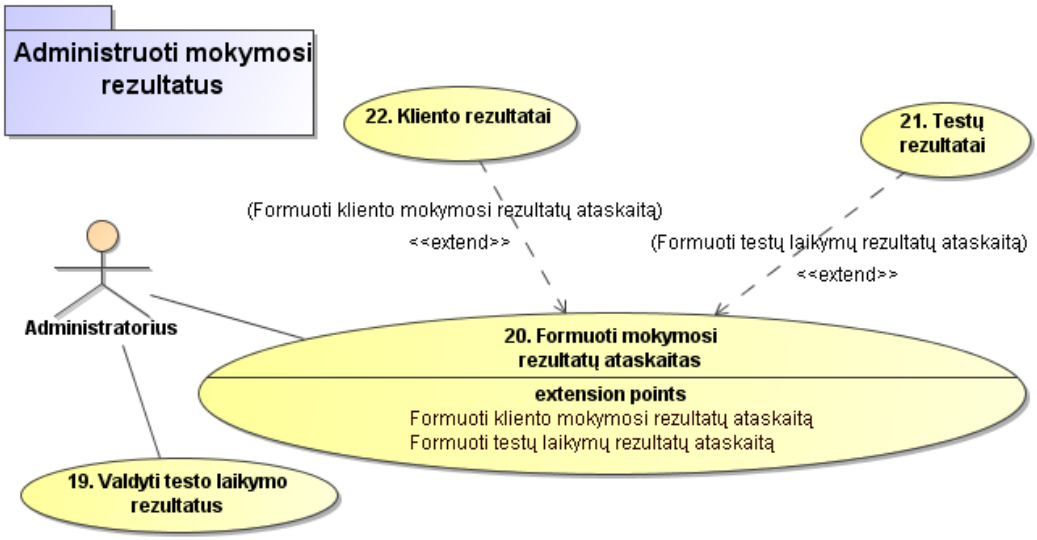
Paveikslėlis 17. CTT sistemos panaudojimo atvejai.



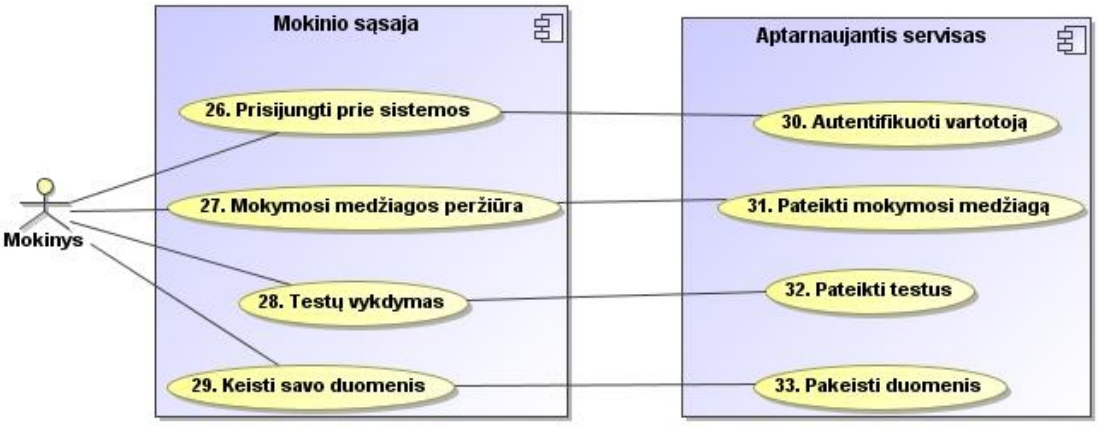
Paveikslėlis 18. CTT sistemos panaudojimo atvejai.



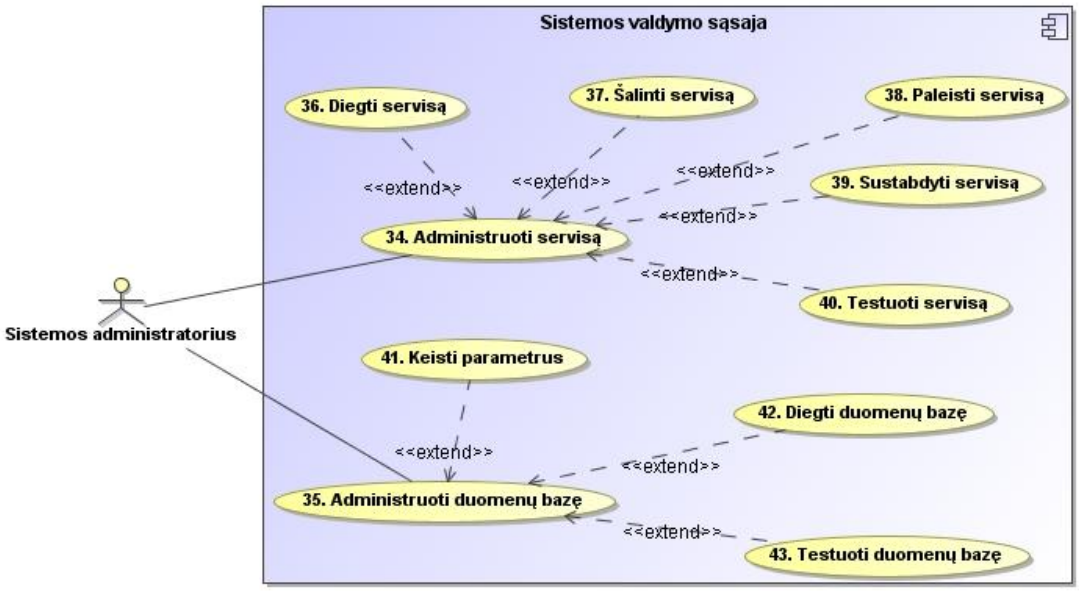
Paveikslėlis 19. CTT sistemos panaudojimo atvejai.



Paveikslėlis 20. CTT sistemos panaudojimo atvejai.

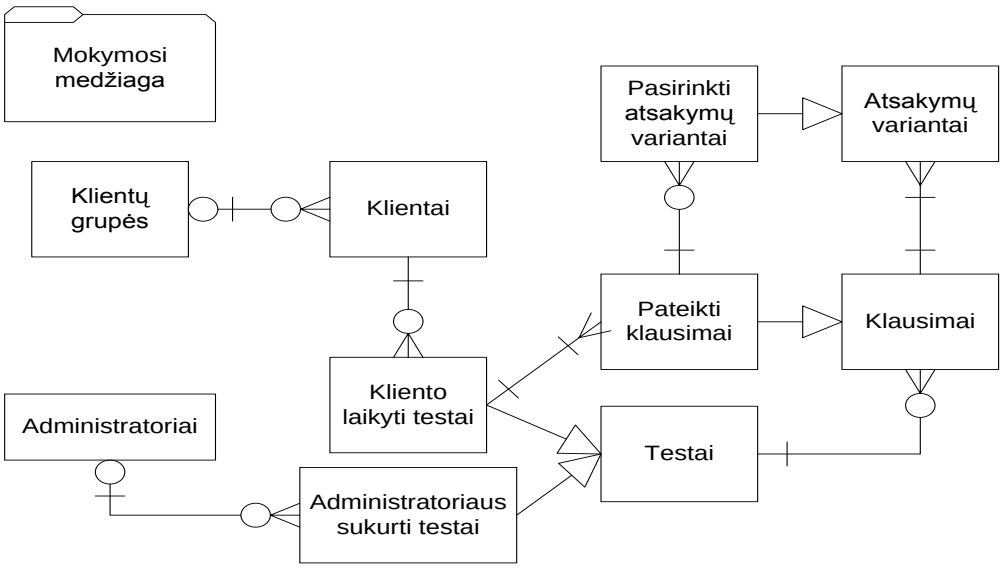


Paveikslėlis 21. CTT sistemos panaudojimo atvejai.



Paveikslėlis 22. CTT sistemos panaudojimo atvejai.

3.3 Sistemos duomenų modelis



Paveikslėlis 23. CTT sistemos duomenų modelis.

3.4 Sistemos architektūra

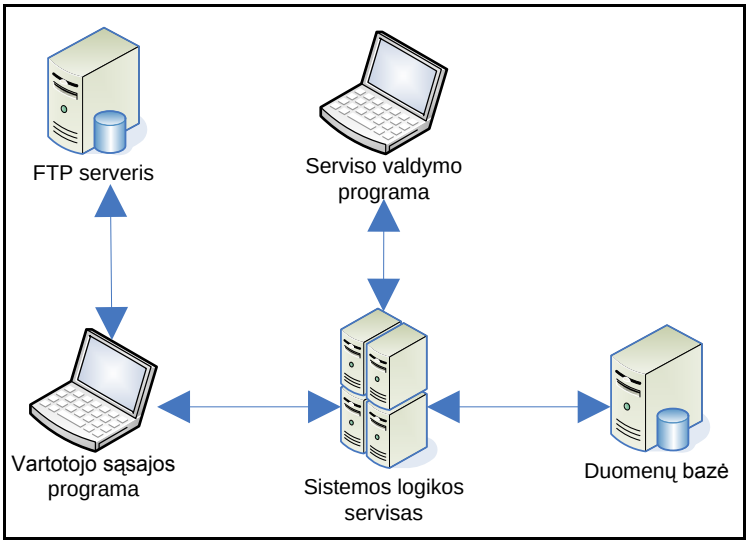
Sistemą sudaro penkios dalys: testų vykdytojo programa, testų administratoriaus programa, dvi serveryje veikiančios ir vartotojų sąsajos programas aptarnaujančios programos (Windows servisai), serverio programas valdanti administratoriaus programa.

Kliento (testų vykdytojo) programa suteikia galimybę naudotis daugialype mokymosi medžiaga (garsinė bei vaizdinė medžiaga, paveikslėliai ir kt.), leidžia vartotojams bendrauti tarpusavyje, dalintis informacija, atlikti žinių patikrinimo testus, matyti mokymosi rezultatus

bei testų laikymo tvarkaraščius. Administratoriaus programa leidžia kurti daugialypę mokymosi medžiagą bei žinių patikrinimo testus, valdyti kliento programos vartotojus, matyti bei koreguoti jų mokymosi rezultatus, skirti testus, sudaryti jų laikymo tvarkaraščius.

Programinė įranga buvo kuriama kliento ir serverio architektūros principu. Kliento ir serverio architektūra suteikia sistemai daug lankstumo ją atnaujinant, perkeliant į naujas platformas, plečiant galimybių ir atliekamų funkcijų sąrašą. Sistemos duomenų valdymo perdavimas atskirai sukurtiems logikos servisams leidžia keisti duomenų struktūrą nereikalaujant kliento programų pakeitimų. Taip pat atsiranda galimybė lengvai prisitaikyti prie naujų duomenų bazių technologijų. Atskyrus tris programinės įrangos esybes (sąsajos/atvaizdavimo, logikos, duomenų), atsiranda galimybė lygiagrečiai kurti/modifikuoti atskirų tarpinių serverių programinę įrangą. Reikėtų pastebėti, kad kvalifikacija, reikalinga kurti vienam kompiuteriui skirtas programas ir kliento/serverio architektūros programas, labai skiriasi. Skirtingi specialistai gali susikoncentruoti išskirtose trijose srityse ir taip padidinti bendrą galutinio produkto kokybę.

Kliento ir serverio architektūra taip pat suteikia galimybę lanksčiau paskirstyti resursus. Tarpiniai serveriai gali būti lengvai perkeliami ir atnaujinami. Kadangi kliento programos skirtos tik atvaizdavimui, reikalavimai kliento kompiuterių atminčiai bei reikalingai kietojo disko vietai sumažėja. Pasirinkta architektūra leidžia lanksčiau reaguoti į veiklos taisyklių pasikeitimus, suteikia galimybę paprasčiau palaikyti ir taisyti programų kodą, lengviau migruoti į naujas serverių platformas. Pagrindiniai kliento ir serverio architektūros trūkumai ir problemos: daugiau programinio kodo skirtingose vietose padidina sistemos klaidos tikimybę; reikalinga papildomai spręsti tinklo valdymo, serverių resursų paskirstymo problemas.



Paveikslėlis 24. CTT sistemos išdėstymo vaizdas.

4 SAUGUMO VALDYMO PLANAS IR RIZIKŲ VERTINIMAS

CTT sistemos realizavimo laikotarpiu buvo susidurta su organizacinėmis problemomis. Nors sistemą realizavimo tik du asmenys, buvo susidurta su darbų organizavimo, derinimo problemomis. Programinės įrangos kūrimo principai, stilius, kokybės valdymo priemonės nebuvo bendrai sutarti. Taigi CTT sistemą sudarančios atskiros programos neturi vieningų priemonių priežiūrai, palaikymui, o dėl to sumažėjo visos CTT sistemos kaip vieno produkto kokybė. Organizacinės priemonės kompiuterinių sistemų saugumui valdyti yra šio magistro darbo tyrimo objektas.

Realizavus sistemą ir pristačius ją potencialiems pirkėjams, taip pat buvo susidurta su pasitikėjimo sistema problema. Kadangi sistema dar nėra naudojama praktikoje, potencialūs jos vartotojai negali įvertinti realiai teikiamos sistemos naudos. Taip pat yra bijoma, jog sistemoje gali būti realizuotos slaptos funkcijos / metodai (angl. trapdoor, backdoor), kuriais naudojantis testus atliekantys asmenys galėtų sukčiauti. Piktavalio, slapto kodo realizavimo problema taip pat nagrinėjama šioje dalyje.

Saugumas yra techninių, administracinių ir fizinių priemonių rinkinys. Iki šiol šiame darbe buvo nagrinėjamos techninės priemonės programinės įrangos saugumui užtikrinti. Visgi vien tik techninių priemonių nepakanka. Pvz., kokią naudą gali duoti viešojo rakto sistemos naudojimas, jeigu bet kuris asmuo gali lengvai sužinoti privatų raktą? Kokia nauda iš duomenų prieigos kontrolės sistemos, jeigu įmonės darbuotojas, turintis teisę skaityti slaptus duomenis, perduoda juos konkuruojantiems subjektams? Administracinė ir fizinė kontrolė nėra tokia sudėtinga kaip techninės priemonės, tačiau gali būti ypatingai svarbi užtikrinant saugumą.

4.1 Saugumo planavimas

Dauguma paprastų vartotojų nežino arba ignoruoja paprastus dalykus, kurie gali labai komplikuoti jų pačių arba jų atstovaujamų subjektų saugumą. Pvz., visi žinome, kad kompaktinis diskas gali talpinti daugybę slaptų dokumentų, tačiau jame saugoma informacija nėra aiškiai matoma kaip kad atspausdinto dokumento atveju, todėl daugelis pasirūpina spausdinto dokumento saugumu, bet pamiršta pasirūpinti kompaktinio disko saugumu ir šis gali būti lengvai pasisavintas trečiųjų asmenų. Taigi kiekviena organizacija, naudojanti kompiuterius vertingai informacijai saugoti, turi kruopščiai ir efektyviai planuoti saugumą.

Saugumo planas – tai dokumentas, kuris apibūdina, kaip organizacija valdys su saugumu susijusias problemas. Saugumo planas turi būti peržiūrėtas pasikeitus saugumo poreikiams.

Saugumo plano struktūra. Saugumo planas identifikuoja ir organizuoja su saugumu susijusias veikas. Tai esamos būsenos apibūdinimas ir ją tobulinančių priemonių planas. Kiekvienas saugumo planas turėtų turėti šiuos punktus:

- Saugumo politika. Saugumo tikslai ir susijusių asmenų pasiryžimas siekti užsibrėžtų tikslų.
- Dabartinė būsena. Saugumo situacija plano sudarymo metu.
- Reikalavimai, kaip užtikrinti, kad bus pasiekti saugumo politikoje užsibrėžti tikslai.
- Priemonės, kurių pagalba siekiama įgyvendinti apibrėžtus reikalavimus.
- Atsakingumas. Sąrašas asmenų, subjektų, atsakingų už atitinkamų veiksmų vykdymą, sąlygų laikymąsi ir pan.
- Tvarkaraštis, numatantis, kada imtis tam tikrų saugumo funkcijų.
- Nuolatinė peržiūra, apibūdinanti, kaip dažnai reikia peržiūrėti, keisti sudarytą saugumo planą.

Saugumo politika – tai gana abstrakti saugumo tikslų formuluotė. Gali pasirodyti, kad visos formuluotės turėtų būti vienodos – užkirsti kelią saugumo spragoms. Visgi saugumo politika yra viena sunkiausiai aprašomų skilčių saugumo plane. Saugumo politika nusako kompromisą tarp saugumo lygio, kaštų, nepatogumo vartotojams ir kt. Pvz., turime nuspręsti, ar norime užtikrinti aukščiausią saugumo lygį, kuriam pasiekti greičiausiai tektų naudoti nepopuliarias priemones, ar norime sumažinti galimų saugumo spragų sukeltus padarinius. Saugumo politika turi atsakyti į šiuos esminius klausimus:

- Kam suteikiama prieiga prie slaptos informacijos,
- Kokia informacija laikoma slapta,
- Kokios prieigos teisės (skaitymo, rašymo, atnaujinimo ir pan.) suteikiamos konkretiems vartotojams konkretiems resursams.

Organizacijos tikslais galėtų būti apsauga nuo informacijos nutekėjimo, duomenų apsauga katastrofų atvejais, verslo funkcionalumo išsaugojimas sugedus kompiuterinėms sistemoms ir pan. Svarbu paminėti, kas rūpinasi saugumu: ar specialus įmonės dalinys, ar kiekvienas darbuotojas, ar darbuotojų grupės vadovas, ar kt.

Dabartinė saugumo būsena apibūdina jau žinomas su saugumu susijusias problemas, darbo aplinką, galimas saugumo spragas. Būseną gali apibūdinti sąrašas įmonės vertybių/turto kartu su jiems galimomis grėsmėmis ir šiuo metu naudojamomis priemonėmis grėsmėms išvengti ar jų žalai sušvelninti. Kiekvienam sąraše esančiam elementui gali būti nurodytas už jo saugumą atsakingas subjektas. Saugumo plane taip pat turėtų būti informacija, kaip elgtis atradus naujas grėsmes saugumui.

Reikalavimai. Esminė saugumo plano dalis – sąrašas reikalavimų, kurie užtikrintų norimą sistemos saugumo lygį. Reikalavimai paprastai kyla iš organizacijos poreikių. Tokiu poreikiu gali būti reikalavimas atitikti specifinius trečiųjų šalių reikalavimus, pvz., valstybinėms ar komercinėms įstaigoms taikomi specialūs duomenų apsaugos įstatymai. Pfleeger [PFL91] pabrėžia, kad labai svarbu atskirti reikalavimus nuo priemonių. Reikalavimu turi būti nusakomas bendras konkretus poreikis, o priemonės nurodo, kaip tą poreikį patenkinti. Pvz., organizacija, siekianti saugumo politikoje nurodytos apsaugos nuo informacijos nutekėjimo, gali kelti reikalavimą identifikuoti asmenis, besinaudojančius slapta informacija. Priemonių sąrašas turi nurodyti galimus reikalavimo išpildymo būdus: pvz., specialios programinės įrangos naudojimas, pirštų atspaudų skaitymo įrengimo naudojimas, magnetinių tapatybės kortelių naudojimas ir pan. Žemiau esančioje lentelėje (lentelė 3) kaip pavyzdys pateikiamas teisingai ir neteisingai suformuluotų reikalavimų sąrašas.

Lentelė 3. Korektiški ir nekorektiški reikalavimai saugumo tikslams pasiekti.

1	Slapta įmonės informacija bei privatūs klientų duomenys turi būti pilnai apsaugoti nuo nutekėjimo tretiesiems asmenims.
2	Kiekvienas dokumentas turi būti pažymėtas etikete, kurios spalva nurodo dokumento slaptumo lygį.
3	Turi būti numatyti būdai atskirti, kuri informacija yra slapta, o kuri ne.
4	Priemonės, realizuojančios saugumo plane numatytus reikalavimus, turi būti apsaugotos nuo nesankcionuoto pakeitimo.
5	Sistema turi saugoti veiksmų, turinčių įtakos sistemos saugumui, sąrašą. Šie veiksmai yra: naujų sistemos vartotojų sukūrimas, informacijos slaptumo lygio keitimas, nesėkmingi bandymai prisijungti prie sistemos.

Pirmasis „reikalavimas“ iš tiesų yra saugumo politika. Antrasis „reikalavimas“ apibūdina priemones, realizuojančias reikalavimą nustatyti, kuri informacija yra slapta. Taigi antrasis „reikalavimas“ iš tiesų yra priemonė. Trečias, ketvirtas ir penktas reikalavimai išties yra

korektiški reikalavimai, nes nurodo poreikį pasiekti konkrečią sistemos charakteristiką, ypatybę ir nenurodo (neapriboja) reikalavimo įgyvendinimo būdų.

Apibendrinant, reikalavimai turi nurodyti, kas turi būti padaryta, o ne kaip.

Atsakingumas. Šioje saugumo plano skiltyje turi būti nurodyta, kas yra atsakingas už numatytų saugumo priemonių įgyvendinimą. Pvz., asmeninio kompiuterio saugumu gali rūpintis pats darbuotojas arba specialiai tam paskirtas darbuotojas. Vadybininkai gali būti atsakingi už priežiūrą, kaip kiti darbuotojai laikosi saugumo reikalavimų; duomenų bazių administratorius paprastai atsakingas už duomenų prieigos teisių valdymą; personalo skyrius gali būti atsakingas už buvusių klientų privačių duomenų sunaikinimą ir pan.

Tvarkaraštis. Saugumo planas negali būti įgyvendintas akimirksniu. Tvarkaraštis reikalingas tam, kad numatyti, kokiais etapais planas bus vykdomas. Tvarkaraštyje taip pat turėtų atsispindėti priemonių įgyvendinimo tvarka, kad jautriausios saugumui problemos būtų išspręstos pirmiausiai. Tvarkaraštis taip pat tarnauja kaip gairės, pagal kurias galima spręsti, kaip vykdomas saugumo planas.

Nuolatinė peržiūra. Be saugumo reikalavimų ir priemonių apibrėžimo mes taip pat turime galėti įvertinti sistemos saugumą, kad galėtumėme įsitikinti, jog sistema yra saugi. Todėl saugumo plane turi būti numatyta, kaip dažnai, kokioms situacijoms esant, planas turi būti peržiūrėtas. Besikeičiant vartotojams, techninei įrangai gali atsirasti naujų grėsmių. Be to, šiuo metu naudojamos priemonės saugumui užtikrinti gali pasenti, pasidaryti neefektyviomis, pvz., pasirodžius greitesniems procesoriams, naudojamas šifravimo algoritmas gali būti „nulaužtas“. Plano peržiūrėjimo datos gali būti nustatytos atsižvelgiant į kalendorinį laiką, pvz., kas devynis mėnesius, arba esant tam tikroms situacijoms, pvz., išleidus naują programinės įrangos versiją.

Saugumo plano sudarytojų komanda. Kas atlieka saugumo analizę, rekomenduoja problemų sprendimo būdus, sudaro saugumo planą? Organizacijoje šiuos veiksmus turėtų atlikti specialiai suburtas komitetas ar darbuotojų grupė. Grupės narių skaičius priklauso nuo saugomų sistemų sudėtingumo ir siekiamo saugumo lygio. Visgi grupę turėtų sudaryti nuo penkių iki devynių asmenų. Komandos nariai turėtų atstovauti šias vartotojų grupes:

- Techninės įrangos specialistas,
- Sistemos administratorius,

- Sistemos kūrėjas,
- Duomenų bazės administratorius,
- Fizinės apsaugos specialistas,
- Galutinis vartotojas.

Kai kuriais atvejais vienas asmuo gali atstovauti kelias grupes. Kai kurias grupes gali atstovauti specialiai samdomas asmuo ir t.t.

Sudarytas saugumo planas turi būti patvirtintas ir visi susiję asmenys turi sutikti jo laikytis. Planas, kuris tiesiog guli lentynoje ir renka dulkes, bus neveiksmingas.

4.2 Nenutrūkstamo verslo planas

Mažoms kompanijoms svarbu turėti nenutrūkstamo verslo planą. Labai dažnai, sutrikus kompiuterių darbui, piktavaliams vartotojams įsilaužus į kompiuterių sistemas ar sugadinus duomenis, sutrinka ir verslo procesai. Negalėjimas pasiekti svarbių duomenų gali reikšti negalėjimą aptarnauti klientų, kas veda prie verslo stojimo ir pelno nebuvimo. Nenutrūkstamo verslo planas dokumentuoja, kaip bus vykdomi verslo procesai kompiuterinių sistemų incidentų atvejais. Saugumo planas sprendžia saugumo problemas įprasto darbo metu, o nenutrūkstamo verslo planas apibūdina situacijas, kada kompiuterinės sistemos yra kritiškai pažeidžiamos ir jų nebegalima naudoti, kada kritinių situacijų sukeltų padarinių likvidavimas yra ilgas procesas. Pvz., tokiomis situacijomis galėtų būti gaisras, kurio metu sugadinama techninė įranga, prarandamas interneto ryšio tinklas, klaida programinėje įrangoje neleidžia pasinaudoti ypač svarbiais duomenimis ir pan. Pagrindiniai žingsniai sudarant nenutrūkstamo verslo planą turėtų būti tokie:

- Nustatyti poveikį verslui krizės metu,
- Sukurti strategiją krizės padariniams kontroliuoti,
- Sukurti strategijos vykdymo planą.

Kad nustatyti krizės poveikį verslui, reikia atsakyti į du pagrindinius klausimus:

- Kas yra būtini, esminiai dalykai verslui funkcionuoti? Atsakymais į šį klausimą galėtų būti „tinklas“, „klientų duomenų bazė“ ir pan.
- Kas gali sužlugdyti, nutraukti esminių verslo procesų vykdymą? Svarbu nustatyti, kas gali atsitikti, o ne kaip tai gali atsitikti. Pvz., kompiuterinė įranga gali būti sugadinta, nesvarbu, ar gaisro, ar potvynio metu.

Strategija numato, kaip esminiai verslui dalykai gali būti apsaugoti ar pakeisti. Pvz., netekus elektros energijos ir praradus galimybę aptarnauti klientus internetu, gali būti naudojamas specialiai tokiems atvejams įsigytas atsarginis elektros generatorius arba klientai gali būti aptarnaujami telefonu. Kritinės situacijos metu naudojamos priemonės priklauso ir nuo krizės trukmės laiko. Taigi strategija apibūdina galimas krizes ir esminius sprendimus, kuriuos reikia vykdyti priklausomai nuo situacijos.

Planas numato, kas yra atsakingas krizės atveju, kokius konkrečius veiksmus reikia atlikti ir kas juos atlieka. Plane neturi būti tokių veiksmų kaip pagalbos tarnybų iškvietimas ar evakuacija. Plano sudarymo tikslas – numatyti, kaip išsaugoti funkcionuojantį verslą esant kritinėms situacijoms.

Pasibaigus krizei, reikia atlikti jos analizę ir pasistengti atsakyti į pagrindinius du klausimus:

- Ar reikia imtis papildomų priemonių saugumui užtikrinti? Pvz., įsibrovimo atveju reikėtų nustatyti jo priežastį. Jeigu įsibrovimas pavyko dėl sistemos atnaujinimų nebuvimo, reikėtų apsvarstyti procedūras, užtikrinančias naujausių pataisų įdiegimą. Jeigu įsilaužimas pavyko dėl prastai pasirinkto vartotojo slaptažodžio, gal reikėtų apmokyti vartotojus, kaip sudaryti saugius slaptažodžius. Bendru atveju turi būti apsvarstytos priemonės, galinčios padėti išvengti panašių problemų ateityje.
- Ar nenutrūkstamo verslo planas suveikė? Ar visi žinojo, ką reikia daryti, ar visiems buvo prieinami reikiami resursai planui vykdyti? Ką reikėtų daryti kitaip pasikartojus incidentui?

4.3 Rizikų analizė ir valdymas

Saugumo planavimas apima galimų rizikų analizę. Rizika vadiname potencialią problemą, kuri gali kilti sistemoje. Siūloma rizikos veiksnius apibūdinti trimis charakteristikomis:

- Nuostolis, kuris būtų patirtas tam tikro įvykio metu. Nuostolis gali būti apibūdintas kaip „sukompromituotas saugumas“, „sumažėjusi kokybė“, „prarastas turtas“, „prarasta kontrolė“ ir pan. Patiriami nuostoliai dar vadinami rizikos poveikiu (risk impact).

- Tikimybė, kad įvyks nuostolius atnešantis įvykis. Tikimybė vertinama skaičiumi nuo 0 (neįmanoma) iki 1 (neabejotina). Kada tikimybė yra lygi 1, laikome, kad turime problemą.
- Santykinis dydis, kuriuo galime sumažinti galimus nuostolius. Turime nustatyti, ką galima padaryti, jeigu tai apskritai įmanoma, kad išvengti numatytų nuostolių ar juos sumažinti.

Šiais trim punktais aprašius rizikos veiksnius, galima suskaičiuoti rizikos pasirodymo poveikį. Rizikos poveikį apskaičiuoti galime rizikos pasirodymo tikimybę padauginę iš galimų nuostolių vertės. Pvz., jeigu viruso atsiradimo sistemoje tikimybė yra 0,3 ir viruso padarytai žalai likviduoti reikėtų skirti 10000 litų, tada šios rizikos pasirodymo poveikis būtų 3000 litų. Atlikus tokius skaičiavimus galima nuspręsti, kad antivirusinės programos už 100 litų įdiegimas yra gera investicija, galinti apsaugoti nuo daug didesnių nuostolių. Kadangi rizikų pasirodymo tikimybės kinta laikui bėgant, rizikos analizė turi būti periodiškai kartojama.

Bendrai kalbant, rizikai valdyti galime naudoti vieną iš trijų pagrindinių strategijų:

- Išvengti rizikos. Pvz., galime pakeisti reikalavimus saugumui ar tam tikras sistemos charakteristikas, kad eliminuoti rizikos pasirodymo galimybę.
- Perleisti rizikų sukeltus nuostolius. Pvz., įmonė gali sudaryti sutartį su kita organizacija, kuri būtų atsakinga už saugumą. Taip pat galima apsidrausti nuo tam tikrų įvykių, kuriems įvykus nuostolius atlygintų draudimo įmonė.
- Priimti riziką susitaikant su galimais nuostoliais. Prisiėmus galimą riziką, svarbu numatyti galimų nuostolių dydį, pasiruošti valdyti rizikos sukeltus padarinius, numatyti būdus juos sumažinti.

Pagrindiniai žingsniai rizikos analizės metu:

- Identifikuoti saugomus, vertingus dalykus;
- Nustatyti pažeidžiamas vietas;
- Įvertinti tikimybę, kad bus pasinaudota nustatytomis pažeidžiamomis vietomis;
- Apskaičiuoti galimų metinių nuostolių vertę;
- Apžvelgti rizikos kontroliavimo priemones ir jų įsigijimo bei palaikymo kaštus;
- Apskaičiuoti, kiek tam tikrų priemonių naudojimas gali sumažinti nuostolių per metus.

Saugomų vertybių identifikavimas. Prieš nustatant pažeidžiamas sistemų dalis, turime nuspręsti, ką norime saugoti. Priklausomai nuo organizacijos saugoti galime:

- Techninę įrangą,
- Programinę įrangą,
- Duomenis,
- Žmones (jeigu sistemą projektuoja vienas asmuo, šis yra labai svarbus projekto sėkmei užtikrinti, jeigu sistemą projektuoti gali dešimt asmenų, tai kiekvienas projektuotojas atskirai nėra gyvybiškai svarbus, nes jį galima lengvai pakeisti),
- Dokumentaciją,
- Materialias atsargas.

RAND korporacijos [ANT01] sukurta pažeidžiamumų nustatymo ir jų mažinimo metodologija taip pat rekomenduoja į vertybių sąrašą įtraukti:

- Organizacijos infrastruktūrą;
- Pastatus ir transporto priemones, kuriuose veikia (yra laikomos) saugomos sistemos;
- Aplinkos sąlygas, reikalingas tinkamai veiklai;
- Žmogiškuosius išteklius, tokius kaip darbo procedūros ir metodai.

Pažeidžiamumų nustatymas. Antrasis žingsnis rizikos analizės metu – saugomų vertybių pažeidžiamų vietų nustatymas. Šio žingsnio metu reikia nuspėti, kokia žala gali būti padaryta saugomiems objektams, kas gali sukelti žalą. Pažeidžiamumu laikoma bet kuri situacija, galinti įtakoti objekto konfidencialumą, integralumą ar prieinamumą.

Rizikos pasirodymo tikimybės įvertinimas. Numatyti rizikos pasirodymo tikimybę yra labai sunku. Paprastai tikimybė skaičiuojama vienu iš keturių būdų:

- Klasikinis tikimybės skaičiavimo būdas. Tikimybės skaičiavimas šiuo būdu yra pats paprasčiausias ir pagrįstas teorija. Pvz., tikimybei, kad metus žaidimų kauliuką iškris tam tikras skaičius, apskaičiuoti nereikia jokių empirinių[8] duomenų. Tikimybei rasti pakanka žinoti patį objekto modelį. Klasikinis tikimybės skaičiavimo būdas netinka vertinant aplinkybes, kurių aibė nėra baigtinė. Be to, reikia turėti išbaigtą korektišką sistemos modelį.

[8] Empirinis – patirtinis, susijęs su patyrimu.

- Tikimybės skaičiavimas naudojantis statistika. Kai nėra galimybės naudoti klasikinį tikimybės skaičiavimo modelį, naudojamas statistika paremtas skaičiavimo būdas. Norėdami apskaičiuoti, kokia tikimybė, kad metus žaidimų kauliuką iškris norimas skaičius, mesime kauliuką keletą kartų ir fiksuosime iškritusį skaičių. Tikimybei apskaičiuoti naudosimės gautais statistiniais duomenimis. Šiuo būdu apskaičiuota tikimybė nėra tiksli, tačiau, vertinant sistemų rizikas, tikimės, kad ji yra artima teorinei.
- Subjektyvus tikimybės vertinimas. Vertinant rizikų pasirodymo tikimybes kompiuterinėje sistemoje pirmuose dviejuose punktuose aprašytais būdais, reikalaujama turėti jau sukurtą ir tam tikrą laiką veikiančią sistemą. Kadangi rizikos planas sudaromas ankstyvoje projekto vykdymo stadijoje, nei teorinių, nei statistinių duomenų nėra. Šiuo atveju pasikliaujama eksperto nuomone. Skirtingi ekspertai tikimybę gali įvertinti labai skirtingai. Labai svarbu, kad vertinantis ekspertas gerai pažintų sistemą, jos veikimo aplinką ir kt.
- Delphi metodas. Tai subjektyvus tikimybės vertinimas, pasiūlytas RAND korporacijos [HAL67]. Šiuo būdu tikimybę vertina keli ekspertai ir priima bendrą sprendimą. Pirmiausiai ekspertai supažindinami su sistema, jos architektūra, naudojimo aplinkybėmis ir kt. Po to kiekvienas ekspertas atskirai įvertina galimą tam tikro įvykio tikimybę. Vertinimai yra surenkami ir pateikiami visiems ekspertams. Individualūs vertinimai pateikiami anonimiškai. Tada ekspertų klausiama, ar šie nori pakeisti savo individualius vertinimus. Jeigu pamačius kolegų vertinimus nutariama iš naujo vertinti, kiekvienas ekspertas iš naujo nurodo galimo įvykio tikimybę. Vertinimas baigiamas, kada individualūs vertinimai yra artimi vienas kitam ir pasiekiamas konsensusas[9].

Galimų metinių nuostolių vertės skaičiavimas. Tam tikro tipo nuostolius galima lengvai įvertinti, pvz., kaštus, reikalingus programinei ar techninei įrangai pakeisti. Tačiau konfidencialios informacijos nutekėjimo atveju padaryta žala sunkiai įvertinama. Pvz., konkurencinės organizacijos dėl to galėjo įgyti pranašumą rinkoje, paveikti produktų kainas ir pan. Privačių duomenų atskleidimas gali paveikti klientų pasitikėjimą įmone ir t.t. Visai tai įtakoja padarytų nuostolių vertę.

[9] Konsensusas – priėmimas nutarimų bendru sutarimu (be balsavimo, niekam nepriešTaraujant).

Rizikos kontroliavimo priemonių apžvalga. Galimoms problemoms spręsti paprastai yra keli būdai. Svarbu nustatyti kiekvieno iš jų efektyvumą konkrečioje situacijoje, kad būtų galima tinkamai pasirinkti vieną iš jų ar keleto priemonių derinį. Pvz., riziką, kad saugomi duomenys bus sugadinti, galime valdyti periodiškai kurdami atsargines duomenų kopijas, vienu metu saugodami duomenis keliose vietose, valdydami duomenų prieigą, fiziškai saugodami duomenų laikymo patalpas ir pan. Svarbu įvertinti tai, kad tam tikros priemonės gali turėti ir neigiamą efektą. Pvz., duomenų šifravimas gali labai sulėtinti sistemos darbą, taip pat iškyla raktų saugojimo problema.

Skaičiavimai, kiek tam tikrų priemonių naudojimas gali sumažinti nuostolius per metus. Pasiekus šį rizikos analizės etapą, turime būti nustatę galimas grėsmes ir pasirinkę priemones joms valdyti. Reikia įvertinti, ar pasirinktų priemonių teikiama nauda nusveria priemonių naudojimo kaštus. Priemonių efektyvumą galima apskaičiuoti taip:

- Iš galimų nuostolių vertės atimame priemonės sutaupomų nuostolių vertę ir pridedame priemonės įsigijimo ir palaikymo kaštus. Gauname nuostolių vertę naudojant pasirinktą priemonę.
- Iš galimų nuostolių vertės atimame nuostolius naudojantis apsaugos priemonėmis ir gauname priemonės teikiamą naudą – sutaupomų nuostolių vertę.

Priemonių efektyvumo įvertinimo pavyzdžiai pateikti toliau esančiose lentelėse (lentelė 4, lentelė 5).

Lentelė 4. Priemonių rizikoms valdyti efektyvumo vertinimas.

Rizika: Konfidencialių įmonės duomenų sugadinimas dėl netinkamo darbuotojų naudojimosi duomenimis.	
Galimi nuostoliai (rizikos poveikis)	
Duomenų atstatymo kaštai: 1000000 LTL, rizikos pasirodymo tikimybė: 10%.	100000 LTL
Nuostoliai naudojantis saugumo priemonėmis	
Programinė įranga duomenų prieigos kontrolei: efektyvumas 60%, palaikymo kaštai: 25000 LTL.	100000 – (100000 * 0,6) + 25000 = 65000 LTL
Sutaupytos lėšos naudojant priemonę	100000 – 65000 = 35000 LTL

Skaičiuojant efektyvumą kelių vienu metu naudojamų priemonių, reikia naudotis matematine tikimybių teorija. Pvz., tarkime, kad naudosime dvi priemones, kurių kiekvienos suveikimo tikimybė (efektyvumas) yra 60%. Laikykime, kad kiekvienos priemonės

efektyvumas nepriklauso nuo kitos priemonės efektyvumo (tikimybių teorijoje įvykiai būtų laikomi nepriklausomais). Taigi bendras abiejų priemonių efektyvumas yra tikimybė, kad suveiks priemonė A arba priemonė B, arba abi priemonės: $0{,}6 \times 0{,}4 + 0{,}4 \times 0{,}6 + 0{,}6 \times 0{,}6 = 0{,}84 = 84\%$. Kaip atrodytų priemonių efektyvumo vertinimo lentelė aprašytuoju atveju galite matyti toliau esančioje lentelėje 5.

Lentelė 5. Priemonių rizikoms valdyti efektyvumo vertinimas.

Rizika: Konfidencialių įmonės duomenų sugadinimas dėl netinkamo darbuotojų naudojimosi duomenimis	
Galimi nuostoliai (rizikos poveikis)	
Duomenų atstatymo kaštai: 1000000 LTL, rizikos pasirodymo tikimybė: 10%.	100000 LTL
Nuostoliai naudojantis saugumo priemonėmis	
Programinė įranga duomenų prieigos kontrolei: efektyvumas 60%, palaikymo kaštai: 25000 LTL.	100000 – (100000 * 0,84) + 50000 = 66000 LTL
Papildoma priemonė: efektyvumas 60%, palaikymo kaštai: 25000 LTL.	
Sutaupytos lėšos naudojant dvi priemones	100000 – 66000 = 34000 LTL

Matome, kad aprašytuoju atveju naudoti vieną apsaugos priemonę naudingiau nei dvi. Praktiškai nenaudinga gali būti saugotis nuo rizikų, kurių tikimybė labai maža, t.y. apsaugos priemonių kaštai gali viršyti galimos žalos vertę.

Pinigine išraiška vertinti saugumo priemonių teikiamą naudą yra labai naudinga, nes versle dažnai reikia lėšų ir kitoms investicijoms. Techninių žinių neturintys ir atsakingi už lėšų valdymą asmenys gali lengvai suprasti saugumo svarbą ir reikalingumą.

5 DES ALGORITMO KRIPTO ANALIZĖ

Nustačius saugumo problemas CTT sistemoje ir įvertinus jų sprendimo būdus nutarta privačius asmenų duomenis duomenų bazėje saugoti užšifruotus DES algoritmu. Šis algoritmas pasirinktas dėl to, kad yra greitas ir gali būti lengvai pritaikomas. Žinant, jog DES algoritmas buvo „nulaužtas“ naudojant super-mašiną, eksperimento metu buvo stengiamasi nustatyti laiką, reikalingą DES šifro raktui surasti, naudojant asmeninį kompiuterį. Eksperimento tikslas – surasti DES šifro raktą naudojant laisvai prieinamas šiandienos technologines priemones ir įvertinti DES algoritmo saugumą.

Iškart po DES algoritmo pasirodymo vyko daug viešų diskusijų apie jo saugumą. Algoritmo kūrimo metu JAV Nacionalinė Saugumo Agentūra (National Security Agency, NSA) pareiškė, kad algoritmo realizavimo ypatybės yra „jautrios“ (sensitive) ir nebus viešai atskleistos. Dėl realizacijos slaptumo iškilo dvi problemos. Pirmiausiai buvo baiminamasi, kad algoritme yra slaptas įėjimo kelias (trapdoor), leidžiantis NSA dešifruoti bet kurį pranešimą ir sekti privatų asmenų bendravimą. Antra, buvo manoma, kad algoritme buvo surasta projektavimo klaida, kelianti pavojų algoritmo saugumui, kurią buvo norima paslėpti. Taigi niekas negalėjo būti tikras, jog algoritmas yra saugus. Šios dvi priežastys privertė NSA atskleisti DES algoritmą. Algoritmą kruopščiai išnagrinėjo „Bell Laboratories“ [MOR77] ir „Lexan Corporation“ ir jame nebuvo rasta jokių rimtų trūkumų ar saugumo spragų.

Daugelis kripto analitikų dvejojo, ar 16 iteracijų pakanka saugiai „paslėpti“ slaptą informaciją. Eksperimentai [KON81] parodė, kad pakanka 8 iteracijų eliminuoti priklausomybes ar dėsningumus tarp pranešimo ir šifro. Taigi 16 ciklų DES algoritme pilnai pakanka.

Didžiausius ginčus dėl DES saugumo kėlė algoritmo naudojamo rakto ilgis (56 bitai). Juo didesnis rakto ilgis, juo sunkiau yra perrinkti visus galimus raktus. Kadangi algoritmas yra atsparus leksikografinei analizei, tarp pranešimo ir šifro neįmanoma įžiūrėti jokių dėsningumų, vienintelis būdas surasti dešifravimo raktą – perrinkti visus galimus raktus. Tokia ataka vadinama jėgos metodu (brute-force attack). Kadangi DES algoritmo naudojamas raktas yra 56 bitų ilgio, iš viso yra 2^{56} galimų raktų (72057594037927936).

Viena iš strategijų surasti DES šifro raktą – paieška lentelėje. Tarkime, kad turime pranešimą P. Šį pranešimą galime užšifruoti visais galimais raktais ir išsaugoti visus gautus šifrus lentelėje. Norint sužinoti, koks raktas buvo naudojamas, kad gauti šifrą C, tereikia

surasti jį lentelėje. Tokio tipo atakai surengti, reikia turėti galimybę priversti siuntėją išsiųsti tam tikrą mums iš anksto žinomą pranešimą. Taip pat reikia galėti perimti išsiųstą užšifruotą pranešimą. Perėmus šifrą, galima greitai sužinoti siuntėjo naudojamą slaptą raktą.

Naudojantis jėgos metodu (brute-force attack), DES algoritmas buvo nulaužtas 1997 metais. Per keturių mėnesių laikotarpį naudojant apytiksliai 3500 kompiuterių buvo surastas DES šifro raktas. Rakto paieška tarp kompiuterių buvo padalinta taip: kompiuteris A raktus perrinkinėjo pradedant raktu 0000..., kompiuteris B – pradedant raktu 0001..., kompiuteris C – pradedant raktu 0010... ir t.t.

Viena iš žinomų DES algoritmo silpnų vietų yra susijusi su dvejetainių skaičių inversija (inversija vadinama tokia dvejetainio skaičiaus operacija, kada visi 0 pakeičiami 1, o visi 1 pakeičiami 0). Jeigu tam tikrą pranešimą *p* užšifruosime raktu *k* ir gausime šifrą *c*, tai šifro *c* inversijai gauti šifravimo metu reikia naudoti invertuotus pranešimą *p* ir raktą *k*:

$$c = DES(p,k)$$

$$\bar{c} = DES(\bar{p},\bar{k})$$

Čia $\bar{x}$ žymi x inversiją, p yra pranešimas, k – raktas.

Praktikoje ši matematinė savybė yra laikoma nesvarbia.

Kita iš žinomų silpnų algoritmo vietų yra tam tikrų raktų naudojimas. Kadangi raktas yra padalinamas į dvi dalis ir su jomis nepriklausomai viena nuo kitos atliekamos perstūmimo operacijos, naudojant raktą, kurio pusę sudaro visi 0 arba visi 1, postūmio operacijos rezultatas visada yra visi 0 arba visi 1. Naudojant tokius raktus, antrą kartą šifruojant šifrą gaunamas originalus pranešimas, t.y. tas pats rezultatas kaip ir dešifruojant šifrą:

$$c = DES_{encrypt}(p,k)$$

$$DES_{encrypt}(c,k) = DES_{decrypt}(c,k) = p$$

Čia $DES_{encrypt}$ yra šifravimo algoritmas, $DES_{decrypt}$ – dešifravimo algoritmas, p – pranešimas, k – raktas, c – šifras po pirmojo šifravimo proceso.

Kadangi tokie raktai yra žinomi (žr. lentelę 6), jų tiesiog nereikia naudoti. Taigi ši problema taip pat nelaikoma rimta. Lentelėje 6 raktai pavaizduoti šešioliktainėje formoje. Kadangi prieš pirmąjį rakto padalinimą atliekamas bitų perstatymas, pradinis raktas nėra

„pusė nulių, pusė vienetų“ formos. Nurodyti raktai yra tokie, iš kurių prieš pirmąjį padalinimą gaunama minėtoji forma.

Lentelė 6. Nenaudotini DES raktai.

	Kairioji pusė	Dešinioji pusė	Raktas
1	Nuliai	Nuliai	`0101 0101 0101 0101`
2	Vienetai	Vienetai	`FEFE FEFE FEFE FEFE`
3	Nuliai	Vienetai	`1F1F 1F1F 0E0E 0E0E`
4	Vienetai	Nuliai	`E0E0 E0E0 F1F1 F1F1`

Tiriant daugelį šifravimo algoritmų, bandoma surasti tokius skirtingus raktus, kuriuos naudojant šifravimo metu būtų gautas identiškas šifras. Šis reiškinys angliškai vadinamas *key clustering*. Keturi nenaudotini DES algoritmo raktai ir yra tokie raktai, tačiau DES algoritmo tyrimo metu buvo bandoma surasti kitus tokius raktus. Tyrimo metu buvo rasta tokių raktų, kurie davė tuos pačius tarpinius rezultatus iki trečiojo ciklo (iš 16). Tolesni ciklai nebuvo nagrinėjami dėl per didelio sudėtingumo.

Darbo metu buvo sukurta programa, matuojanti laiką, reikalingą DES šifro raktui surasti. Programos realizacija ypatinga tuo, kad rakto gali ieškoti daug gijų (thread) vienu metu. Tokiu būdu galima imituoti daugelio lygiagrečiai sujungtų procesorių darbą. Teoriškai, jeigu vienas procesorius per vieną mikrosekundę patikrina vieną raktą, tai per dieną būtų galima patikrinti $8{,}6 \times 10^{10}$ raktų, o per 10^6 dienų būtų galima patikrinti visus $2^{56} \approx 7 \times 10^{16}$ raktus. Tuomet galima manyti, kad 10^6 lygiagrečiai sujungtų procesorių raktą galėtų surasti per vieną dieną. Kadangi tokios galingos mašinos kaina yra milžiniška, mažai tikėtina, kad DES algoritmą gali nulaužti paprastas vartotojas, naudojantis asmeninį kompiuterį. Vis dėlto pingant techninei įrangai ir tobulėjant technologijoms, galimas daiktas, kad DES algoritmą ateityje bus galima lengvai įveikti. Praktikoje sistemos sparta paprastai neišauga dvigubai padvigubinus lygiagrečiai veikiančių procesorių skaičių. Priežastis yra ta, kad reikalingi papildomi resursai procesams sinchronizuoti, iškyla techninės problemos dėl bendrų duomenų pasiekimo ir pan. Sukurta programa buvo bandoma apskaičiuoti realų rakto paieškos pagreitėjimą priklausomai nuo veikiančių procesorių skaičiaus.

Eksperimentinės programos įėjimo duomenys yra šie:

- Šifruojamas pranešimas;
- Šifravimo raktas;

- Rakto ieškančių procesų skaičius;
- Rakto ilgis, nuo kurio pradedama paieška;
- Rakto ilgis, kurį pasiekus baigiama paieška.

Eksperimentinės programos pateikiami rezultatai:

- Visas rakto paieškos laikas;
- Apskaičiuotas vidutinis laikas, reikalingas 10000 raktų patikrinti.

Eksperimento metu nebuvo galimybės įvertinti programos darbo rezultatų naudojant daugia-procesorinę sistemą. Programa buvo bandoma kompiuteryje, kuriame veikė keturių branduolių procesorius (Intel Q6600 2,40 GHz), todėl realų rakto paieškos pagreitėjimą galime vertinti tik iš eksperimentų, kuriuose naudojamų procesų skaičius yra nuo 1 iki 4.

Pagrindinis programos uždavinys – nurodyti procesams tikrinti skirtingus raktus. Raktų paskirstymo būdas nėra labai svarbus, jeigu reikia perrinkti visus galimus raktus, tačiau praktikoje jis gali būti reikšmingas, nes daug kas priklauso nuo paties rakto. Pvz., sakykime, kad ieškome rakto ZZ, sudaryto iš 2 simbolių. Galimi raktai yra: `AA, AB, AC, ..., AZ, BA, BB, ..., ZZ`. Du procesai rakto ieškoti galėtų taip:

- Pirmasis procesas ieško nuo rakto AA iki vidurio, antrasis procesas – nuo vidurio iki rakto ZZ. Jeigu laikysime, kad pusę raktų vienas procesas perrenka per N laiko, tai raktą ZZ rasime po N laiko (po N laiko pirmasis procesas pasieks vidurinį raktą, o antrasis – paskutinįjį raktą.).
- Pirmasis procesas paiešką pradeda nuo rakto AA ir tikrina kas antrą raktą, antrasis procesas pradeda nuo rakto AB ir taip pat tikrina kas antrą raktą. Naudojant šį modelį, raktas ZZ bus surastas po 2N laiko, tačiau tai nereiškia, kad šis modelis prastesnis. Pvz., raktą, esantį prieš pat vidurinįjį raktą, šiuo modeliu būtų galima surasti per vidutiniškai N/2 laiko, o pirmasis modelis rastų po vidutiniškai N laiko.

Sukurtoje eksperimentinėje programoje kiekvienas procesas tikrina kas n-tąjį raktą, kur n – ieškančių procesų skaičius. Pvz., kai ieškančių procesų skaičius yra 4, pagal aukščiau pateiktą pavyzdį raktai būtų tikrinami taip:

```
Gija1: AA, AE, AI, …, ZT.
Gija2: AB, AF, AY, …, ZU.
```

```
Gija3: AC, AG, AJ, …, ZV.
Gija4: AD, AH, AK, …, ZZ.
```

Svarbu paminėti, kad eksperimentinėje programoje raktai buvo generuojami ne iš visos simbolių aibės, o tik iš mažųjų raidžių nuo a iki z. Nuo simbolių aibės priklauso galimų raktų skaičius, tačiau, skaičiuojant procentinį rakto paieškos laiko pagreitėjimą, simbolių aibė nėra svarbi. Rakto simbolių aibė buvo apribota, kad sumažinti rakto paieškos laiką eksperimentų metu.

Eksperimento rezultatai

Lentelė 7. DES šifro rakto paieškos laikas.

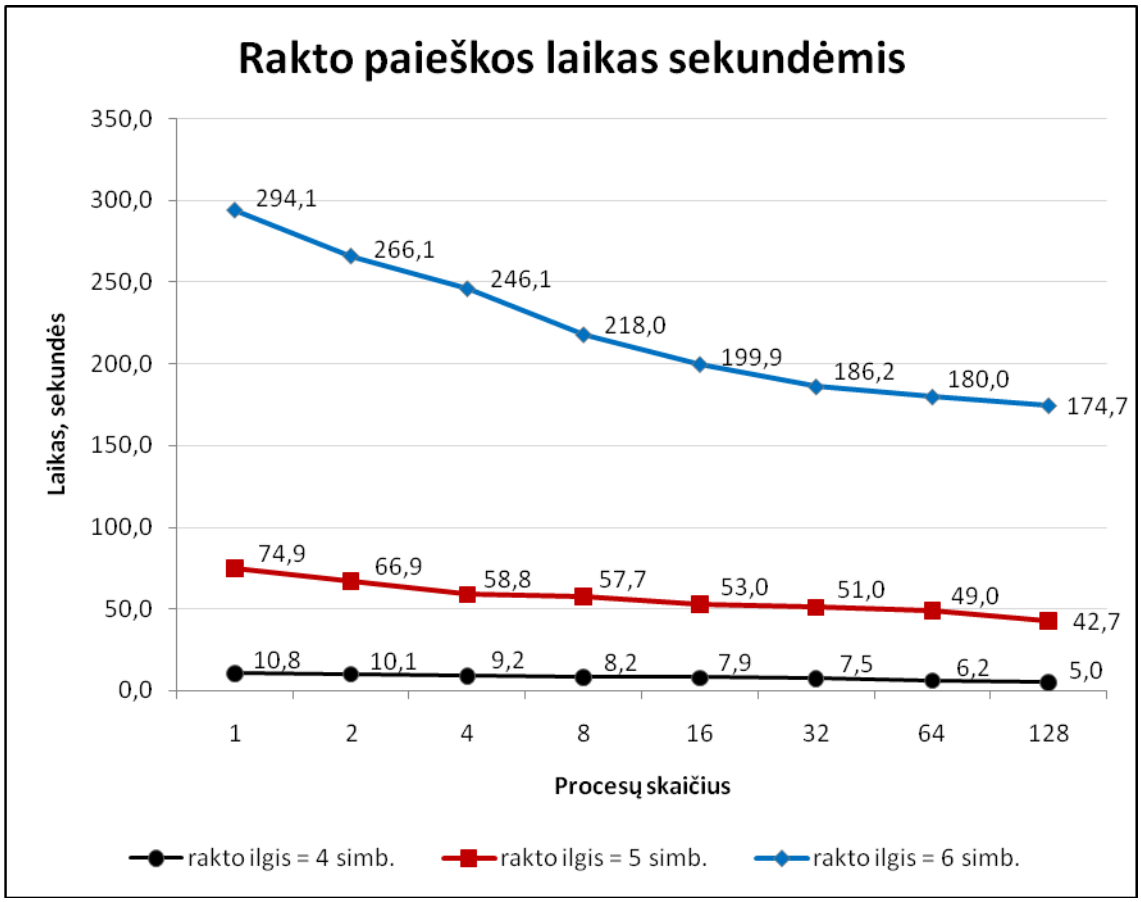
Rakto paieškos laikas sekundėmis			
Rakto ilgis / Procesų skč.	4 ("zzzz")	5 ("ggggg")	6 ("abbbbb")
1	10,8	74,9	294,1
2	10,1	66,9	266,1
4	9,2	58,8	246,1
8	8,2	57,7	218,0
16	7,9	53,0	199,9
32	7,5	51,0	186,2
64	6,2	49,0	180,0
128	5,0	42,7	174,7

Lentelė 8. DES šifro rakto paieškos laiko pagreitėjimas.

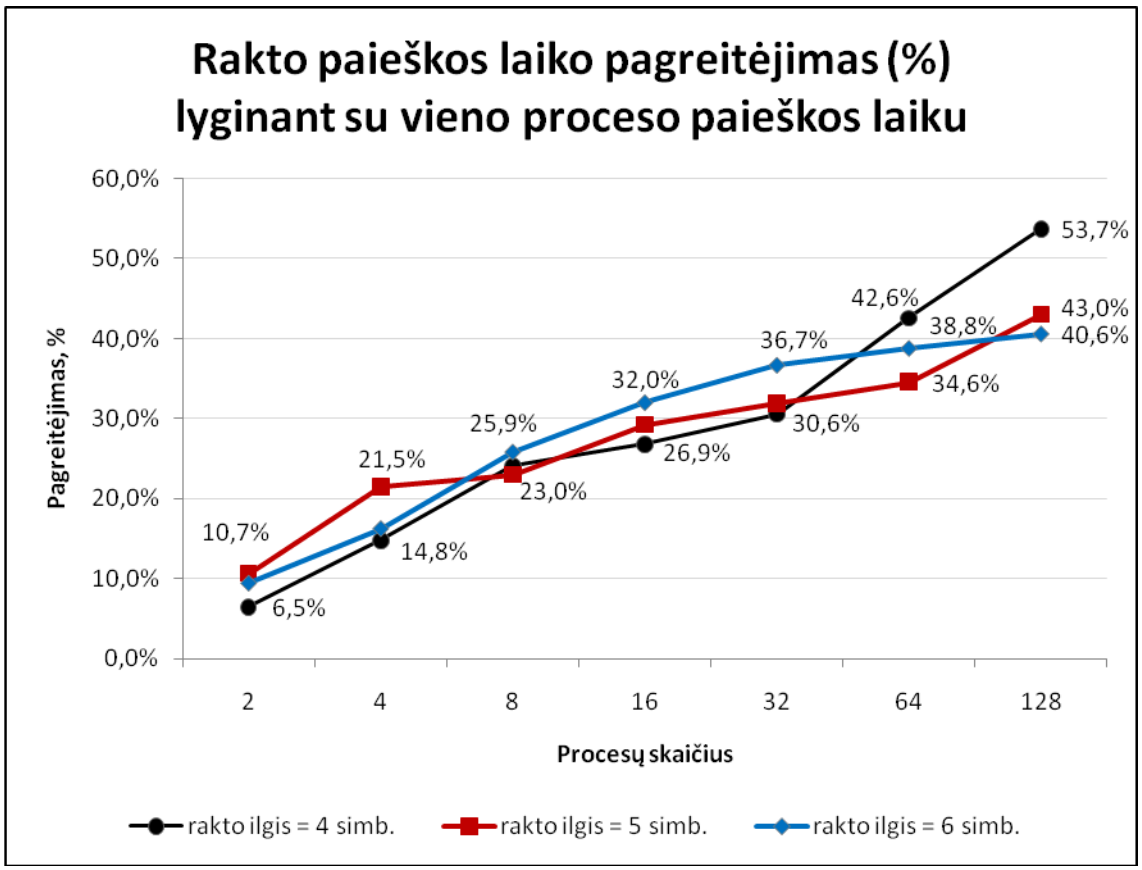
Rakto paieškos laiko pagreitėjimas (%) lyginant su vieno proceso paieškos laiku			
Rakto ilgis / Procesų skč.	4 ("zzzz")	5 ("ggggg")	6 ("abbbbb")
2	6,5%	10,7%	9,5%
4	14,8%	21,5%	16,3%
8	24,1%	23,0%	25,9%
16	26,9%	29,2%	32,0%
32	30,6%	31,9%	36,7%
64	42,6%	34,6%	38,8%
128	53,7%	43,0%	40,6%

Lentelė 9. DES šifro rakto paieškos laiko pagreitėjimas.

Pagreitėjimas (%) dvigubinant procesų skaičių			
Rakto ilgis / Prcs. skč. dvg.	4 ("zzzz")	5 ("ggggg")	6 ("abbbbb")
nuo 1 iki 2	6,5%	10,7%	9,5%
nuo 2 iki 4	8,9%	12,1%	7,5%
nuo 4 iki 8	10,9%	1,9%	11,4%
nuo 8 iki 16	3,7%	8,1%	8,3%
nuo 16 iki 32	5,1%	3,8%	6,9%
nuo 32 iki 64	17,3%	3,9%	3,3%
nuo 64 iki 128	19,4%	12,9%	2,9%



Paveikslėlis 25. DES šifro rakto paieškos laikas.



Paveikslėlis 26. DES šifro rakto paieškos laiko pagreitėjimas.

Kaip ir buvo tikimasi, rakto paieškos laiko pagreitėjimui rakto ilgis įtakos neturi. Nuo rakto ilgio priklauso tik galimų raktų skaičius, nes bet kuriuo atveju raktu operuojama kaip 56 bitų skaičiumi.

Prieš eksperimentą buvo daroma prielaida, kad, „įdarbinus“ du kartus daugiau procesorių, paieškos laikas taip pat dvigubai sumažės. Visgi realūs rezultatai rodo ką kitą. Du kartus padidinus procesų skaičių, paieškos laikas vidutiniškai sumažėdavo tik 9% (vertinant tik procesų skaičiaus padidėjimą nuo 1 iki 2 ir nuo 2 iki 4). Nedidelį pagreitėjimą galėjo lemti kelios priežastys:

- Vienas keturių branduolių procesorius ir keturi lygiagrečiai sujungti procesoriai nėra tas pats.
- Keturi vieno procesoriaus branduoliai naudojasi bendra atmintimi ir magistrale, todėl pilnu pajėgumu gali veikti tik tam tikromis specialiomis sąlygomis.
- Eksperimentinė programa veikia kaip vienas procesas, o rakto paieškos procesai realizuoti kaip gijos (thread), o ne kaip atskiri procesai (process).
- Skaitymo iš atminties laikas yra didesnis už matematinių operacijų laiką.
- Procesoriaus laiko paskirstymas gijoms priklauso nuo procesoriaus architektūros, operacinės sistemos (eksperimentas atliktas su Windows XP SP2 64 bitų sistema).

Naudojant daugiau nei 4 gijas, pagreitėjimas tebebuvo matomas. Tą lėmė operacinė sistema, programai skirdama daugiau procesoriaus laiko dėl jos didelio gijų skaičiaus, taip pat kol dalis iš atminties skaitančių gijų laukė duomenų, kitos gijos galėjo naudotis procesoriumi skaičiavimams atlikti.

Galime daryti išvadą, kad DES algoritmas naudojantis šiandienos technine įranga yra vis dar neįveikiamas, t.y. raktui surasti reikalingas laikas yra pakankamai ilgas.

6 IŠVADOS

Nustačius saugumo problemas CTT sistemoje ir įvertinus jų sprendimo būdus pasiūlyta:

- Nesaugaus interneto ryšio problemą spręsti šifruojant „jautrius“ duomenis viešo rakto sistema RSA.
- Privačius asmenų duomenis duomenų bazėje saugoti užšifruotus DES algoritmu (šifravimo rakto saugumas priklauso nuo serverio saugumo).
- Organizacines ir pasitikėjimo programine įranga problemas spręsti organizaciniu lygiu – suformuluoti saugumo politiką ir jos laikytis, propaguoti kokybės valdymo standartus, sistemų testavimą atlikti ne tik organizacijos viduje, bet ir patikėti šį uždavinį nepriklausomoms testuotojų komandoms.

Duomenims šifruoti pasirinktas DES algoritmas. Nors šis algoritmas yra įveiktas naudojant super-mašiną, jo teikiamas saugumo lygis yra pakankamas CTT sistemai. Be to, DES algoritmas yra greitas ir nesunkiai pritaikomas. DES algoritmo saugumo buvo įsitikinta atlikus eksperimentą, kurio metu buvo bandoma surasti DES šifro raktą. Aštuonių ir daugiau simbolių rakto naudojimas šiandien gali užtikrinti informacijos saugumą.

Darbo metu nustatyta, kad pagrindinės priežastys, lemiančios saugumo spragų programinėje įrangoje pasirodymą yra tyčinės ir netyčinės klaidos bei testavimo nebuvimas. Pagrindinės veiklos klaidoms PĮ surasti ir išvengti yra: (1) Nuolatinė peržiūra, (2) Rizikų analizė ir prognozavimas, (3) Testavimas, (4) Statinė kodo analizė, (5) Produkto versijų kontrolė, (6) Klaidų analizė, (7) Komponentinis/modulinis kūrimas, (8) Kokybės standartų naudojimas.

7 LITERATŪRA

[ANT01] ANTON, P., et al. *Finding and Fixing Vulnerabilities in Information Systems: The Vulnerability Assessment & Mitigation Methodology*. RAND Corporation, MR-1601-DARPA, 2004. ISBN/EAN: 0833034340.

[BAL04] BALFANZ, D., et al. In Search of Usable Security: Five Lessons from the Field. *IEEE Security & Privacy*, 2004, vol. 2 nr. 5, p. 19-24.

[CHA81] CHAUM, D. Untraceable Electronic Mail, Return Addresses and Digital Pseudonyms. *Communications of the ACM*, 1981, vol. 24 nr. 2, p. 84-88.

[CHA85] CHAUM, D. Security Without Identification: Transaction Systems to Make Big Brother Obsolete. *Communications of the ACM*, 1985, vol. 28 nr. 10, p. 1030-1044.

[CLA06] CLARK, N.; ir WALD, M. Hurdle for US in Getting Data on Passengers. *New York Times*, 2006 m. gegužės 31 d. Prieiga internete: <http://www.nytimes.com/2006/05/31/world/europe/31air.html>.

[CMM01] NSA (National Security Agency). *SSE-CMM: Systems Security Engineering – Capability Maturity Model*. Prieiga internete: <http://www.sse-cmm.org>.

[HAL67] HALMER, O. *Analysis of the Future: The Delphi Method*. RAND Corporation, P-3558, 1967. ISBN/EAN: 0833034340.

[ISO06] ISO (International Organization for Standardization). *ISO 9001:2000: A Workbook for Service Organizations*. International Organization for Standardization, 2006.

[JON91] JONES, T. *Applied Software Measurement*. McGraw-Hill, 1991. ISBN: 0070328137.

[KON81] KONHEIM, A. *Cryptography, A Primer*. John Wiley & Sons Inc., 1981. ISBN: 0471081329.

[LAM76] LAMPSON, B.; ir STURGIS, H. Reflections on an Operating System Design. *Communications of the ACM*, 1976, vol. 19 nr. 5, p. 251-265.

[LAN94] LANDAU, S., et al. Crypto Policy Perspectives. *Communications of the ACM*, 1994, vol. 37 nr. 8, p. 115-121.

[MER81] MERKLE, R.; ir HELLMAN, M. On the Security of Multiple Encryption. *Communications of the ACM*, 1981, vol. 24 nr. 7, p.465.

[MOR77] MORRIS, R., et al. Assessment of the National Bureau of Standards Proposed Data Encryption Standard. *Cryptologia*, 1977, vol. 1 nr. 3, p. 281-291.
[MUD95] MUDGE. How to Write Buffer Overflows. *L0pht Heavy Industries Report*, 1995 m. spalio 20 d. Prieiga internete: <http://reactor-core.org/overflow-howto.html>.

[MUL99] MULLIGAN, D. Testimony of the Center for Democracy and Technology. *Public Workshop on Online Profiling*, 1999 m. lapkričio 30 d. Prieiga internete: <http://www.cdt.org/testimony/ftc/991130mulligan.shtml >.

[OWA06] OWASP (Open Web Application Security Project). A Guide to Building Secure Web Applications and Web Services. *OWASP Guide Project*, ver. 2.1 draft 3, 2006 m. vasaris. Prieiga internete: <http://www.owasp.org/index.php/Guide_Table_of_Contents>.

[PAU93] PAULK, M., et al. Capability Maturity Model, ver. 1.1. *IEEE Software*, 1993, vol. 10 nr. 4, p. 18-27.

[PFL01] PFLEEGER, S., et al. *Solid Software*. Prentice-Hall PTR, 2001. ISBN: 0130912980.

[PFL91] PFLEEGER, S. A framework for Security Requirements. *Computers & Security*, 1991, vol. 10 nr. 6, p. 515-523.

[REZ03] REZGUI, A., et al. Privacy on the Web: Facts, Challenges, and Solutions. *IEEE Security & Privacy*, 2003, vol. 1 nr. 6, p. 40-49.

[SHO82] SHOCK, J.; ir HUPP, J. The "Worm" Programs – Early Experience with a Distributed Computation. *Communications of the ACM*, 1982, vol. 25 nr. 3, p. 172-180.

[TSI05] TSIPENYUK, K., et al. Seven Pernicious Kingdoms: A Taxonomy of Software Security Errors. *IEEE Security & Privacy*, 2005, vol. 3 nr. 6, p. 81-84.

[TUR05] TUROW, J., et al. Open to Exploitation: American Shoppers Online and Offline. *Annenberg Public Policy Center of the University of Pennsylvania*, 2005. Prieiga internete: < http://repository.upenn.edu/asc_papers/35/>.

[TUR75] TURN, R.; ir WARE, W. *Privacy and Security in Computer Systems*. RAND Corporation, P-5361, 1975.

Dalį literatūros sąraše paminėtų leidinių galima rasti internete, žemiau nurodytais adresais.

Žurnalas „IEEE Security & Privacy" internete:
<http://csdl2.computer.org/persagen/DLPublication.jsp?pubtype=m&acronym=sp>

Žurnalas „IEEE Software" internete:
<http://csdl2.computer.org/persagen/DLPublication.jsp?pubtype=m&acronym=so>

Žurnalas „Communications of the ACM“ internete:
<http://portal.acm.org/toc.cfm?id=J79&idx=J79&type=periodical&coll=portal&dl=ACM&part=magazine&WantType=Magazines>

Žurnalas „Computers & Security“ internete:
<http://www.sciencedirect.com/science/journal/01674048>

Žurnalas „Cryptologia“ internete:
<http://www.informaworld.com/smpp/title~content=t725304178~db=all>

RAND korporacijos elektroniniai dokumentai ir knygos:
<http://rand.org/pubs/technical_reports>

Security in Computing: Threats and their Management, Encryption Systems

Summary

Is security in computing a problem? There are many methods and ways developed to help in software quality management. However, the need to quickly release a new software or its version is often more important than software security requirements or threats analysis.

The main goal of this work is to analyse the threats in computing and methods to manage software security. The main objectives are:

- To recognize and discuss the threats in computing;
- To find the reasons that are causing security problems while studying the software development process;
- To find ways to eliminate the threats or minimize their impact.

First of all, there is a study of technical reasons that are causing vulnerabilities and threats in software to appear. Then there is a review of existing methods to manage software security and quality. An important part of this work is dedicated to privacy, private data management problem. There are methods discussed that ensures privacy in computing.

After the review of technical aspects of security, there are methods discussed that insures security from organization's point of view.

Finally, there is a review of encryption systems, there types, differences and measures.