KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

Donatas Brazauskas

Skaitmeninio turinio teisių valdymo peržiūros atjungties režimu sistemos sudarymas ir tyrimas

Baigiamasis magistro darbas

Vadovas
Doc. Agnius Liutkevičius

KAUNAS, 2017

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
KOMPIUTERIŲ KATEDRA

Skaitmeninio turinio teisių valdymo peržiūros atjungties režimu sistemos sudarymas ir tyrimas

Baigiamasis magistro darbas
Informacijos ir informacinių technologijų sauga (kodas 621E10003)

Vadovas
(parašas) Doc. Agnius Liutkevičius
(data)

Recenzentas
(parašas) Prof. Egidijus Kazanavičius
(data)

Projektą atliko
(parašas) Donatas Brazauskas
(data)

KAUNAS, 2017

KAUNO TECHNOLOGIJOS UNIVERSITETAS

Informatikos fakultetas

(Fakultetas)

Donatas Brazauskas

(Studento vardas, pavardė)

„Informacijos ir informacinių technologijų sauga“ (621E10003)

(Studijų programos pavadinimas, kodas)

„Baigiamojo projekto pavadinimas“

AKADEMINIO SĄŽININGUMO DEKLARACIJA

20 17 m. gegužės 22 d.

Kaunas

Patvirtinu, kad mano **Donato Brazausko** baigiamasis projektas tema „Skaitmeninio turinio teisių valdymo peržiūros atjungties režimu sistemos sudarymas ir tyrimas“ yra parašytas visiškai savarankiškai, o visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

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

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

Brazauskas, D. „Skaitmeninio turinio teisių valdymo peržiūros atjungties režimu sistemos sudarymas ir tyrimas“. Magistro baigiamasis projektas / vadovas doc. Agnius Liutkevičius; Kauno technologijos universitetas, Informatikos fakultetas, Kompiuterių katedra.

Kaunas, 2017. 60 p.

SANTRAUKA

Šiuo metu skaitmeninio turinio teisių valdymo sistemos yra vis plačiau naudojamos visuose išmaniuose įrenginiuose, pradedant išmaniaisiais TV imtuvais, kompiuteriais, baigiant mobiliaisiais įrenginiais. Šios sistemos leidžia tik teisėtam savininkui naudotis skaitmeniniu turiniu. Tačiau jos nėra tokios patogios vartotojo atžvilgiu, kaip galėtų būti.

Darbe išanalizuoti skaitmeninio turinio teisių valdymo sistemų veikimo principai ir jų galimi apsaugos metodai. Palyginus rinkoje esančias skaitmeninio turinio teisių valdymo sistemas, nustatyta, kad nei viena sistema neturi galimybės perkelti vaizdo failą į kitą to paties vartotojo įrenginį ir jį ten peržiūrėti. Todėl nuspręsta sukurti savo skaitmeninio turinio teisių valdymo sistemą, kuri turėtų tokią galimybę.

Skaitmeninio turinio teisių valdymo sistemos kūrimui pasirinkta „Android“ OS, multimedijos failus užšifruojant *AES-CTR* šifravimo metodu. Multimedijos failų iššifravimo raktai saugomi tarp mobiliosios programėlės failų, užšifruotame tekstiniame faile, o failo iššifravimo raktas sugeneruojamas pagal mobilaus išmaniojo įrenginio techninius duomenis ir mobilios programėlės reikšmes. Taip mobilioji programėlė susieta tik su vienu įrenginiu ir tik jame gali tinkamai funkcionuoti atjungties režimu. Mobiliąją programėlę apsaugoti nuo piktavalių parinkta „ProGuard“ apsauga.

Sukurtos skaitmeninio turinio teisių valdymo sistemos tyrimas sudarytas iš galimybės peržiūrėti vaizdo failą kituose vartotojo įrenginiuose ir mobiliosios programėlės apsaugos apėjimo, naudojant apgrąžos inžineriją. Pirmojo tyrimo rezultatai yra teigiami ir patvirtina darbe pasiūlytos skaitmeninio turinio teisių valdymo sistemos idėją. Antroji tyrimo dalis pavyko labai gerai, nes pavyko nustatyti, jog standartinė „Proguard“ apsauga nepakankama ir jos geriau nenaudoti. Ji tik prailgino sugaištą laiką nuo penkių iki dvidešimt minučių, ieškant turinio apsaugos funkcijų pirminiame kode, bet funkcijos vis tiek buvo rastos ir išanalizuotos. Kadangi „ProGuard“ apsauga yra nepakankama ir tinkamai neapsaugo mobiliosios programėlės kaip tikėtasi, šiai problemai spręsti rekomenduojama naudoti komercinius apsaugos nuo apgrąžos inžinerijos analogus, turinčius daugiau apsaugos galimybių, tame tarpe ir apsaugą realiu laiku.

Brazauskas, Donatas. The creation and investigation of digital rights management system for viewing digital content offline: *Master's* thesis in "Information and Information Technology Security" / supervisor assoc. prof. Agnius Liutkevičius. The Faculty of Informatics, Kaunas University of Technology.

Research area and field: offline DRM system in mobile devices

Key words: Android, mobile devices, DRM, offline

Kaunas, 2017. 60 p.

SUMMARY

Today digital rights management systems are more and more widely used in Smart devices from Smart TV receivers, PC and mobile phones as well. These systems allow only the rightful owner to use the digital content. However, they are not as user-friendly as they could be.

The principles and the protection methods of digital rights management systems are analyzed in the work. While comparing digital rights management systems, which are presented in the market, it is noticed, that no one of these have the possibility to transfer the video file to another device and view it there. Therefore, it was decided to create an own digital rights management system with this opportunity.

Android OS was chosen for the implementation of digital rights management system, encrypting multimedia files with *AES-CTR* encryption method. Multimedia content decryption keys are stored together with mobile application files in encrypted text file, while this text file decryption key is generated using the mobile device technical data and application values. This application is associated with one device, and it can properly function offline. *ProGuard* software was selected as protection from reverse engineering.

The research of created digital rights management system is composed from the ability to view the video file in other user devices and application resistance from reverse engineering. The first results of research are positive and confirms proposed digital rights management system concept. The second part of research was successful, because it helped to determine that standard *Proguard* protection is insufficient and it is better not to use. It just prolonged the time to the search for protection functions in the original code from five to twenty minutes, but these functions were still found and analyzed. Since *ProGuard* protection is insufficient and does not protect properly mobile application as expected, to resolve this problem it is recommended to use commercial anti-reverse engineering analogs with more security options, including real-time protection.

TURINYS

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

TERMINŲ IR SANTRUMPŲ ŽODYNAS

DRM – Digital rights management

DB – Duomenų bazė

OS – Operacinė sistema

STTV – Skaitmeninio turinio teisių valdymas

IV – inicializacijos vektorius

Android – atviro kodo operacinė sistema, skirta išmaniesiems įrenginiams

Android_ID – *Android* OS identifikacinis įrenginio sugeneruotas kodas

IMEI – unikalus kodas, skirtas identifikuoti mobilųjį įrenginį (angl. *International Mobile Equipment Identities*)

ESN – elektroninis serijos numeris (angl. *Electronic serial number*)

MEID – mobilaus įrenginio identifikacinis numeris (angl. *Mobile Equipment Identifier*)

Serial – identifikacinis išmaniojo įrenginio kodas, skirtas įrenginiams, kurie neturi IMEI/MEID ar ESN kodo.

UML – modeliavimo ir specifikacijų kūrimo kalba (angl. *Unified Modeling Language*)

ĮVADAS

Šiandien beveik kiekvienas žmogus turi išmanųjį telefoną. Pagal „Kleiner Perkins Caufield & Byers“ (KPCB) įmonės atliktą tyrimą JAV [1] jau 2014 m. žmonės daugiau laiko praleidžia prie telefono ar planšetės, nei prie asmeninio ar nešiojamo kompiuterio, ir kiekvienais metais šis skaičius didėja.

Išmanūs telefonai yra mobilūs, jais galima naudotis nepriklausomai nuo esamos vietos, turi didelius ekranus, pajėgią techninę įrangą, kuri gali atvaizduoti geros kokybės vaizdo ir garso turinį. Todėl šiuos įrenginius labai plačiai pradėta naudoti skaitmeninio turinio peržiūrai. Dėl to pradėta galvoti, kaip apsaugoti skaitmeninį turinį nuo „piratų“, kad turinys nebūtų kopijuojamas, jis priklausytų tik tam vienam ar keliems asmenims, kurie jį įsigijo. Todėl mobiliesiems įrenginiams buvo pritaikytos skaitmeninio turinio teisių valdymo sistemos, kurios leidžia tik savininkui naudotis jam priklausančiu turiniu.

Darbo problematika ir aktualumas

Pirmoji problema – vartotojui sukeliamas nepatogumas, kai yra reikalinga nuolatinė interneto prieiga skaitmeninio turinio peržiūros metu. Todėl, esamos STTV sistemos netenkina vartotojų poreikių ir yra nepatogios naudotis, nes neleidžia peržiūrėti turinio atjungties režimu.

Skaitmeninio turinio apsauga nuo neteisėto naudojimo. Šiuo metu yra sugalvota daug būdų, kaip apsaugoti turinį. Darbe nagrinėjama skaitmeninio turinio apsaugos technologijos, kurios veiktų be interneto prieigos ir tuo pat metu užtikrintų pakankamą turinio apsaugą.

Energijos sąnaudų sumažinimas. Mobiliuosiuose įrenginiuose vienas svarbiausių komponentų – akumuliatorius. Turint išmanųjį telefoną, juo naudojamės didžiąją dienos dalį, o tai lemią, kad akumuliatorius ilgai netarnauja, tai telefonas vakaro gali nesulaukti – išsikrauti. Naudojantis apsaugotu skaitmeniniu turiniu pagrindiniai energijos suvartojimo kaltininkai yra nuolatinis interneto ir iššifravimo veikimas.

Darbo tikslas ir uždaviniai

Darbo tikslas – sukurti mobiliesiems įrenginiams skirtą skaitmeninio turinio teisių valdymo sistemą, kuri veiktų be nuolatinės interneto prieigos, turėtų galimybę perkelti parsisiųstą multimedijos failą į kitą vartotojo įrenginį ir taip pat gerai apsaugotų skaitmeninį turinį nuo neteisėto naudojimo.

Uždaviniai:

1. atlikti analizę ir ištirti esamas, mobiliesiems įrenginiams skirtas skaitmeninio turinio teisių valdymo sistemas, jų problemas ir pažeidžiamumus;
2. pasiūlyti tinkamą apsaugos metodą ir priemones iškeltam tikslui pasiekti;
3. eksperimentiškai ištirti siūlomą apsaugos metodą, realizuojant skaitmeninio turinio apsaugos sistemos prototipą ir jį išbandant.

Darbo rezultatai ir jų svarba

Šiame magistriniame darbe pasiūlytas originalus skaitmeninio turinio teisių valdymo apsaugos metodas, kuris, skirtingai nuo egzistuojančių metodų, leidžia skaitmeninio turinio failus peržiūrėti atjungties režimu tik turinį įsigijusio vartotojo naudojamuose įrenginiuose. Pasiūlyto metodo pagrindu realizuota „Android“ mobilioji programėlė, kurios efektyvumas patvirtintas eksperimentiškai. Pasiūlytas metodas ne tik apsaugo turinį nuo neteisėto disponavimo ir peržiūros neautorizuotuose įrenginiuose, bet ir leidžia vartotojams peržiūrėti turinį bet kuriuo metu, neturint interneto prieigos, tuo pat metu taupant peržiūrai naudojamų įrenginių energiją.

Darbo struktūra

Šiame darbe yra trys pagrindiniai skyriai.

Pirmajame skyriuje pateikta skaitmeninio turinio apsaugojimo būdai ir skaitmeninio turinio teisių valdymo sistemų palyginimas. Skyriaus tikslas yra išanalizuoti skaitmeninio turinio apsaugojimo budus, ištirti esamas skaitmeninio turinio teisių valdymo sistemų problemas ir pažeidžiamumus.

Antrajame skyriuje aprašytas, sugalvotas naujas, originalus skaitmeninio turinio teisių valdymo metodas. Skyriaus tikslas detaliai aprašyti sugalvotą naująjį metodą, tam pateikiamos *UML* detalizuojančios diagramos ir jų paaiškinimai.

Trečiame skyriuje aprašytas skaitmeninio turinio teisių valdymo realizavimas su naujuoju metodu, bei jo tyrimas. Skyriaus tikslas ištirti ar sugalvota ir realizuota naują sistema išsprendžia išsikeltas problemas.

Darbo pabaigoje pateikta apibendrintas visas darbas, jo išvados, naudotas literatūros sąrašas.

1. SKAITMENINIO TURINIO TEISIŲ VALDYMO SISTEMŲ MOBILIEMS ĮRENGINIAMS ANALIZĖ

1.1. Analizės tikslas

Analizės tikslas yra ištirti esamas mobiliesiems įrenginiams skirtas skaitmeninio turinio teisių valdymo (STTV) sistemas, jų teigiamas ir neigiamas puses.

Analizės uždaviniai:

a) išsiaiškinti STTV sistemų veikimo principus;
b) išanalizuoti galimas STTV sistemas, įvertinti jų privalumus, trūkumus;
c) surasti turinio apsaugos sistemas ar metodus, kurie gali veikti atjungties režimu;
d) apibendrinti gautus analizės rezultatus.

1.2. Tyrimo objektas, sritis ir problema

Tyrimo objektas – skaitmeninio turinio teisių valdymo sistemos.

Tyrimo sritis – skaitmeninio turinio sauga. Tiriamos skaitmeninio turinio apsaugos sistemos ir metodai mobiliuose įrenginiuose.

STTV sistemų problema, kurios leidžia parsisiųsti multimedijos failus į „Android“ įrenginį, juos susieja prie pačių įrenginių ir neleidžia parsisiųsto failo perkelti į kitus vartotojo įrenginius. Taip vartotojui sukelia nepatogumus, nes failus jam reikia parsisiųsti iš naujo į visus savo įrenginius.

1.3. Skaitmeninio turinio teisių valdymo sistemų ir jų kūrimo analizė

1.3.1. Skaitmeninio turinio teisių valdymo sistema

STTV (angl. DRM – Digital Rights Management) – prieigos kontrolės sistemos, skirtos apsaugoti turinį nuo vartotojų, kurie neturi teisių juo naudotis, ir turinio kopijavimo. Šių sistemų paskirtis - apsaugoti bet kokį turinį, pradedant knygomis, muzika, baigiant programine įranga ir žaidimais [2].

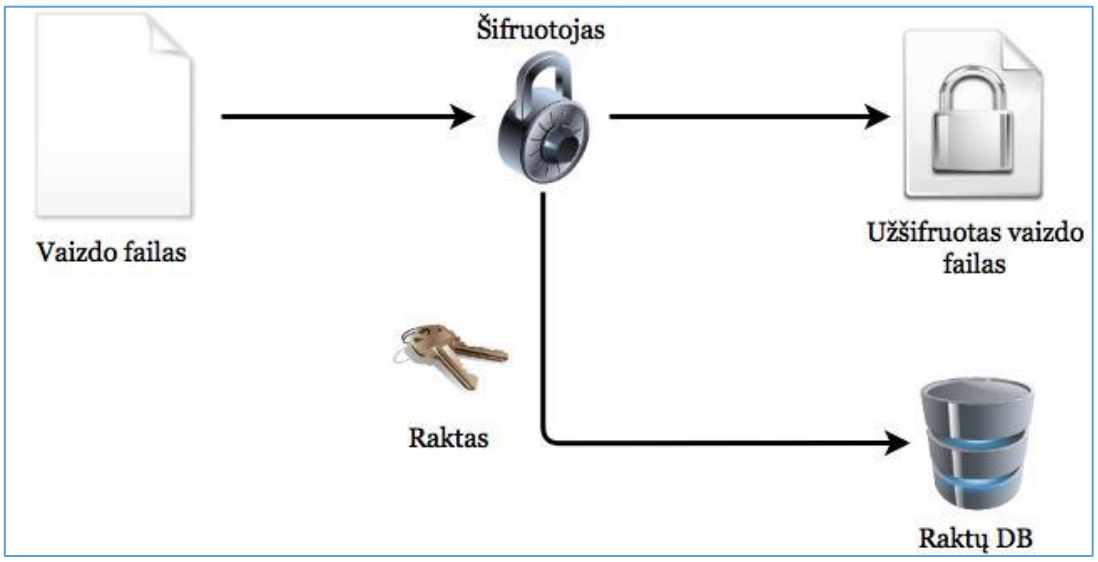
STTV sistemos yra sudarytos iš dviejų modulių: užšifravimo – apsaugoti skaitmeninį turinį ir autentifikavimo sistemos – užtikrinti, kad tik vartotojai, turintys teisę, tai pasiektų.

1.3.1.1. Skaitmeninio turinio teisių valdymo sistemų veikimo principas

Kad geriau suprasti, kaip veikia STTV sistemos, reikia išsiaiškinti bent paprasčiausią jų veikimo metodą. STTV bendras veikimo principas yra sudarytas iš trijų žingsnių.

1.3.1.1.1. Turinio užšifravimas

Prieš paleidžiant vaizdo failą į rinką, jį reikia užšifruoti ir paversti į kitą formatą.

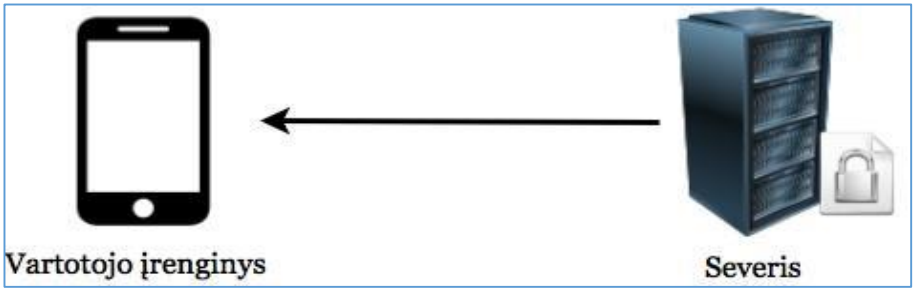


1 pav. Turinio užšifravimas

Kai failas yra užšifruotas, raktas ir metaduomenys, reikalingi iššifruoti vaizdo failą, yra išsaugomi apsaugotoje „Raktų“ duomenų bazėje vėlesniam vartotojui (žr. 1 pav.) [3].

1.3.1.1.2. Turinio pristatymas, atsisiuntimas ir peržiūra

Kai vaizdo failas yra užšifruotas, jis paruoštas įkelti į serverį. Vartotojui, panorėjus peržiūrėti vaizdo failą, įrenginys kreipiasi į serverį ir serveris grąžina manifesto failą. Manifesto faile yra užšifruota vaizdo failo buvimo vieta, STTV sistemos informacija ir metaduomenys (žr. 2 pav.) [3].



2 pav. Turinio pristatymas/atsisiuntimas/peržiūra

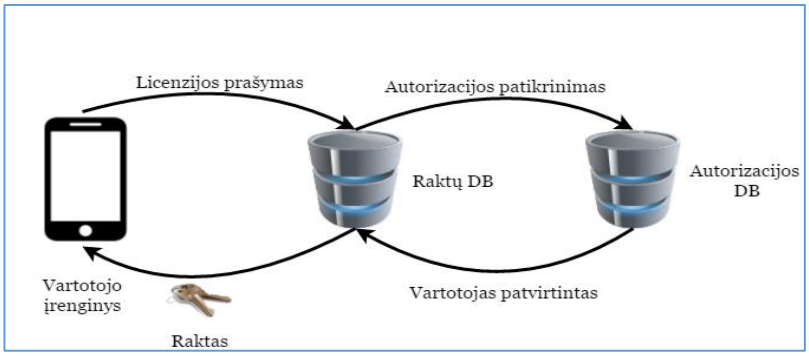
Turiniui peržiūrėti STTV sistemos grotuvas pirmiausia turi turėti licencijos raktą, kad galėtų iššifruoti vaizdo failą. Be šio rakto turinio peržiūrėti neįmanoma.

1.3.1.1.3. Licencijos gavimas

Specialus vaizdo turinio grotuvas su integruotu STTV įskiepiu, pasinaudoja informacija iš manifesto failo ir siunčia užklausą licencijai gauti. Yra daug sugalvotų būdų kaip gauti licenciją, bet šie du būdai yra populiariausi [3].

Pirmas būdas:

Vaizdo turinio grotuvas siunčia prašymą gauti licencijai į „Raktų“ DB.

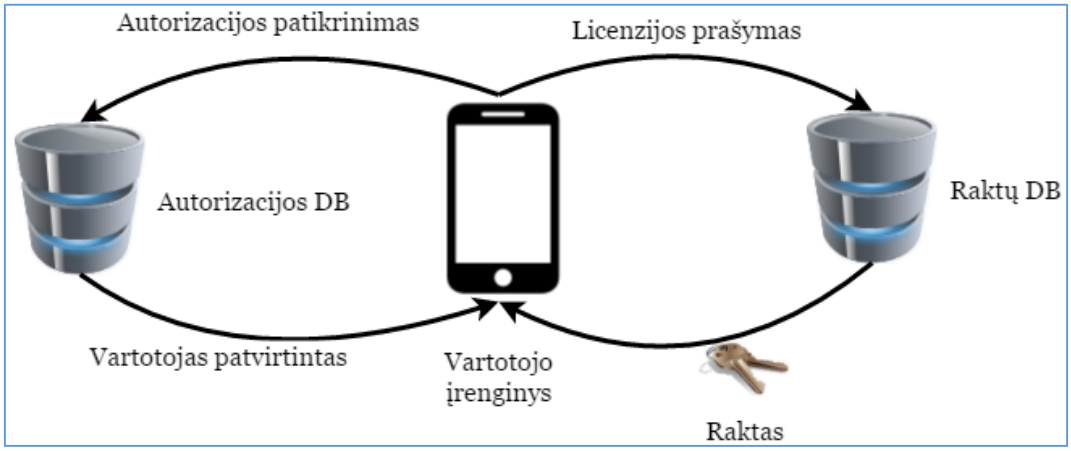


3 pav. Licencijos gavimas, pirmas būdas

„Raktų“ DB siunčia užklausą į „Autorizacijos“ DB, kad patikrintų, ar vartotojas turi teisę peržiūrėti šį turinį. Jei vartotojas turi teisę peržiūrėti turinį, yra gražinamas patvirtinimas į „Raktų“ DB. „Raktų“ DB, gavusi teigiamą atsakymą, išsiunčia vartotojui licenciją su iššifravimo raktu turiniui atrakinti (žr. 3 pav.) [3].

Antras būdas:

Pirmiausia, vaizdo turinio grotuvas siunčia užklausą į „Autorizacijos“ DB ir tikrina, ar vartotojas turi teisę peržiūrėti turinį. Jei vartotojas turi tokią teisę, „Autorizacijos“ DB sugeneruoja požymį (angl. *token*), kuriame slypi autorizacijos duomenys ir gražina jį vaizdo grotuvui [3].



4 pav. Licencijos gavimas, antras būdas

Tik vaizdo grotuvas ir duomenų bazės gali peržiūrėti požymį. Naudojant požymį, vaizdo grotuvas, siunčia užklausą į „Raktų“ DB. Išsiuntus požymį, jis nustoja galioti, kad daugiau juo nebebūtų galima pasinaudoti. „Raktų“ DB išsiunčia vartotojui licenciją su iššifravimo raktu, skirtu atrakinti turinį [3].

1.3.2. Kiti skaitmeninio turinio apsaugos būdai

1.3.2.1. Skaitmeninio turinio vandenženklio žymėjimas

Skaitmeninė vandenženklio žymė (angl. *Digital watermark*) – duomenys, įterpti į skaitmeninį turinį, skirti parodyti turinio kūrėją ar savininką. Skaitmeninio turinio vandenženklių žymėjimas – procesas nusakantis metodus ir technologijas, kuris paslepia informaciją skaitmeniniame turinyje, pvz.: skaitmeninio teksto dokumentuose, paveikslėliuose, garso ir vaizdo failuose [4].

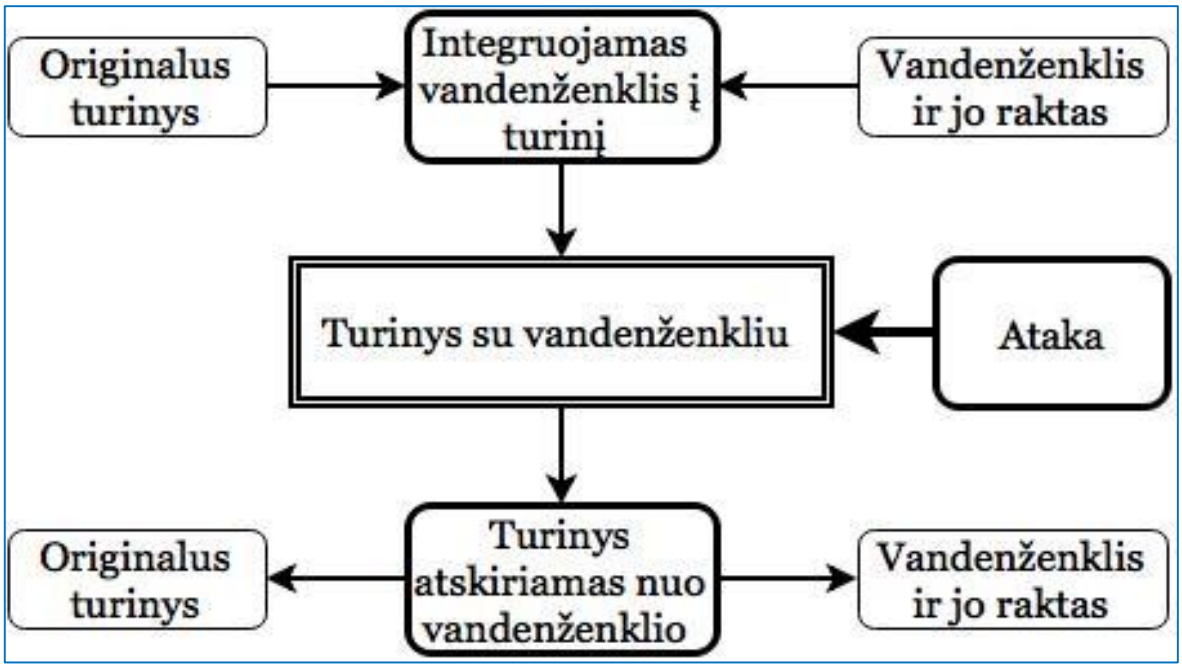
1.3.2.1.1. Skaitmeninio turinio vandenženklio reikalavimai

Egzistuoja trys pagrindiniai reikalavimai skaitmeninio turinio vandenženklio žymėjimui. Tai nepastebimumas (angl. *perceptivity*), atsparumas (angl. *robustness*) ir disko vietos talpa [5].

a) nepastebimumas – į turinį įterptas skaitmeninis vandenženklis, dėl kurio neturėtų suprastėti turinio kokybė;
b) atsparumas nuo atakų – gebėjimas aptikti vandenženklį po bendro signalo apdorojimo operacijos. Vandenženkliai, turi būti atsparūs įvairiems geometriniams ir ne geometriniams išpuoliams;
c) disko vietos talpa – bitų skaičius, kuriuos užima užkoduotas vandenženklis per darbo laiką įrenginyje. Tai nusako, kiek disko vietos reikia, sėkmingai ištraukti aptiktą vandenženklį.

1.3.2.1.2. Skaitmeninio turinio vandenženklio žymėjimo veikimas

Skaitmeninio turinio vandenženklio žymėjimo veikimas yra sudarytas iš trijų žingsnių: vandenženklio integravimas į turinį, ataka ir turinio atskyrimas nuo vandenženklio (žr. 5 pav.) **[6]**.

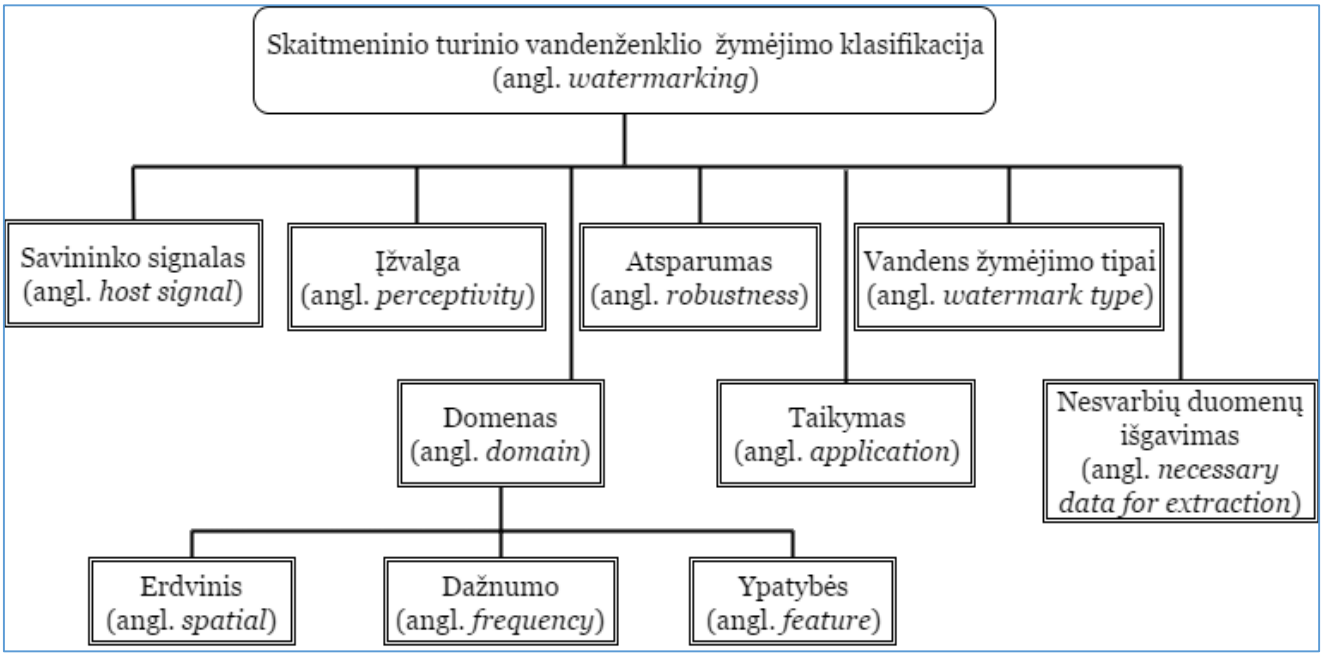


5 pav. Vandenženklio veikimo principas

1. Integruojamas vandenženklis į turinį – originaliam turiniui, vandenženkliui ir jo raktui susijungus į vieną turinį, jis būna paruoštas naudojimui.
2. Ataka – tai skaitmeninis turinys, kuris perduodamas kitiems vartotojams, patalpinamas į serverį, laikmeną ar į kitą talpyklą. Asmenys, norintys peržiūrėti turinį, bando jį atsidaryti ar kitaip paveikti apsaugos mechanizmą. Atakos yra įvairiausių formų, pvz.: vaizdo ir paveikslėlių failai – karpomi, apsukami, keičiama jų rezoliucija ir t. t.
3. Turinio atskyrimas nuo vandenženklio - t. y. iš turinio išimamas vandenženklis ir vartotojas gali mėgautis turiniu.

1.3.2.1.3. Skaitmeninio turinio vandenženklio klasifikavimas

Skaitmeninio turinio vandenženklio žymėjimo klasifikacija, sudaryta iš septynių tipų (žr. 6 pav.) [5].



6 pav. Skaitmenio turinio vandenženklio žymėjimo klasifikacija

1. **Savininko signalas (angl. *host signal*)**
   a) Tekstas – į šifro formą, tarp simbolių ir tarp linijos tarpų erdvės įdedamas vandenženklis.
   b) Paveikslėlis – į paveikslėlį įdedama speciali informacija ir aptikus vandenženklį ar jį ištraukus yra nustatomas, kas paveikslėlio savininkas.
   c) Vaizdo failas – tai paveikslėlio pratęsimas.
   d) Garso failas – į garso failo takelį yra įrašoma informacija. Žmogaus ausis tokio dažnio negirdi.

2. Įžvalga (angl. *perceptivity*)

Skaitmeniniai vandenženkliai yra padalinti į matomus ir nematomus [5].

a) Matomas vandenženklis – tai lyg logotipas ar štampas ant dokumentų, pvz.: televizijoje naudojami kanalo logotipai.
b) Nematomas vandenženklis – naudojamas identifikuoti turinio teises, autorius ir kitokią informaciją.

3. Atsparumas (angl. *robustness*)

Atsparumas – kaip gerai vandenženklis apsaugo turinį nuo įvairių atakų. Atsparumo yra trys klasifikacijos [5].

a) Stiprus – atsparus įvairioms geometrinėms ir negeometrinėms atakoms nepakenkiant turiniui.
b) Vidutinis – apsaugo tam tikrą laipsnį atakų, neleidžia pakeisti paveikslėlio su vandenženkliu.
c) Silpnas – vandenženklis yra skirtas lengvai susinaikinti, t. y. jeigu paveikslėlis su vandenženkliu yra manipuliuojamas bent menkiausiu būdu. Ši klasifikacija yra skirta originalaus turinio patikrai.

4. Vandenženklio tipai (angl. *watermark type*)

Vandenženklio tipai gali būti klasifikuojami į du tipus [5]:

a) Triukšmo – turi netikrą (angl. *pseudo*) triukšmą, Gaus‘o atsitiktines ir chaotiškas sekas.
b) Paveikslėlio – dvejetainis vaizdas, antspaudas, logotipas ar etiketė.

5. Svarbių duomenų išgavimas (angl. *necessary data for extraction*)

Vandenženklio informacijai nustatyti yra naudojamos trys schemos [7]:

a) Matomas vandenženklis – schema, dar žinoma, kaip privati skaitmeninio turinio vandenženklio schema. Ši sistema, reikalauja originalių turinio duomenų, vandenženklio aptikimui. Sistema ištraukia vandenženklį W iš galimai sugadintų duomenų I' ir panaudoja originalų turinį kaip užuominą, kur vandenženklis gali būti I'. Ši schema yra saugiausia iš šių trijų.

   Formulė I x I' x K x W -> {0,1}; (1)

   čia I' – vandenženklio duomenys, K – raktas, W – vandenženklis.
b) Beveik matomas vandenženklis – schema, dar žinoma, kaip pusiau privati skaitmeninio turinio vandens žymėjimo schema. Ši sistema nereikalauja originalių turinio duomenų vandenženklio aptikimui. Sistemos tikslas – aptikti ar vandenženklis egzistuoja.

   Formulė I' x K x W -> {0,1}; (2)
c) Nematomas vandenženklis – schema, dar žinoma, kaip vieša schema. Tai yra pati sunkiausia vandenženklio sistema, nes jai nereikia, nei turinio originalių duomenų, nei

vandenženklio. Ši sistema ištraukia *n* bitus iš vandenženklio duomenų, kurie yra paveikslėlyje su vandenženkliu.

Formulė I' x K -> W; (3)

6. Domenai (angl. *domain*)

Skaitmeninio turinio vandenženklio žymėjimą galima pritaikyti erdvinio ir ypatybės domenui [5].

Erdvinis domenas (angl. *spatial domain*) – vandenženklis yra įdedamas į intensyvumo reikšmes. Geriausias ir plačiausiai žinomas algoritmas – LSB metodas. Jis pagrįstas redaguojantis mažiausiai reikšmingų paveikslėlio sluoksnių bitų. Todėl paveikslėlis išlieka nepakitęs ar iškraipytas [5].

Ypatybės domenas (angl. *Transform Domain*) – apsaugotas paveikslėlis yra perduodamas į procesą, kuriame jis yra transformuojamas į neapsaugotą paveikslėlį, taip apsaugojamas nuo pašalinių asmenų. Transformuoti galima pasitelkus tokius procesus kaip *Discrete Cosine Transform (DCT), Discrete Wavelet Transform (DWT), Discrete Fourier Transform (DFT), Contourlet Transform (CT)* ir *Singular Value Decomposition (SVD)* [5].

7. Taikymas (angl. *applications*)

Skaitmeninio turinio žymėjimas gali būti pritaikytas:

a) skaitmeninio turinio teisių apsaugai ir autentifikavimui;
b) piršto anspaudams ir skaitmeniniams parašams;
c) duomenų autentifikavimui;
d) apsaugai nuo turinio kopijavimo ir įrenginio kontrolės;
e) transliacijos stebėjimui;

1.4. Skaitmeninio turinio teisių valdymo sistemų analizė

Šiame skyriuje STTV valdymo sistemos apžvelgiamos ir palyginamos viena su kita.

Lyginimo kriterijus sudaro:

1. operacinė sistema, turi būti „Android“, nes ji yra populiariausia tarp mobiliųjų įrenginių;
2. vaizdo ir garso turinio atvaizdavimas;
3. „nenulaužta“ STTV sistemos apsauga;
4. vaizdo ir garso turinio perkėlimo galimybė į kitą įrenginį;
5. veikimas atjungties režimu.

Lyginamos šios penkios populiariausios STTV sistemos, kurios buvo atrinktos *Google, Google Mokslinčiuje* ir KTU bibliotekos duomenų bazėse, kurių paieškoje suvedus raktinius žodžius „DRM systems for mobile“ ir žinomiausios STTV sistemos kaip „Netflix“ ar „Flixter“.

„Flixster“ yra *Time Warner* kompanijos produktas, saugantis skaitmeninį turinį. Jis veikia „Windows 8“ir produktuose, palaikančiuose *Silverlight* technologiją. Taip pat gali veikti ir „Android“,

„Blackberry“, „iOS“ ir „Windows Phone“ platformose [8]. „Flixster“ naudoja „Ultraviolet“ debesų tarnybomis paremtą skaitmeninio turinio apsaugos sistemą [9]. Šis produktas leidžia parsisiųsti multimedijos failus į mobilųjį įrenginį, peržiūrai atjungties režimu, bet juos galima matyti tik per „Flixster“ mobiliąją programėlę, bei jų negalima perkelti į kitą įrenginį.

„VUDU“ yra *Vudu* kompanijos produktas. Jis skirtas apsaugotam turiniui perduoti internetu į išmaniuosius įrenginius ir televizorius. „Vudu“ gali veikti išmaniuosiuose televizoriuose, žaidimų kompiuteriuose (*Xbox One, Playstation 4* ir t. t.), o išmaniesiems telefonams yra reikalinga mobilioji programėlė [10]. Šis produktas taip pat naudojasi „Ultraviolet“ bibliotekomis [9]. „VUDU“ produktas kaip ir „Flixster“ leidžia parsisiųsti multimedijos failus į išmanųjį įrenginį, peržiūrai atjungties režimu, bet jie matomi tik jų mobiliąją programėlę ir jų negalima perkelti į kitą įrenginį.

„Amazon Prime Video“ yra *Amazon* internetinis multimedijos pareikalavus (angl. *internet video on demand*) paslauga. Ji veikia išmaniuosiuose televizoriuose, optinių diskų grotuvuose (*Blu-ray*), žaidimų kompiuteriuose, mobiliuose įrenginiuose su „iOS“, „Android“ operacinėmis sistemomis [11]. Šis produktas taip pat leidžia parsisiųsti multimedijos failus į mobilųjį įrenginį, peržiūrai atjungties režimu, bet vartotojas šiuos failus gali matyti tik per „Amazon Prime Video“ mobiliąją programėlę, be to jų negalima perkelti į kitą vartotojo įrenginį.

„Ultraviolet“ – *DECE* kompanijos debesų tarnybomis paremtas produktas. Jis gali veikti ir produktuose su *Android* OS ir kompiuteriuose su *Windows, Mac* ar *Linux* operacinėmis sistemomis. Jis apjungia šešias STTV sistemas į vieną sistemą [9] [12]:

a) *Adobe Primetime DRM* (*formerly Adobe Flash Access 2.0*);
b) *CMLA-OMA V2*;
c) *Marlin DRM Open Standard*;
d) *Microsoft PlayReady*;
e) *Widevine* (*at the time an independent company, later acquired by Google*);
f) *DivX DRM* (*approved after the original five*).

Šios kompanijos produktas leidžia parsisiųsti multimedijos failus į mobilųjį įrenginį, peržiūrai atjungties režimu, bet vartotojas juos mato tik per „Ultraviolet“ mobiliąją programėlę ir jų negalima perkelti į kitą įrenginį.

„Netflix“ – JAV kompanija, teikianti multimedijos transliavimo paslaugas internetu [13]. Ji veikia visame pasaulyje išskyrus Kiniją, Siriją, Šiaurės Korėją ir Krymo teritoriją [14]. Netflix veikia optinių diskų grotuvuose (*Blu-ray*), išmaniuosiuose TV, „PlayStation“, „Xbox“, „Chromecast“ įrenginiuose, taip pat išmaniuosiuose įrenginiuose su „Android OS“, „IOS“ ir kompiuteriuose su „Windows“, „Mac“ ar „Linux“ operacinėmis sistemomis [15]. „Netflix“ kaip visos lyginamos STTV sistemos leidžia parsisiųsti multimedijos failus į mobilųjį įrenginį, peržiūrai atjungties režimu, bet juos galima matyti tik per „Netflix“ mobiliąją programėlę ir jų negalima perkelti į kitą įrenginį.

1 lentelė Skaitmeninio turinio teisių valdymo sistemų lyginamoji lentelė

Lyginimo kriterijai	„Flixter“	„VUDU“	„Amazon Prime Video“	„Ultraviolet“	„Netflix“
Veikia „Android“ OS	+	+	+	+	+
Atvaizduoja vaizdo failus	+	+	+	+	+
„Nenulaužtas“	+	+	+	+	+
Galimybė perkelti parsisiųsta vaizdo failą į kitą įrenginį	-	-	-	-	-
Veikia atjungties režimu.	+	+	+	+	+

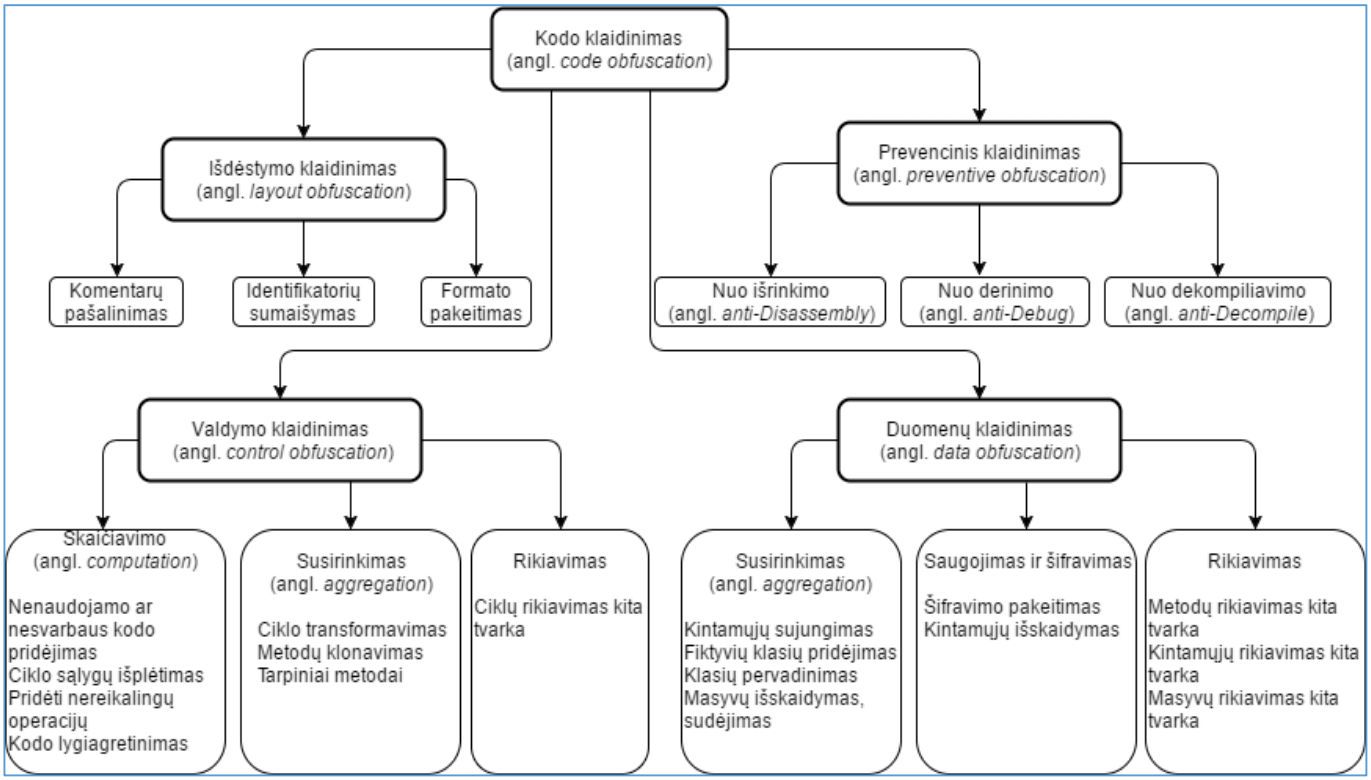
Išvada: atsižvelgiant į 2 lentelėje pateiktus palyginimo rezultatus, visos STTV sistemos palaiko multimedijos failų peržiūrą be interneto prieigos. Taip pat visos sistemos neturi galimybės perkelti multimedijos failo į kitą įrenginį, todėl su šia galimybe bus kuriama savo sistema.

1.5. „Android“ mobiliųjų programėlių apsaugojimas

Viena svarbesnių STTV sistemos kūrimo dalių yra apsaugoti mobiliąją programėlę nuo piktavalių. Šiame skyrelyje aprašoma, kokiomis priemonėmis, galima apsunkinti piktavaliams pasiekti „Android“ mobiliosios programėlės pirminį kodą, jį išanalizuoti ir įveikti apsaugą.

„Android“ mobiliųjų programėlių apsaugai yra sukurta daug metodų ir tai vadinama kodo klaidinimu (angl. *code obfuscation*). Kodo klaidinimas – pradinio kodo pakeitimas taip, kad žmogui, panaudojus apgrąžos inžineriją (angl. reverse engineering), būtų jį sunku skaityti, suprasti. Apgrąžos inžinerija – egzistuojančios sistemos analizės procesas, atliekamas sistemos projektinės dokumentacijos atkūrimui [16]. Ją dažniausiai naudoja programišiai, kurie nori apeiti programinės įrangos apsaugos mechanizmus. Panaudojus apgrąžos inžinerijos įrankius, programišius atkuria originalų mobiliosios programėlės pradinį kodą. Taip pat programišiui gali ir nepasisekti atkurti viso pradinio kodo, kadangi mobilioji programėlė apsaugota kodo klaidinimo programomis.

Kodo klaidinimas skirstomas į pagrindines keturias grupes: išdėstymo, prevencinį, valdymo ir duomenų klaidinimo [17] (žr. 7 pav.).



7 pav. Kodo klaidinimo pagrindinės grupės

Išdėstymo klaidinimas – išdėstymo transformacija, kuri naudojama modifikuoti programos pradinį kodą ir dvejetainę struktūrą. Pakeičiant kintamųjų vardus į kinų, o ne alfabetinius simbolius, sudėtingėja kodo skaitymas [17].

Prevencinis klaidinimas – prevencinės transformacijos metodai, kurie naudojami išvengti komercinių derinimo ir dekompiliavimo įrankių. Prevencinis klaidinimas prideda nenaudojamų ar retai naudojamų *bytecode* į pirminį kodą, nepakeičiant jo pirminio veikimo, taip apsunkina perprasti pirminio kodo veikimą. [17].

Valdymo srauto klaidinimu yra siekiama suklaidinti asmenį, bandanti perprasti, kaip veikia pradinis kodas. Funkciniai blokai, kurie kartu priklauso, yra išskaidomi, o funkciniai blokai, kurie kartu nepriklauso, yra sujungiami vienas su kitu, siekiant kuo labiau suklaidinti. Valdymo srautas, pakeičia programos vykdymo kelius, bet išlaiko nepakeistą originalų funkcionalumą. Valdymo srauto klaidinimas susideda iš trijų dalių: skaičiavimo, susirinkimo ir rikiavimo (žr. 7 pav.) [17].

Duomenų klaidinimo metodas modifikuoja mobiliosios programėlės struktūrą. Jis klasifikuojamas į šias tris kategorijas: surinkimo, saugojimo ir šifravimo, rikiavimo (žr. 7 pav.) [17].

Panaudojus bent vieną iš šių metodų apsaugoti mobiliąją programėlę, programišiui bus sunkiau perskaityti ir analizuoti pirminį kodą. Kuo daugiau metodų panaudota, tuo didesnė tikimybė, kad

programišiui nepavyks greitai perprasti pradinio kodo. Taip pat šios apsaugos neužtikrina pilnos sistemos saugumo, nes turint resursų ir laiko, visos apsaugos yra įveikiamos.

Apsisaugoti nuo apgrąžos inžinerijos analizuojamos šiam tikslui sukurtos programos, pagal šiuos lyginimo kriterijus:

1. veikia *Java* kalboje, nes *Android* mobiliosios programėlės kuriamos šia kalba;
2. pašalina nereikalingą tekstą, komentarus, komentaruose gali būti aprašytas funkcijų veikimas;
3. pervadina procedūrų pavadinimus ir kintamuosius, kad klaidintu;
4. užšifruoja klases, procedūras ir kintamuosius, kad klaidintu;
5. tikrina, ar nėra kode pakeitimų, kad įsitikinti, ar pirminis kodas yra originalus;
6. tikrina, ar įrenginys turi administratoriaus teises, nes turint šias teises padidėja rizika išanalizuoti mobilios programėlės veikimą
7. nemokamas.

Apsisaugoti nuo apgrąžos inžinerijos yra sukurta ne viena programinė įranga ar integruojama biblioteka. Tokių sistemų yra daug, bet pasirinktos populiariausios šios trys apsaugos sistemos:

ProGuard – programa, sukurta *Free Software Foundation*. Ji aptinka ir pašalina nenaudojamas klases, laukus, metodus ir atributus. Ji optimizuoja *bytecode* ir pašalina nenaudojamas instrukcijas. Taip pat pervadina likusias klases, laukus ir metodus, naudojant trumpus beprasmius pavadinimus. Gauti pradinio kodo failai ir bibliotekos yra mažesnės, greičiau ir geriau apsaugo nuo apgrąžos inžinerijos, nei nenaudojant jokios apsaugos [18].

DexGuard – mokamas *GuardSquare* kompanijos produktas. Jis apsaugo programas nuo klonavimo, klastojimo, raktų gavybos ir „piratavimo“. Ji naudoja tokius metodus, kaip tekstinės reikšmės šifravimas, klasės šifravimas, kreipimusi į procedūras paslėpimas ir kodo klaidinimas [19].

DexProtector – mokamas produktas. Ji skirta apsaugoti „Android“ programas nuo sugadinimų ar pakeitimų. Jis turi daug funkcijų, kurios padeda apsaugoti „Android“ programėlės kodą nuo apgrąžos inžinerijos [20].

2 lentelė Apsaugos nuo apgrąžos inžinerijos lyginamoji lentelė

Lyginimo kriterijai	„ProGuard“	„DexGuard“	„DexProtector“
Veikia *Java* kalboje	+	+	+
Pašalina nereikalingą tekstą, komentarus	+	+	+
Pervadina procedūrų pavadinimus ir kintamuosius	+	+	+
Užšifruoja klases, procedūras ir kintamuosius	-	+	+
Tikrina, ar nėra kode pakeitimų	-	+	+
Tikrina, ar įrenginys turi administratoriaus teises	-	-	+
Nemokama	+	-	-

Išvada: remiantis 2 lentelės duomenimis, „DexProtector“ apsauga yra geriausia, bet ji yra mokama. Pasirinkta „ProGuard“ apsauga, nors ji nusileidžia savo galimybėmis mokamiems produktams, bet ji yra nemokama.

1.6. Šifravimo algoritmas multimedijos failams

Norint apsaugoti failą su vartotojo multimedijos failų licencijomis reikia jį užšifruoti. Aiškinantis, kuris šifravimo metodas yra tinkamiausias kuriamai sistemai, remtasi moksliniu straipsniu „A Study of Encryption Algorithms AES, DES and RSA for Security“ [21].

Lyginimo kriterijai sudaryti iš:

1. raktų dydis ne mažesnis nei 256 bitai;
2. blokų dydis ne mažesnis nei 128 bitai;
3. užšifravimo greitis mažesnis nei 1 MB per 3 sek.;
4. iššifravimo greičio mažesnis nei 1 MB per 3 sek.;
5. mažų energijos sąnaudų, reikalingų iššifruojant failą;
6. apsaugos lygis saugus.

DES – duomenų šifravimo standartas (angl. *Data Encryption Standard*) – blokinis simetrinis algoritmas, kurio blokų ilgis yra 64 bitai, rakto ilgis yra 56 bitai. DES duomenų šifravimo standartu paskelbtas JAV 1977 metais [22].

RSA – viešojo rakto kriptosistema, kurios algoritmą 1978 m. sukūrė R. Rivest‘as , A. Šamir‘as ir L. Eidlman‘as. Tai – dažniausiai naudojamas viešojo rakto algoritmas. RSA algoritmas gali būti naudojamas ir užšifruoti, ir dešifruoti. Jis yra laikomas saugiu, kai yra naudojami „dideli“ raktai (512 bitų raktas yra nesaugus, 768 bitų – vidutiniškai saugus, 1024 bitų ─ saugus) [23].

AES – pažangus šifravimo standartas (angl. *Advanced Encryption Standard*) šifravimo algoritmas, 2001 metais JAV paskelbtas standartu. Dar vadinamas Rijndaelo algoritmu, jo autoriai yra Joanas Daemen‘as ir Vincentas Rijmen‘as. Rijndael‘as yra blokinis simetrinis algoritmas. Šifravimo raktai gali būti 128, 192 arba 256 bitų ilgio, o bloko ilgis 128 bitų [24].

3 lentelė Šifravimo algoritmų lyginamoji lentelė [21]

Lyginimo kriterijai	DES	RSA	AES
Raktų dydis	–	+	+
Blokų dydis	–	+	+
Užšifravimo greičio	–	–	+
Iššifravimo greičio	+	–	+
Energijos sąnaudos, reikalingos iššifruojant	+	–	+
Apsaugos lygis	–	–	+

Remiantis 3 lentelės duomenimis, AES šifravimo metodas geriausiai tinka, nes jis greičiausiai užšifruoja ir iššifruoja failus, vartoja mažai energijos sąnaudų ir yra saugiausias iš lyginamųjų šifravimų. Pasirinktam *AES* šifravimo metodui, reikia nustatyti režimą.

Geriausią režimą išrinkti padeda sudaryti lyginimo kriterijai:

1. multimedijos failo užšifravimo;
2. galimybės iššifruoti failą ne nuo pradžių;
3. kiek kartų reikia iššifruoti raktui;
4. energijos sąnaudos sunaudojimo apie 1000 šifravimui.

4 lentelė Šifravimo algoritmas multimedijos failams [25]

Lyginimo kriterijai	ECB	CBC	CFB	OFB	CTR
Multimedijos failo užšifravimas	-	+	+	+	+
Galimybė iššifruoti failo ne nuo pradžių	+	-	+	-	+
Kiek kartų reikia iššifruoti raktui	12440	13549	88709	38465	46474
Energijos sąnaudos sunaudojimas apie 1000 šifravimui	0,348	0,404	0,418	0,412	0,375

Išvada: remiantis 4 lentelės duomenimis, geriausiai tinkantis šifravimo algoritmas yra AES-CTR. Jis nėra saugiausias ar greičiausias, bet jis yra idealus tarpinis taškas tarp saugumo ir energijos sąnaudų eikvojimo [25].

1.7. Multimedijos failų susiejimas su vartotojo „Android“ įrenginiais

Norint apsaugoti skaitmeninio turinio failus nuo jų peržiūros bet kuriuose „Android“ įrenginiuose, reikia juos susieti su vartotojo įrenginiais. Tam reikalingas užšifruotas skaitmeninio turinio failas ir licencijų failas, kuriame laikysime multimedijos failų šifravimo raktus ir inicializacijos vektorių.

Rakto dedamosios licencijų failo užšifravimui/iššifravimui:

„Android_ID“ – tai identifikacinis įrenginio 64 bitų sugeneruotas kodas, kuris sugeneruojamas įrenginiui pirmą kartą įsijungus. Kodas pasikeičia įrenginiui atstačius gamyklinius parametrus [26].

IMEI (*International Mobile Equipment Identities*) – 15–17 skaičių unikalus kodas, skirtas identifikuoti mobilųjį įrenginį [27]. Įrenginiai, kurie neturi *GSM* modulio, bet turi belaidžio interneto modulį arba tai yra muzikiniai grotuvai, šio numerio neturi, nes pas juos nėra įdiegta skambinimo funkcija [26].

Serial – identifikacinis išmaniojo įrenginio kodas, skirtas įrenginiams, kurie neturi IMEI/MEID ar ESN kodo. Taip pat gali pasitaikyti atveju, kad kai kurie įrenginiai turi ir IMEI ir Serial kodą. [26].

1.8. Siekiamo sprendimo apibrėžimas

Šiuo darbu siekiama sukurti skaitmeninio turinio apsaugos sistemą, kuri gali veikti be interneto prieigos, bei turėtu galimybę dalintis failais tarp „savo“ įrenginių.

1.9. Analizės išvados

Remiantis 1 lentelės duomenimis, nei viena STTV sistema pilnai neatitinka iškeltų reikalavimų. Visos sistemos šiuo metu nėra „nulaužtos“, bet nei viena neleidžia perkelti skaitmeninio turinio failų į kitus to paties vartotojo įrenginius. STTV sistemų gamintojai neatskleidžia, kokiu principu veikia jų apsaugos sistemos. Daroma prielaidą, kad taip elgiamasi STTV sistemos saugumo labui, nes žinant veikimo principą, pasidaro daug lengviau jį apeiti. Telieka tik spėlioti, kokiu principu yra apsaugoti multimedijos failai. Todėl bus kuriama sistema, kuri apims visus išsikeltus tikslus: „Android“ OS palaikymas, veikimas be interneto prieigos ir galimybė perkelti parsisiųstą vaizdo failą į kitą „savo“ įrenginį.

Skaitmeninio turinio vandenženklio žymėjimas yra metodas, kuriam nėra būtina interneto prieiga. Ar šiuo metodu veikia analizuotos sistemos, nėra žinoma. Šiam metodui įgyventi reikia į vaizdo failą pridėti vandenženklį, taip pat reikia pritaikyti vaizdo failo grotuvą iššifruoti vaizdo failą. Todėl šis metodas nepasirinktas, nes jis yra žymiai sudėtingesnis ir sunkiau įgyvendinamas, nei vaizdo failo užšifravimas, tuo pat metu nesuteikdamas jokios papildomos apsaugos.

Licencijų failui ir multimedijos failams apsaugoti geriausiai tinka AES-CTR šifravimo metodas, nes jis geriausiai balansuoja tarp saugumo ir energijos sąnaudų iššifruojant failus (žr. 4 lentelė)

Apsaugoti „Android“ mobiliąją programėlę nuo apgrąžos inžinerijos pasirinkta „ProGuard“ apsauga, nors ji nusileidžia savo galimybėmis mokamiems produktams, bet yra nemokama (žr. 2 lentelė).

2. SKAITMENINIO TURINIO TEISIŲ VALDYMO NEREIKALAUJANČIOS INTERNETO PRIEIGOS METODO KŪRIMAS

Analizės dalyje nustatyta, kad visos sistemos gali veikti be interneto prieigos. Bet jos neturi vieno funkcionalumo – galimybės perkelti vaizdo failo į kitus vartotojo įrenginius. Šiame skyriuje aprašomi funkciniai ir nefunkciniai reikalavimai, skirti projektuojamai sistemai. Toliau bus projektuojama sistema, kuri turėtų galimybę perkelti vaizdo failus į kitą „savo“ vartotojo įrenginį ir veiktų be interneto prieigos.

2.1. Reikalavimų specifikavimas

2.1.1. Funkciniai reikalavimai

Kuriamos sistemos funkciniai reikalavimai:

1. mobilioji programėlė turi leisti vaizdo įrašą persukti;
2. mobilioji programėlė turi leisti peržiūrėti tik prie programėlės prisijungusio vartotojo vaizdo įrašus;
3. mobilioji programėlė turi atvaizduoti neužšifruotus vaizdo failus;
4. mobilioji programėlė turi turėti prisijungimų formą.

2.1.2. Nefunkciniai reikalavimai

Kuriamos sistemos nefunkciniai reikalavimai:

1. telefono vartotojas negali turėti administratoriaus teisių.
2. mobiliąją programėlę gali įrašyti tik į telefono atmintį.
3. vartotojams reikalingi prisijungimo duomenys, kad galėtų prisijungti prie programėlės.
4. programa veikia „Android“ 6.0.0 arba naujesnėje operacinėje sistemoje.
5. mobilioji programėlė turi paleisti vaizdo įrašą ne ilgiau kaip per 60 sekundžių.

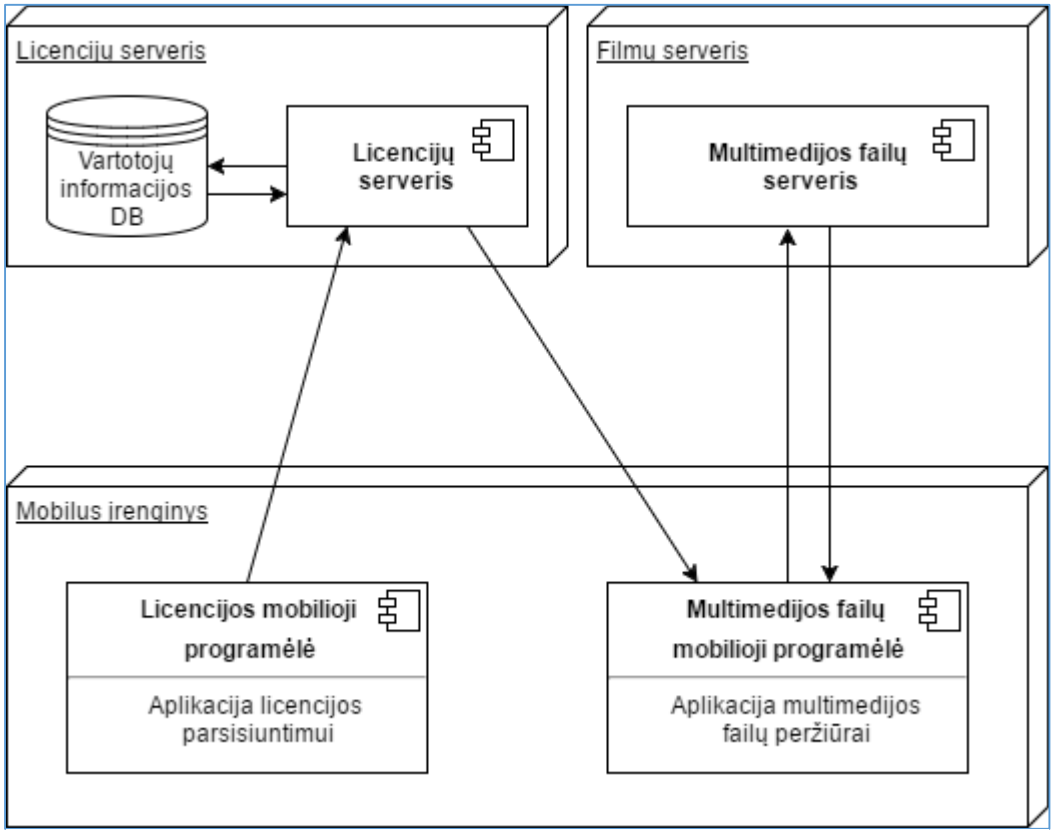
2.2. Skaitmeninio turinio teisių valdymo sistemos prototipo kūrimas

STTV sistemos prototipo kūrimo dalyje pateiktos sistemos ir jos dalių specifikacijos *UML* kalboje. Su *UML* diagramomis galima tiksliau pažvelgti į sistemos komponentus, veiklas, valdymą, panaudojimą ir atskirų projekto dalių komunikaciją. Tam naudojama:

a) išdėstymo diagrama;
b) duomenų bazės diagrama;
c) veiklos diagramas.

2.2.1. Skaitmeninio turinio teisių valdymo prototipo sistemos architektūra

Šioje diegimo diagramoje parodyta, sistemos techninės ir programinės dalies išdėstymas (žr. 8 pav.) :



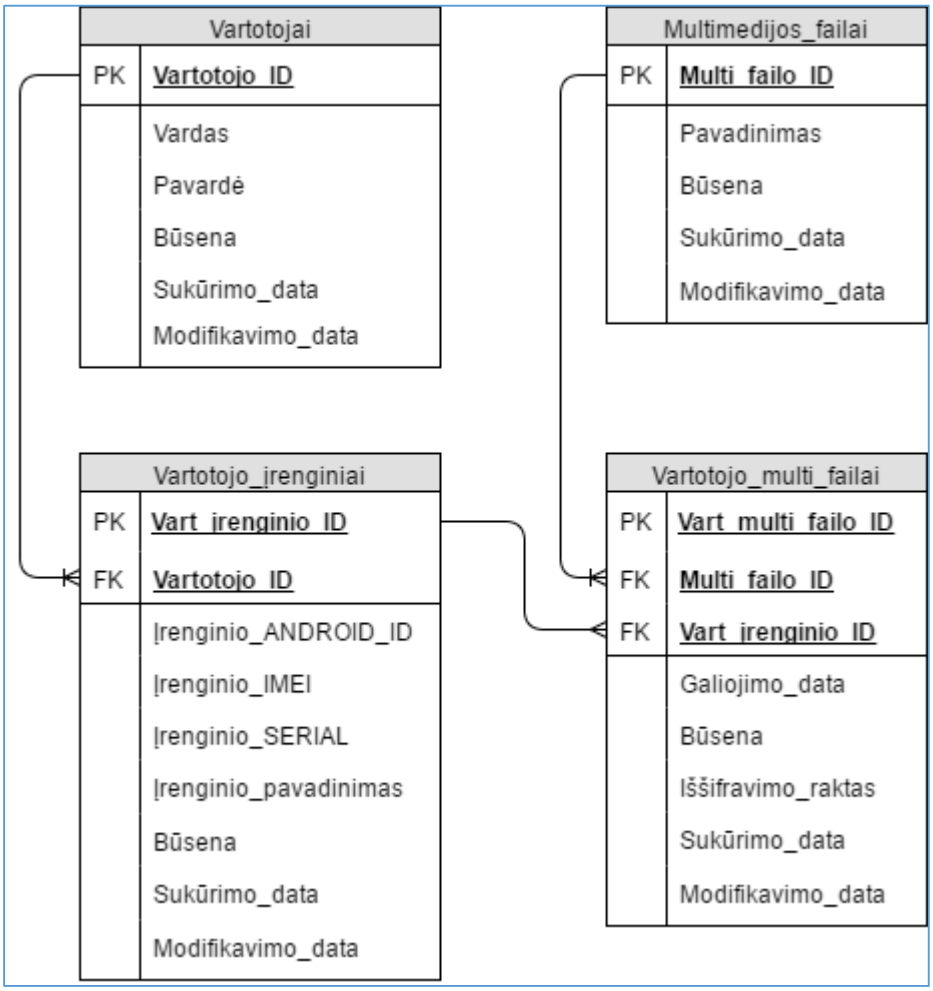
8 pav. Diegimo (angl. *Deployment*) diagrama

Diegimo diagrama (žr. 8 pav.) susideda iš trijų įrenginių: licencijų serverio, filmų serverio ir mobiliojo įrenginio. Pirmiausia, vartotojas turės atsisiųsti „Licencijos mobiliąją programėlę“ ir joje prisijungti. Mobilioji programėlė duos komandą „Licencijų serveriui“ sugeneruoti ir atsiųsti į tą patį vartotojo įrenginį „Multimedijos failų mobiliąją programėlę“, kurios viduje yra to vartotojo filmų licencijos. Instaliavęs antrąją mobiliąją programėlę ir prie jos prisijungęs, vartotojas galės pasirinkti iš savo esamų multimedijos failų arba parsisiųsti naujų iš „Multimedijos failų serverio“. Atsisiuntus ir paleidus multimedijos failą, „Multimedijos failų mobilioji programėlė“ patikrins turimas vartotojo licencijas ir, jei tokią vartotojas turės, pradės rodyti multimedijos failo turinį.

Kartą atsisiuntęs multimedijos failą, vartotojas gali jį peržiūrėti, nepaisant ar turi šiuo metu interneto prieigą, ar ne.

2.2.2. Licencijų serverio duomenų bazės architektūra

9 paveikslėlyje parodyta licencijų serverio „Vartotojų informacijos DB“ duomenų bazės diagrama:

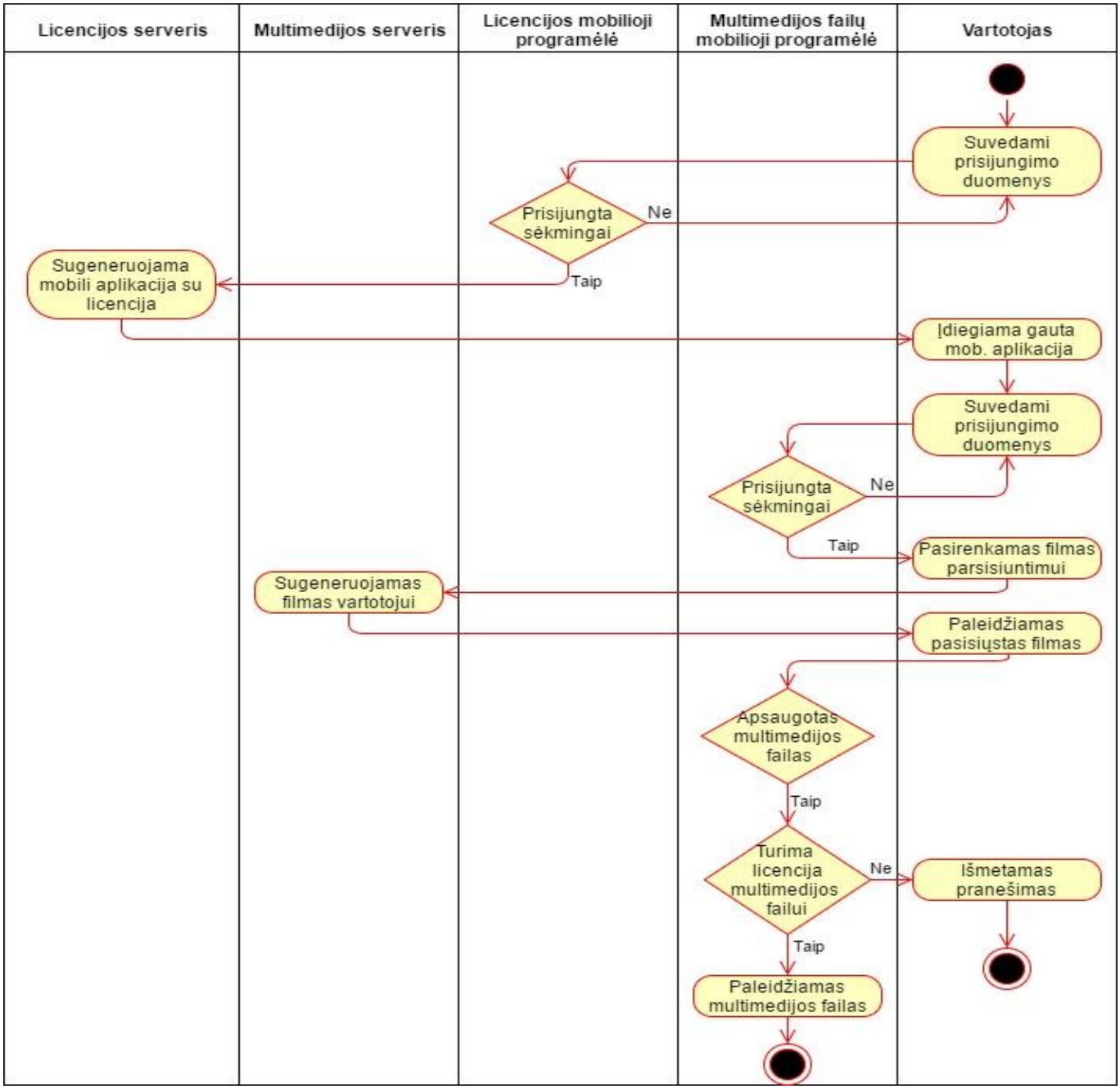


9 pav. Licencijos serverio duomenų bazės architektūra

Duomenų bazės schemoje matyti keturios lentelės (žr. 9 pav.). Jose saugomi vartotojų duomenys, vartotojo įrenginių duomenys, kokie egzistuoja multimedijos failai sistemoje bei kokius multimedijos failus vartotojai jau turi nusipirkę. DB skirta, kad generuojant naują licencijų failą, vartotojui nereikėtų sugeneruoti naujų multimedijos failų raktų. Vartotojo „Savo“ įrenginiai yra laikomi, tam vartotojui priskirti įrenginiai duomenų bazėje.

2.2.3. Skaitmeninio turinio teisių valdymo prototipo veiklos diagramos

10 paveikslėlyje pateiktoje veiklos diagramoje pavaizduotas sistemos programinių komponentų veikimas:



10 pav. Pilnos STTV sistemos veiklos diagrama

Veiklos diagramoje (žr. 10 pav.) parodyta viena iš pagrindinių sistemos dalių yra „Multimedijos failų mobilioji programėlė“. Joje pavaizduota, ar vartotojas turi licenciją multimedijos failui. Daugiau apie šį tikrinimą (žr. 11 pav.).

Šioje veiklos diagramoje pateiktas mechanizmas, kaip tikrinama ar vartotojas turi multimedijos failo licenciją (žr. 11 pav.):

11 pav. Licencijos patikrinimas veiklos diagrama

Veiklos diagramoje (žr. 11 pav.) pavaizduota pagrindinė „Multimedijos failų mobilioji programėlė“ funkcionalumo dalis. Pirmiausia, pagal vartotojo mobilaus įrenginio duomenis, yra sugeneruojamas licencijos raktas. Todėl tik iš vieno įrenginio galima iššifruoti licencijų failą. Inicializacijos vektorius yra išsaugotas programėlės viduje. Su sugeneruotu raktu ir inicializacijos vektoriumi yra iššifruojamas licencijų failas į laikinąją atmintį. Patikrinama, ar faile egzistuoja licencija ir, ar ji galioja paleidžiamam multimedijos failui. Jeigu licencijos nėra, vartotojas gauna pranešimą, kad licencijos neturi, o jeigu ji egzistuoja, paleidžiamas multimedijos failas.

2.2.4. Vaizdo failų užkodavimas

Vaizdo failų užšifravimas yra viena svarbiausių sistemos dalių. Jie yra automatiškai užšifruojami serveryje (žr. 12 pav.):

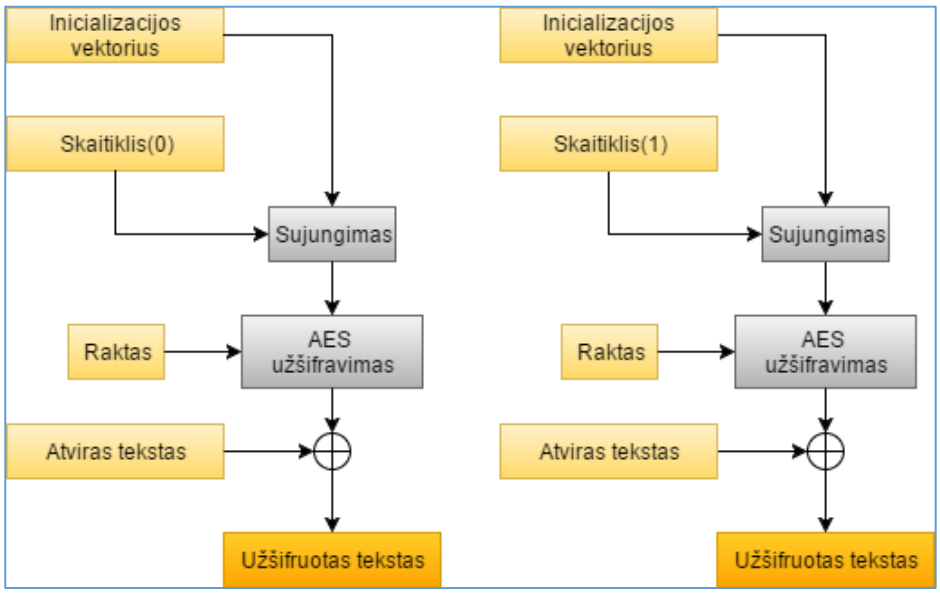
12 pav. Multimedijos failo užšifravimo schema

Proceso pabaigoje gautas *Common File Format* (*CFF*) failas. Šis formatas naudoja *ISO MPEG* failo formato pagrindą. *CFF* vaizdo medžiagos šifravimui naudoja *H.264/AVC (ISO/IEC 14496-15)* ir *H.265/HEVC (ISO/IEC 23008-2)*. *CCF* palaiko garso formatą: *stereo MPEG-4 AAC LC audio (ISO/IEC 14496-3)*. Vaizdo faile gali būti vaizdo medžiaga nuo *SD* (640x480) iki *UHD* (3840x2160) rezoliucijos.

Vaizdo failo užšifravimui naudojama *AES-256 CTR* skaitiklio (angl. *CounTeR*) metodą su 16 baitų blokais. *CRT* šifravimo metodas pasirinktas, nes jis gali failą šifruoti iš bet kurios vietos, t. y. pradžios ar vidurio. Ši savybė yra labai svarbi, nes turi būti galimybė vaizdo failą peržiūrėti nuo bet kurios vietos.

AES-CRT veikimo schemos:

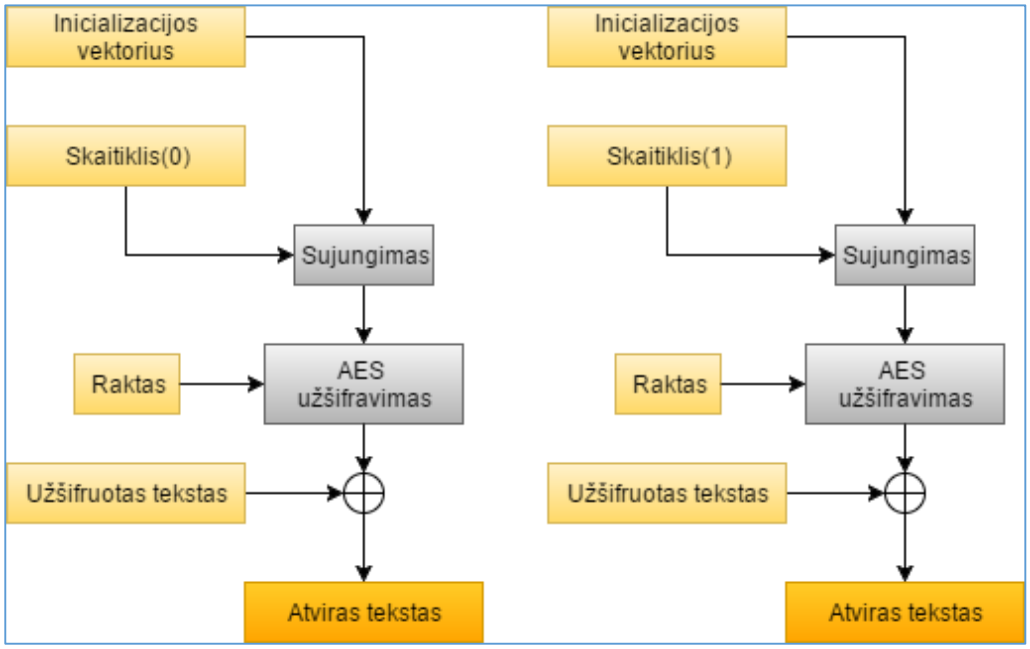
13 pav. pateikta AES-CTR užšifravimo schema.



13 pav. AES-CTR užšifravimo schema

AES-CTR skaitiklio (angl. *counter*) režimo veikimas sudarytas iš inicializacijos vektoriaus ir skaitiklio sujungimo į vieną bloką. Toliau blokas yra užšifruojamas raktu. Šį bloką sujungus su atviro teksto bloku ir panaudojus loginę operaciją *XOR*, gaunamas užšifruotas tekstas [28].

14 pav. pateikta AES-CTR iššifravimo schema.



14 pav. AES-CTR iššifravimo schema

Užšifruotas tekstas iššifruojamas, taip pat kaip ir užšifruojamas.

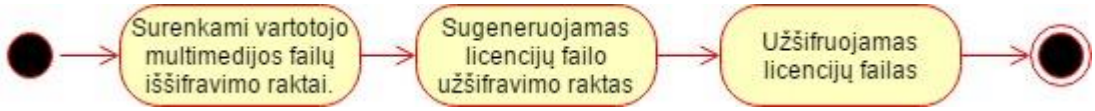
Vaizdo failo iššifravimas:

Atidarius multimedijos failą, iššifruoti duomenys išsaugomi laikinoje atmintyje (RAM). Taip apsisaugoma nuo duomenų atkūrimo iš pastoviosios atminties. Laikinojoje atmintyje yra laikomi tik tie iššifruoti duomenys, kurių prireiks artimiausiu metu. Pavyzdžiui, žiūrimas vaizdo įrašas, o tuo metu į laikiną atmintį iššifruojama ne daugiau vienos minutės į priekį vaizdo įrašas, o seni duomenys, kurie buvo peržiūrėti, išvalomi iš laikinos atminties, kad jos neapkrauti.

2.2.5. Licencijos failo užšifravimas

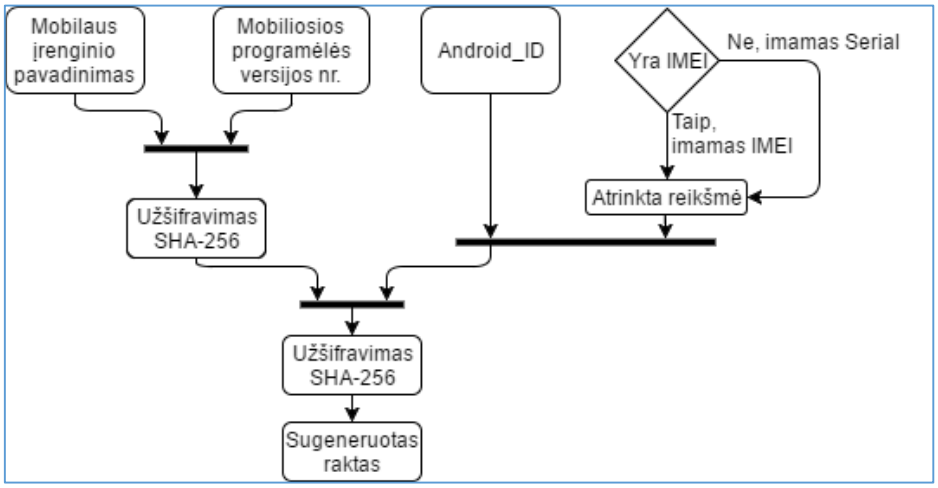
Multimedijos licencijų failo užšifravimas vykdomas serveryje (žr. 15 pav.).

Multimedijos licencijų failo užšifravimo schema:



15 pav. Multimedijos licencijų failo užšifravimo schema

Multimedijos licencijų failas generuojamas „Licencijų serveryje“. Į jį sudedami vartotojo duomenys apie jo turimus multimedijos failus ir failų raktus bei jų galiojimo laikas. Pagal mobiliojo įrenginio duomenis sugeneruojamas 256-bitų raktas ir juo užšifruojamas failas. Kiekvieną kartą šifruojant licenciją, yra sugeneruojamas naujas raktas. Rakto sudarymo schema (žr. 16 pav.).



16 pav. Multimedijos failo rakto sudarymo schema

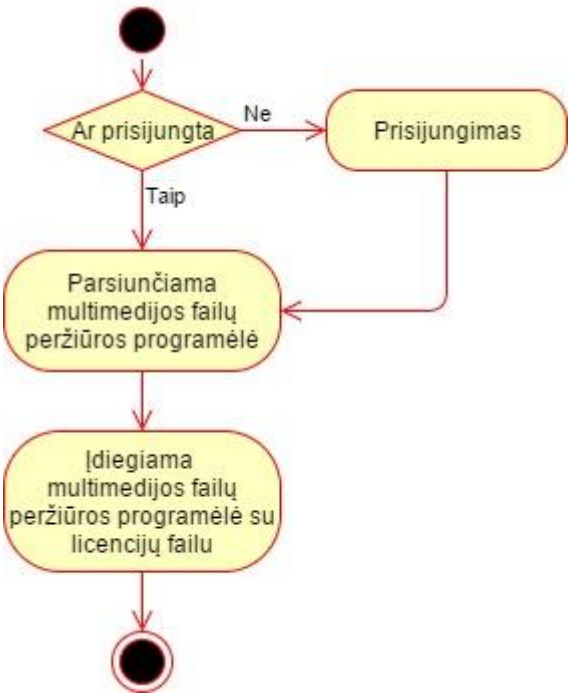
Raktas licencijos failui sudarytas iš sujungtų į vieną tekstą įrenginio „Android_ID“, *Serial* arba *IMEI* numerio (priklauso, kurį numerį turės įrenginys) ir įrenginio modelio pavadinimo, sujungto su mobiliosios programėlės versijos numeriu, kurie užšifruoti *SHA-256* maišos algoritmu. Visa gauta reikšmė užšifruota *SHA-256* algoritmu.

Raktas vaizdo failo užšifravimui/iššifravimui:

Raktas multimedijos failui sudarytas iš atsitiktinai sugeneruotų simbolių. Rakto ilgis 256 bitai. Multimedijos failą su licencijomis iššifruoti galės tik mobilioji programėlė, kuris vartotojui nėra prieinama. Daugiau apie šio failo iššifravimą (žr. 11 pav.).

2.2.6. Licencijų failo perdavimas

Licencijų failo perdavimo etapas, tai vienintelė vieta, kai vartotojui reikia interneto prieigos (žr. 17 pav.). Multimedijos licencijų failas atsiunčiamas kartu su „Licencijų failo mobiliąja programėle“. Atsinaujinus licencijai, t. y. nusipirkus naują filmą, reikia iš naujo atsisiųsti visą programėlę, taip užtikrinant apsaugą nuo licencijos perėmimo, kol ją parsiųs į mobilųjį įrenginį.



17 pav. Licencijų failo perdavimas vartotojui

Šiame etape vartotojas gauna pirmąją mobiliąją programėlę, skirtą parsisiųsti antrai mobiliajai programėlei, kuri skirta multimedijos peržiūrai. Vartotojas, atidaręs pirmąją programėlę ir prisijungęs prie savo paskyros gautais prisijungimo duomenimis iš prototipo sistemos, turi galimybę parsisiųsti antrąją programėlę. Parsisiuntus antrąją programėlę, vartotojui nereikės rūpintis licencijomis, nes jos bus programėlės viduje. Tokiu būdu vartotojas licencijos failo negali nei matyti, nei perkelti, nei jo redaguoti. Vartotojui telieka programėlę įdiegti į savo mobilųjį įrenginį ir peržiūrėti multimedijos turinį.

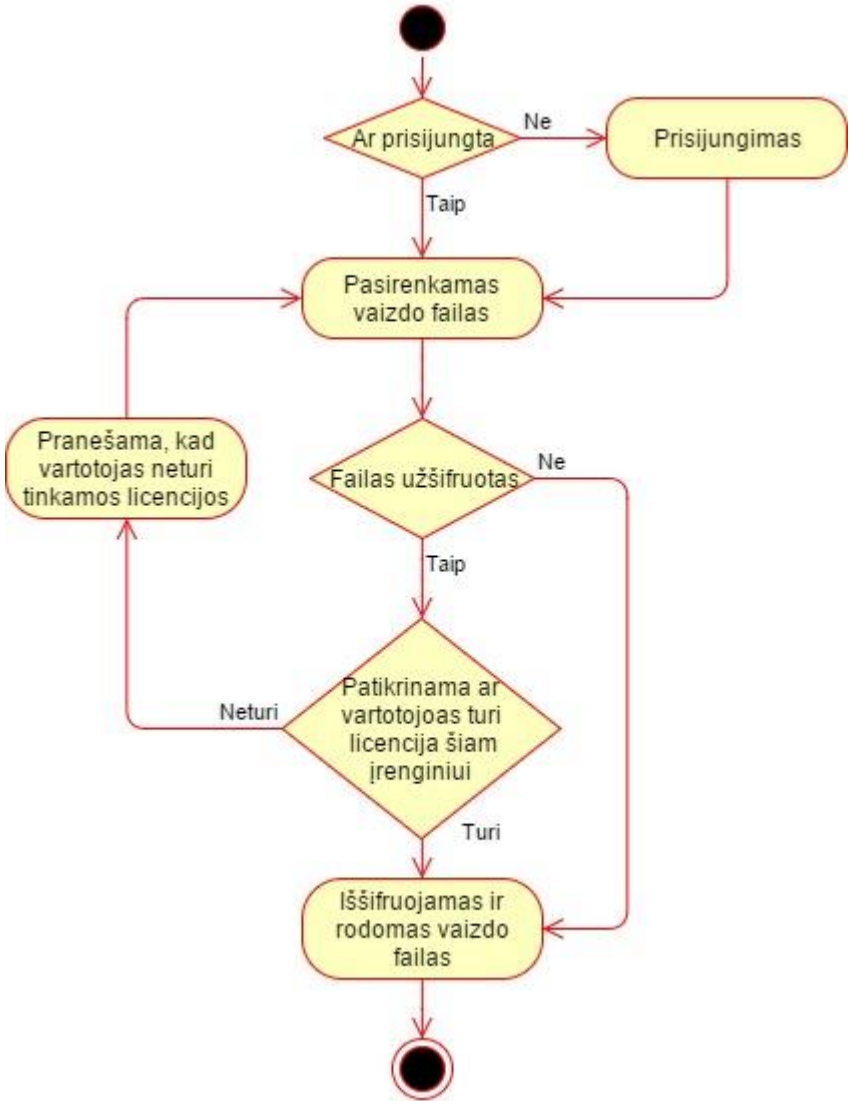
2.2.7. Licencinio failo nuskaitymas

Licencinis failas vartotojui nėra pasiekiamas. Jį gali atidaryti tik mobilioji programėlė.

Paleidus multimedijos failų peržiūros mobiliąją programėlę, vartotojui reikės prisijungti su savo paskyra. Prisijungus prie mobiliosios programėlės, pasirenkamas multimedijos failas. Pasirinkus multimedijos failą, mobilioji programėlė iššifruoja licencijos failą į laikinąją atmintį. Iššifravus licencijos failą, patikrinama, ar jame yra tinkamas multimedijos failo įrašas. Radus įrašą, paimamas iššifravimo raktas. Toliau vyksta multimedijos failo paleidimas (žr. 2.2.8 skyrius).

2.2.8. Vaizdo failo paleidimas

Vartotojas, kuris yra nusipirkęs vaizdo failą, gali jį atsisiųsti iš serverio ir peržiūrėti savo mobiliajame įrenginyje. Šiame paveikslėlyje pateikiama multimedijos failų paleidimo schema (žr. 18 pav.):



18 pav. Prototipo veikimo principas

Prototipo veikimas prasideda nuo multimedijos failų peržiūros programėlės paleidimo. Ją paleidus, vartotojas turi prisijungti arba pasinaudoti anoniminiu vartotoju, jeigu nori peržiūrėti

nešifruotą vaizdo medžiagą. Prisijungęs vartotojas gali peržiūrėti dviejų tipų multimedijos failus: šifruotus ir nešifruotus. Vartotojui prisijungus ir pasirinkus vaizdo failą, kurį parsisiuntė iš multimedijos failų serverio arba jį įsikėlė, sistema patikrina ar failas yra užšifruotas ir ar vartotojas turi teisę jį peržiūrėti. Jeigu vartotojas turi tokią teisę, tai vaizdo failas yra iššifruojamas ir rodomas. Kadangi vaizdo medžiaga yra rodoma ir tuo pat metu iššifruojama, vartotojas paleidęs vaizdo failą jį mato iš karto (priklausomai nuo telefono spartos).

2.2.9. Mobiliosios programėlės apsaugojimas

Viena iš lengviausių atakų, kurią gali padaryti prieš visas *Android* mobiliąsias programėles, tai panaudoti apgrąžos inžinerijos (angl. *reverse engineering*) metodą. Šis metodas skirtas iš sukompiliuotos mobiliosios programėlės išgauti pradinį (angl. *source*) kodą. Šios sistemos pagrindinė vieta yra multimedijos failų peržiūros mobilioji programėlė, kadangi joje saugomas slaptas raktas ir inicializacijos vektorius, kurie iššifruoja multimedijos licencijų failą. Tam, kad apsisaugoti nuo atakų ar apsunkinti pasinaudojimą šiuo metodu, yra pasitelkiama „ProGuard“ programa (žr. 2 lentelė). Ši programa pakoreguoja parašytą kodą, t. y. pervadina kintamuosius, klasių pavadinimus, reikšmes, taip pasunkindama mobiliosios programėlės veikimo analizę.

3. SKAITMENINIO TURINIO TEISIŲ VALDYMO SISTEMOS ATJUNGTIES REŽIMU PROTOTIPO REALIZACIJA IR TYRIMAS

Šiame skyriuje aprašoma, kaip yra įgyvendinta mobilios programėlės realizacija, pateikiama kodo pavyzdžių. Taip pat aprašomi tyrimai, atliekami su mobiliąja programėle. Ar veikia pasiūlytas metodas, kad vartotojas galės dalintis savo multimedijos failu tarp „savo“ įrenginių, ar sudėtinga apeiti mobiliosios programėlės apsaugą ir, ar pasiteisina pasirinktos apsaugos priemonės.

3.1. Skaitmeninio turinio apsaugos atjungties režimu prototipo realizacija

Prototipo realizacijai įgyvendinti panaudota programėlė „Notes“. Patobulintas mobiliosios programėlės išeities (angl. *source*) kodas:

1. integruojamas licencijos failas su multimedijos raktais;
2. integruojamas licencijos iššifravimo rakto generavimas;
3. integruojama procedūra licencijų iš licencijos failo išgauti;
4. integruojama „ProGuard“ apsauga;
5. sukompiliuojama mobilioji programėlė.

Patobulintą mobiliąją programėlę apsaugojama pasirinkta apsaugos programa „ProGuard“. Kaip apsaugoti mobiliąją programėlę, parodyta paveikslėlyje (žr. 19 pav.).

```
buildTypes {
    release {
        minifyEnabled true
        buildConfigField "String", "hiddenSalt", "\"${hiddenSalt}\""
        buildConfigField "String", "hiddenIV", "\"${hiddenIV}\""
    }
```

19 pav. „ProGuard“ apsaugos įjungimas

Norint įjungti „ProGuard“ apsaugą, reikia įrašyti „minifyEnabled true“ eilutę. Sukompiliavus mobiliosios programėlės failą jis bus apsaugotas „ProGuard“ apsauga.

3.2. Skaitmeninio turinio apsaugos atjungties režimu prototipo tyrimas

Prototipo realizacijos tyrimas susideda iš galimybės peržiūrėti multimedijos failą atjungties režimu savo įrenginyje, keliuose savo įrenginiuose, draudimo peržiūrėti „svetimame“ įrenginyje, t. y. įrenginyje, kuris nesusietas su prisijungusio vartotojo mobiliąją programėle ir programėlės apsaugojimu nuo apgrąžos, inžinerijos.

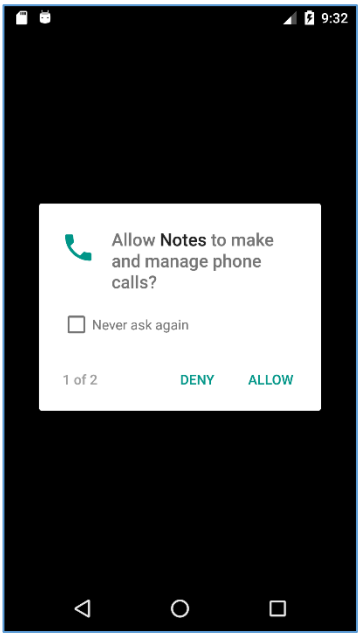
Tyrime sukurta svarbiausia sistemos dalis – licencijos patikrinimas (žr. 11 pav.), kuri atsakinga už tai, ar vartotojo programėlė atjungties režimu gali iššifruoti licencijos failą, ar ne. Jeigu vartotojui iššifruojamas licencijos failas, vadinasi, programėlė galės atverti multimedijos failus, todėl

multimedijos iššifravimo dalis nebuvo kuriama. Visi tyrimai, kuriuose bandomas mobiliosios programėlės funkcionalumas, atlikti atsijungus nuo interneto.

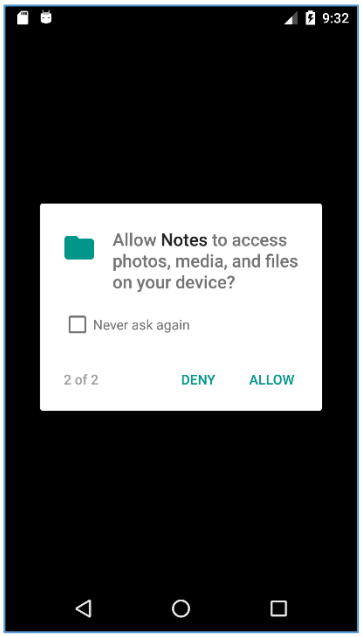
3.2.1. Multimedijos failo peržiūra „savo“ įrenginyje

Šioje dalyje tikrinama, ar sukurta mobilioji programėlė iššifruoja vartotojo licenciją su multimedijos raktais. Mobilioji programėlė, testuojama kompiuteryje, sukurtame virtualiame „Android“ įrenginyje „Nexus 5“ su API 23, dar kitaip vadinama „Android 6.0“.

Įdiegta mobilioji programėlė pirmiausia vartotojo paprašo prieigos prie mobilaus įrenginio techninių duomenų ir duomenų saugyklos (žr. 20, 21 pav.):



20 pav. Techninių duomenų teisė

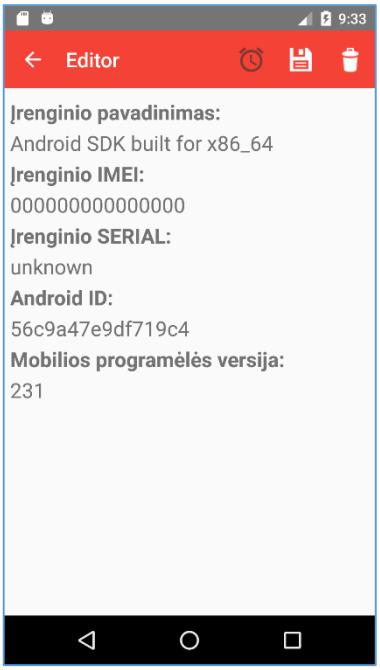


21 pav. Duomenų saugyklos teisė

Suteikus mobiliajai programėlei teises, mobilioji programėlė patikrina, ar failas yra su licencijomis. Radus šį failą, pagal mobilaus įrenginio techninius duomenis ir mobiliosios programėlės versiją, sugeneruojamas iššifravimo raktas. Iššifravus licencijų failą, mobilioji programėlė patikrina, ar gauti duomenis atitinka failo struktūrą. Failo struktūrai atitikus – atvaizduojami licencijos failo duomenys.

Failų struktūros rezultatai pavaizduoti paveikslėliuose (žr. 22, 23 pav.):

22 pav. Licencijų failo duomenys



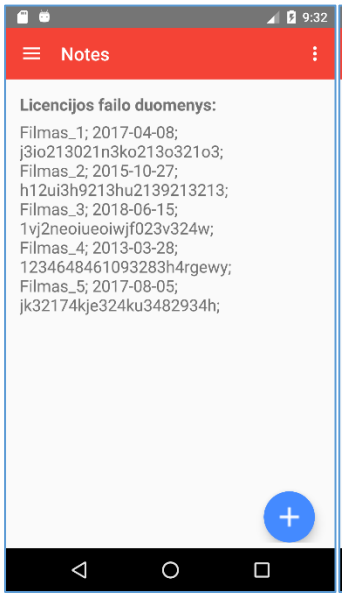
23 pav. Mobilaus įrenginio techniniai duomenys

Mobiliajai programėlei sėkmingai pavyko iššifruoti licencijų failą (žr. 22 pav.). Taip pat galima peržiūrėti mobiliojo įrenginio techninius duomenis ir iš kokių kintamųjų susideda licencijų failo raktas (žr. 23 pav.). Ši galimybė galutinėje mobiliojoje programėlėje nėra numatyta, ji skirta tik testavimui.

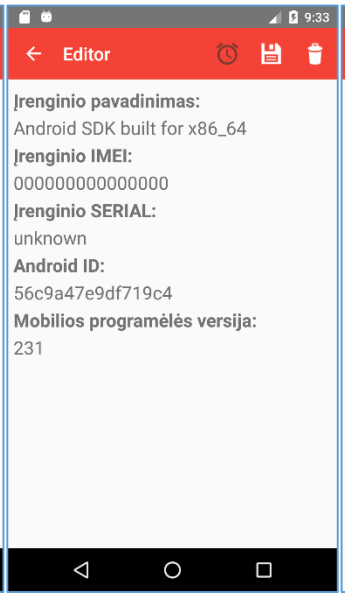
Išvada: šio tyrimo tikslas pasiektas ir gautas rezultatas parodo, kad mobilioji programėlė sėkmingai iššifruoja ir atidaro licencijų failą.

3.2.2. Multimedijos failo peržiūra „savo“ įrenginiuose

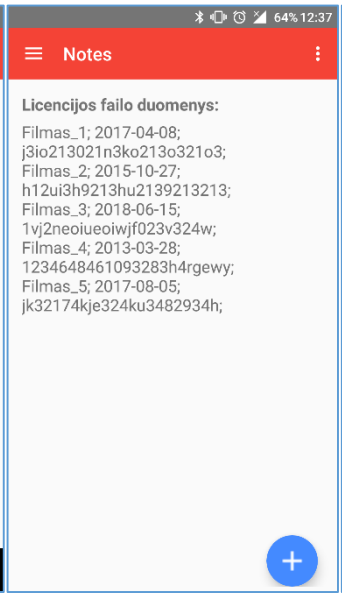
Šioje dalyje, tikrinimui – sukurtos dvi mobiliosios programėlės „Note_one“ ir „Note_pc“. Informaciją apie „Note_pc“ galime rasti aprašyta 3.2.1 skyriuje. Viena mobilioji programėlė yra skirta tik vienam, tam tikram įrenginiui. „Note_one“ – skirta veikti tik mobiliajame įrenginyje „OnePlus 3“ su API 25 „Android 7.1“. Abi mobiliosios programėlės yra priskirtos vienam vartotojui, bet skirtingiems jo įrenginiams, todėl licencijų failas abejose programėlėse yra vienodas.



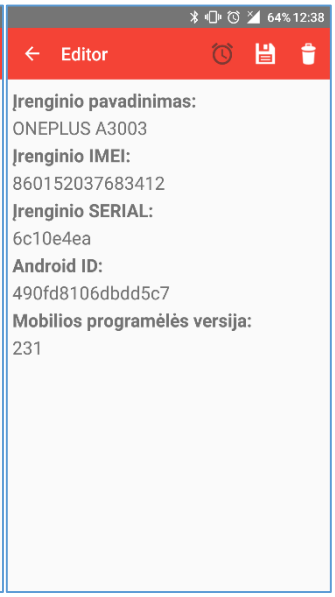
24 pav. „Note_pc“ mobilaus įrenginio techniniai duomenys



25 pav. „Note_pc“ licencijų failo duomenys



26 pav. „Note_one“ mobilaus įrenginio techniniai duomenys



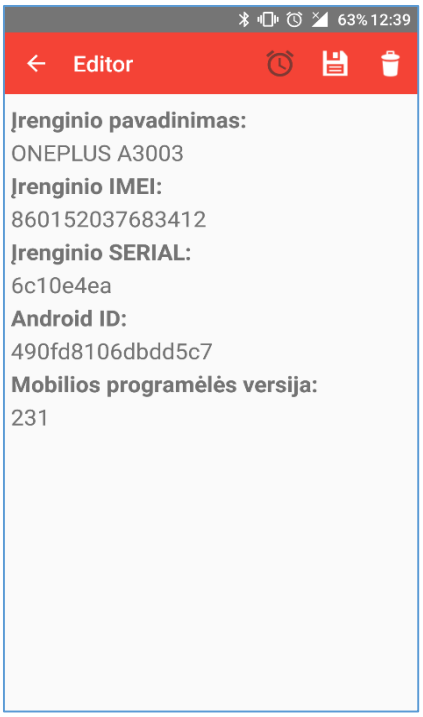
27 pav. „Note_one“ licencijų failo duomenys

Paveikslėliuose (žr. 24 pav., 25 pav., 26 pav., 27 pav.) pavaizduota, kad licencijų failą pavyko sėkmingai iššifruoti abiem įrenginiams, nors jų techninės charakteristikos yra skirtingos.

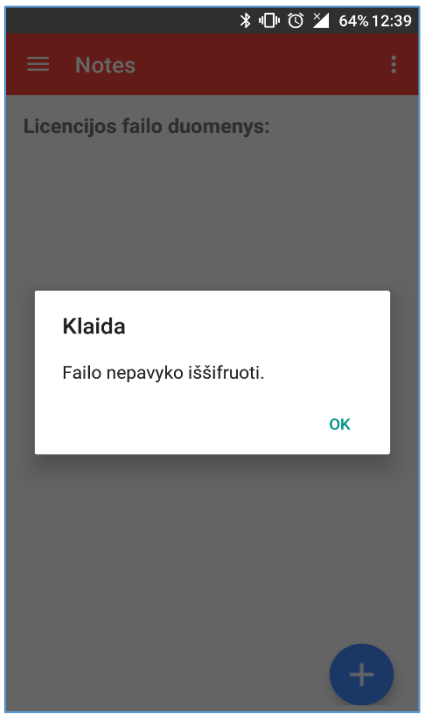
Išvada: šio tyrimo tikslas pasiektas, ir gautas rezultatas parodo, kad vartotojas gali savo licencijų failą naudoti ir kituose įrenginiuose, bet tam reikia ir kitos mobiliosios programėlės, kurią sugeneruoja „Licencijų serveris“, „Multimedijos failų mobilioji programėlė“ dalis (žr. 9 pav.).

3.2.3. Multimedijos failo peržiūra „svetimame“ įrenginyje

Šioje dalyje tikrinama, ar vartotojo mobiliąją programėlę įrašius į ne „savo“ įrenginį, licencijų failas bus iššifruotas. Tam naudota „Note_pc“ mobilioji programėlė ir ji įrašyta į „OnePlus 3“ įrenginį (žr. 29 pav.).



28 pav. Mobilaus įrenginio techniniai duomenys



29 pav. Netinkamas licencijų failo raktas

Mobiliajai programėlei nepavyko iššifruoti licencijų failo, kuris skirtas kitam įrenginiui.

Išvada: šio tyrimo tikslas pasiektas, ir gautas rezultatas parodo, kad kitame mobiliajame įrenginyje mobilioji programėlė tinkamai neveikia ir neatvaizduoja multimedijos įrašo.

3.2.4. Mobilios programėlės apsaugojimas nuo apgrąžos inžinerijos

Prototipo standartiškai „Android“ OS siūlomos apsaugos atsparumui patikrinti, panaudotas apgrąžos inžinerijos metodas. Jis atliktas keliais skirtingais būdais:

1. išrinkimo (angl. *disassembler*);
2. dekompiliavimo (angl. *decompiler*).

Apgrąžos inžinerijos metodais tiriama, ar iš sukompiliuotos mobiliosios programėlės galima išgauti pirminį kodą. Ar išgautame pirminiame kode galima rasti rakto sugeneravimo vietą, ar galima rasti inicializacijos vektorių ir ar failų, aplankalų struktūra atitinka originalią. Šis tyrimas atliktas

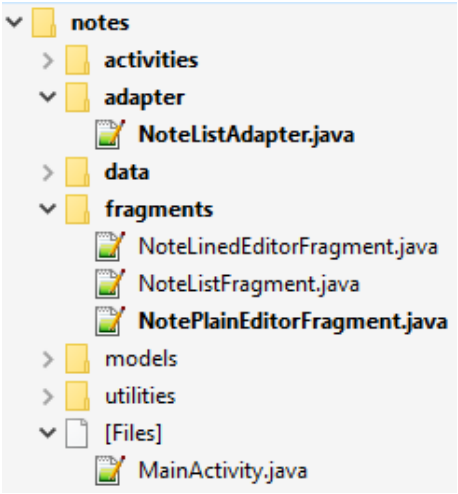
žmogaus turinčio tris metus programavimo patirties. Pirmiausia, ieškoma vienos iš pagrindinių funkcijų raktams generuoti, pavadinimu *generateKey*. Originali funkcija atrodo taip (žr. 30 pav.):

```
private SecretKey generateKey(String pass) throws NoSuchAlgorithmException, InvalidKeySpecException {
    int iterationCount = 1000;
    int keyLength = 256;

    byte[] salt = BuildConfig.hiddenSalt.getBytes();

    KeySpec keySpec = new PBEKeySpec(pass.toCharArray(), salt, iterationCount, keyLength);
    SecretKeyFactory keyFactory = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA1");
    byte[] keyBytes = keyFactory.generateSecret(keySpec).getEncoded();
    SecretKey sks = new SecretKeySpec(keyBytes, "AES/CTR/NoPadding");

    return sks;
}
```

30 pav. Rakto generavimo funkcija

Šioje funkcijoje pateikta (žr. 30 pav.), kaip ir kokiomis funkcijomis iš gauto slaptažodžio *pass* yra sugeneruojamas slaptas raktas *sks*.

Originali aplankalų, failų struktūra atrodo taip (žr. 31 pav.):



31 pav. Aplankalų architektūra

Mobiliosios programėlės pagrindinės funkcijos yra išsaugotos *adapter*, *fragments* aplankaluose. *Adapter* aplankale randasi pagrindinės funkcijos, reikalingos iššifruoti licencijų failą, o *fragments* aplankale – įrenginio techninių duomenų atvaizdavimo funkcijos.

Inicializacijos vektoriaus išsaugojimo vieta (žr. 32 pav.):

```
hiddenSalt=□?z↑□???^xs□I?□??H□H,?□?□9_?
hiddenIV=AZ:□?K?/ □□$K.T-
```

32 pav. Paslėptas inicializacijos vektorius

Inicializacijos vektorius – paslėptas *gradle.properties* faile, kurio duomenys, sukompiliavus mobiliąją programėlę, perkeliami į *BiuldConfig.java* failą.

Iš šių paveikslėlių (žr. 30, 31, 32 pav.) matyti pagrindinės ieškomos dalys, kurių ieškoma tyrime. Tyrime naudojamos dvi identiškos sukompiliuotos mobiliosios programėlės, tik viena apsaugota „ProGuard“ programa, o kita ne.

3.2.4.1. Tyrimo atlikimas išrinkimo metodu

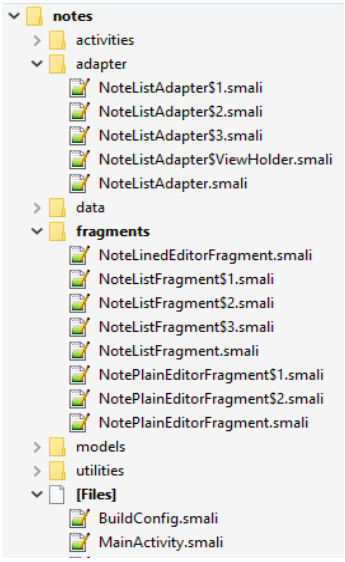
Išrinkimo metodas yra priešingas surinkimo metodui. Jis išverčia vykdomąjį kodą, šiuo atveju iš *Java* į *Assemblerio* kalbą. Šis kodas tampa skaitomas, bet jį sunku tirti.

„APK-Tool“ – paverčia sukompiliuotą mobiliąja programėlę į kodą *Smali* kalba. Pavertę į šį kodą, jo derinti (angl. *debug*) negalima, bet jį galima perskaityti.

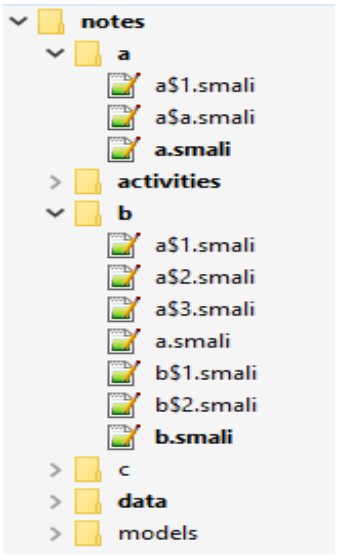
Kaip paleisti „Apk-Tool“ programą, galima rasti jos tinklalapyje [29]. Įdiegtą programą reikia paleisti per „Command Prompt“ pultą (angl. *console*). Jame nurodyti kelią iki „Apk-tool“ programos ir parinkti mobiliosios programėlės failą, kurį norimą išversti į *Smali* kalbą. Failui išversti į *Smali* kalbą panaudota komanda „apktool.bat d Notes_one_v2.apk“.

Rezultatai:

Aplankalų ir failų struktūros tyrimo rezultatai pavaizduoti paveikslėliuose (žr. 33, 34 pav.):



33 pav. „Apk-Tool“ aplankalų ir failų struktūros tyrimas be „ProGuard“ apsaugos



34 pav. „Apk-Tool“ aplankalų ir failų struktūros tyrimas su „ProGuard“ apsauga

Paveikslėlyje (žr. 33 pav.) pateikta, kaip atrodo pagrindinių failų ir aplankalų struktūra, o 34 paveikslėlyje, kaip tai atrodo apsaugota „ProGuard“ programėle. Palyginus šiuos paveikslėlius su originalia aplankalų ir failų struktūra (žr. 31 pav.), matyti, kad išgavus pradinius failus su „Apk-Tool“ programėle, failų padaugėjo. Taip pat matyti, kokiu būdu „ProGuard“ apsauga pervadina dalį aplankalų ir pagrindinius failus, kuriuose saugomas pagrindinis funkcionalumas.

Inicializacijos vektoriaus saugojamos vietos radimas (žr. 35, 36 pav.):

```
.field public static final hiddenIV:Ljava/lang/String; = "AZ:\u0017?K?/ \u0015\u001c$K.T-"
.field public static final hiddenSalt:Ljava/lang/String; = "\u0016?z\u000c\u0013???^xs\u0010I?\u0014??H\u001eH,?\u001e?\u001b9_?"
```

35 pav. „Apk-Tool“ inicializacijos vektoriaus tyrimas be „ProGuard“ apsaugos

Inicializacijos vektorius neapsaugotoje mobiliojoje programėlėje rastas *BuildConfig.smali* faile. Iš paveikslėlio (žr. 35 pav.) matyti inicializacijos vektorius, pavadinimu *hiddenIV*. Jo reikšmė, lyginant su originalu (žr. 32 pav.), atrodo skirtinga, bet taip atrodo dėl koduotės, o reali reikšmė yra tokia pati, kaip originalaus inicializacijos vektoriaus.

```
if-eqz v0, :cond_3
invoke-virtual {p0}, Lcom/donbra/notes/a/a;->d()V
const-string v0, "AZ:\u0017?K?/ \u0015\u001c$K.T-"
invoke-virtual {v0}, Ljava/lang/String;->getBytes()[B
move-result-object v2
```

36 pav. „Apk-Tool“ inicializacijos vektoriaus tyrimas su „ProGuard“ apsauga

Inicializacijos vektorius, apsaugotoje mobiliojoje programėlėje rastas tose funkcijose, kuriose yra naudojamas. Tai net palengvina analizuotojui darbą, padeda suprasti, kaip veikia programinis kodas, nes jam nereikia ieškoti, kurioje vietoje paslėptas inicializacijos vektorius.

Funkcijos paieška, kuri sugeneruota slaptą raktą pagal mobiliojo įrenginio informaciją (žr. 38, 37 pav.):

```
.method private a(Ljava/lang/String;)Ljavax/crypto/SecretKey;
    .locals 5
    const/16 v0, 0x3e8
    const/16 v1, 0x100
    const-string v2, "\u0016?z\u000c\u0013???^xs\u0010I?\u0014??H\u001eH,?\u001e?\u001b9_?"
    new-instance v2, Ljava/security/SecureRandom;
    invoke-direct {v2}, Ljava/security/SecureRandom;-><init>()V
    const-string v2, "\u0016?z\u000c\u0013???^xs\u0010I?\u0014??H\u001eH,?\u001e?\u001b9_?"
    invoke-virtual {v2}, Ljava/lang/String;->getBytes()[B
    move-result-object v2
    new-instance v3, Ljavax/crypto/spec/PBEKeySpec;
    invoke-virtual {p1}, Ljava/lang/String;->toCharArray()[C
    move-result-object v4
    invoke-direct {v3, v4, v2, v0, v1}, Ljavax/crypto/spec/PBEKeySpec;-><init>([C[BII)V
    const-string v0, "PBKDF2WithHmacSHA1"
    invoke-static {v0}, Ljavax/crypto/SecretKeyFactory;->getInstance(Ljava/lang/String;)Ljavax/crypto/SecretKeyFactory;
    move-result-object v0
    invoke-virtual {v0, v3}, Ljavax/crypto/SecretKeyFactory;->generateSecret(Ljava/security/spec/KeySpec;)Ljavax/crypto/SecretKey;
    move-result-object v0
    invoke-interface {v0}, Ljavax/crypto/SecretKey;->getEncoded()[B
    move-result-object v0
    new-instance v1, Ljavax/crypto/spec/SecretKeySpec;
    const-string v2, "AES/CTR/NoPadding"
    invoke-direct {v1, v0, v2}, Ljavax/crypto/spec/SecretKeySpec;-><init>([BLjava/lang/String;)V
    return-object v1
.end method
```

37 pav. „Apk-Tool“ funkcijos *generateKey* tyrimas su „ProGuard“ apsauga

Paveikslėlyje (žr. 37 pav.) pavaizduota *generateKey* funkcija, kurią yra sunkiau skaityti, nei *Java* kalba aprašytos funkcijos (žr. 30 pav.) dėl žemesnio lygio programavimo kalbos – *Smali*. Nepaisant žemesnio kalbos lygio ir „ProGuard“ apsaugos, pradiniame kode, galima įžvelgti, kokios yra naudojamos funkcijos sugeneruoti raktui, pvz.: *generateSecret* ar *getInstance*.

Paveikslėlyje (žr. 38 pav.) pavaizduota pradinis kodas be apsaugos:

```
.method private generateKey(Ljava/lang/String;)Ljavax/crypto/SecretKey;
    .locals 11
    .param p1, "pass"    # Ljava/lang/String;
    .annotation system Ldalvik/annotation/Throws;
        value = {Ljava/security/NoSuchAlgorithmException;,Ljava/security/spec/InvalidKeySpecException;}
    .end annotation
    .prologue
    .line 225
    const/16 v0, 0x3e8
    .line 226
    .local v0, "iterationCount":I
    const/16 v3, 0x100
    .line 227
    .local v3, "keyLength":I
    div-int/lit8 v8, v3, 0x8
    .line 228
    .local v8, "saltLength":I
    const-string v6, "\u0016?s\u000c\u0013???^xs\u0010I?\u0014??H\u001eH,?\u001e?\u001b9_?"
    .line 230
    .local v6, "s":Ljava/lang/String;
    new-instance v5, Ljava/security/SecureRandom;
    invoke-direct {v5}, Ljava/security/SecureRandom;-><init>()V
    .line 231
    .local v5, "random":Ljava/security/SecureRandom;
    const-string v10, "\u0016?s\u000c\u0013???^xs\u0010I?\u0014??H\u001eH,?\u001e?\u001b9_?"
    invoke-virtual {v10}, Ljava/lang/String;->getBytes()[B
    move-result-object v7
    .line 233
    .local v7, "salt":[B
    new-instance v4, Ljavax/crypto/spec/PBEKeySpec;
    invoke-virtual {p1}, Ljava/lang/String;->toCharArray()[C
    move-result-object v10
    invoke-direct {v4, v10, v7, v0, v3}, Ljavax/crypto/spec/PBEKeySpec;-><init>([C[BII)V
    .line 235
    .local v4, "keySpec":Ljava/security/spec/KeySpec;
    const-string v10, "PBKDF2WithHmacSHA1"
    invoke-static {v10}, Ljavax/crypto/SecretKeyFactory;->getInstance(Ljava/lang/String;)Ljavax/crypto/SecretKeyFactory;
    move-result-object v2
    .line 236
    .local v2, "keyFactory":Ljavax/crypto/SecretKeyFactory;
    invoke-virtual {v2, v4}, Ljavax/crypto/SecretKeyFactory;->generateSecret(Ljava/security/spec/KeySpec;)Ljavax/crypto/SecretKey;
    move-result-object v10
    invoke-interface {v10}, Ljavax/crypto/SecretKey;->getEncoded()[B
    move-result-object v1
    .line 237
    .local v1, "keyBytes":[B
    new-instance v9, Ljavax/crypto/spec/SecretKeySpec;
    const-string v10, "AES/CTR/NoPadding"
    invoke-direct {v9, v1, v10}, Ljavax/crypto/spec/SecretKeySpec;-><init>([BLjava/lang/String;)V
    .line 239
    .local v9, "sks":Ljavax/crypto/SecretKey;
    return-object v9
.end method
```

38 pav. „Apk-Tool“ funkcijos *generateKey* tyrimas be „ProGuard“ apsaugos

Šiame kode, taip pat kaip ir apsaugotame kode (žr. 37 pav.), pavaizduotos *generateSecret* ir *getInstance* naudojamos funkcijos.

Išvada: aplankalų, failų struktūra atkurta lyginant su originalia struktūra (žr. 31 pav.), nors failų padaugėjo, bet juose išliko ta pati informacija, kaip originale. Tiek apsaugotame pradiniame kode, tiek neapsaugotame pavyko rasti ieškotos *generateKey* funkcijos. Inicializacijos vektoriaus vieta skiriasi kode, kuris yra apsaugotas, ir kuris nėra. Apsaugotame kode inicializacijos vektorius rastas pačioje funkcijoje, o neapsaugotame *BuildConfig.smali* faile.

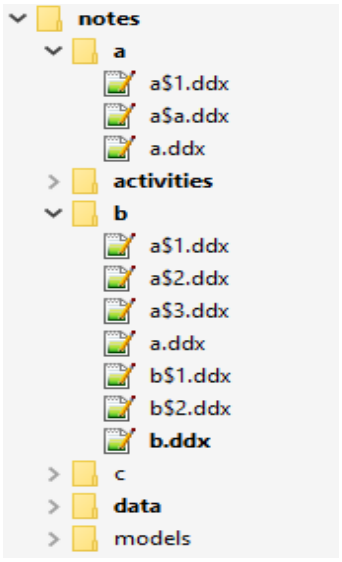
„Dedexer“ – programa iš *Dex* tipo failo suformuoja mobiliosios programėlės pradinių aplankalų struktūrą. Taip pat išrenka visus failus iš *Dex* failo ir perverčia juos *Ddx* formatu. Kaip programą parsisiųsti ir kaip ją paleisti, galima rasti jos tinklalapyje [30].

Pirmiausia parsisiunčiama „Dedexer“ programėlė ir ją sukonfigūravus pagal nurodymus [31], mobiliosios programėlės failas *Note_one_v2.apk* pakeičiamas į *Note_one_v2.zip* , pakeičiant jo tipą. Gautas failas išarchyvuojamas. Aplankale, į kurį išarchyvuotas failas, surandamas *classes.dex* failas. Paleidžiama „Command Prompt“ pultas ir jame nurodoma išarchyvuoto *classes.dex* failo vieta. Toliau

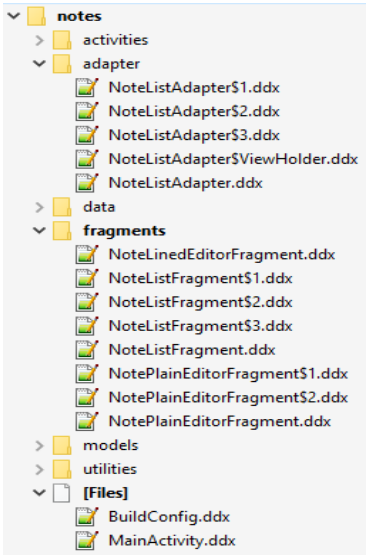
panaudojama komanda – „java -jar ddx.jar -d files classes.dex“. Po šios komandos iš *classes.dex* failo ištraukiami failai *Assemblerio* kalba.

Rezultatai:

Aplankalų ir failų struktūros tyrimo rezultatai pavaizduoti paveikslėliuose (žr. 39, 40 pav.):



39 pav. „Dedexer“ aplankalų ir failų struktūros tyrimas su „ProGuard“ apsauga



40 pav. „Dedexer“ aplankalų ir failų struktūros tyrimas be „ProGuard“ apsaugos

Paveikslėlyje (žr. 40 pav.) pavaizduota, kaip atrodo pagrindinių failų ir aplankalų struktūra, o paveikslėlyje (žr. 39 pav.), apsaugota „ProGuard“ programėle. Palyginus šiuos paveikslėlius su originalia aplankalų ir failų struktūra (žr. 31 pav.), matyti, kad išgavus pradinius failus su „Dedexer“ programėle, failų padaugėjo. Taip pat matyti, kaip tyrime su „Apk-Tool“ programa, „ProGuard“ apsauga pervadina dalį aplankalų ir pagrindinius failus, kuriuose saugomas pagrindinis funkcionalumas. Dar vienas skirtumas, kad failų tipai dabar yra *Ddx*, o originalūs failai – *Java* tipo.

Inicializacijos vektoriaus saugojamos vietos radimas (žr. 41, 42 pav.):

```
.field public static final hiddenIV Ljava/lang/String; = "AZ:\u0017?K?/ \u0015\u001C$K.T-"
.field public static final hiddenSalt Ljava/lang/String; = "\u0016?z\f\u0013???^xs\u0010I?\u0014??H\u001EH,?\u001E?\u001B9_?"
```

41 pav. „Dedexer“ inicializacijos vektoriaus tyrimas be „ProGuard“ apsaugos

Inicializacijos vektorius neapsaugotoje mobiliojoje programėlėje, kaip ir su „Apk-Tool“ programėle, rastas tokio pat pavadinimo faile, bet skirtingo tipo *BuildConfig.ddx*. Iš paveikslėlio (žr. 41 pav.) matyti inicializacijos vektorius, pavadinimu *hiddenIV*. Jo reikšmė, lyginant su originalu (žr. 32 pav.), atrodo skirtinga, bet taip atrodo dėl koduotės, o reali reikšmė yra tokia pati, kaip originalaus inicializacijos vektoriaus.

Inicializacijos vektoriaus apsaugotoje mobiliojoje programėlėje radimas:

```
la428c:
    invoke-direct   {v4},com/donbra/notes/a/a/e ; e()Z
    move-result v0
    if-eqz  v0,la439e
    invoke-virtual  {v4},com/donbra/notes/a/a/d ; d()V
    const-string    v0,"AZ:\u0017?K?/ \u0015\u001C$K.T-"
    invoke-virtual  {v0},java/lang/String/getBytes  ; getBytes()[B
    move-result-object  v2
    invoke-virtual  {v4},com/donbra/notes/a/a/c ; c()Ljava/lang/String;
    move-result-object  v0
```

42 pav. „Dedexer“ inicializacijos vektoriaus tyrimas su „ProGuard“ apsauga

Inicializacijos vektorius apsaugotoje mobiliojoje programėlėje rastas visose funkcijose, kuriose yra naudojamas (žr. 42 pav.). Tai netgi palengvina analizuotojui darbą, padeda išsiaiškinti, kaip veikia programinis kodas, nes jam nereikia ieškoti, kurioje vietoje paslėptas inicializacijos vektorius, jis tiesiog yra priskirtas pačiam kintamajam.

Funkcijos paieška, kuri sugeneruoja slaptą raktą pagal mobiliojo įrenginio informaciją (žr. 43, 44 pav.).

```
.method private generateKey(Ljava/lang/String;)Ljavax/crypto/SecretKey;
.throws Ljava/security/NoSuchAlgorithmException;
.throws Ljava/security/spec/InvalidKeySpecException;
.limit registers 13
; this: v11 (Lcom/donbra/notes/adapter/NoteListAdapter;)
; parameter[0] : v12 (Ljava/lang/String;)
.line 225
    const/16    v0,1000
.line 226
    const/16    v3,256
.line 227
    div-int/lit8    v8,v3,8
.line 228
    const-string    v6,"\u0016?z\f\u0013???^xs\u0010I?\u0014??H\u001EH,?\u001E?\u001B9_?"
.line 230
    new-instance    v5,java/security/SecureRandom
    invoke-direct   {v5},java/security/SecureRandom/<init>  ; <init>()V
.line 231
    const-string    v10,"\u0016?z\f\u0013???^xs\u0010I?\u0014??H\u001EH,?\u001E?\u001B9_?"
    invoke-virtual  {v10},java/lang/String/getBytes ; getBytes()[B
    move-result-object  v7
.line 233
    new-instance    v4,javax/crypto/spec/PBEKeySpec
    invoke-virtual  {v12},java/lang/String/toCharArray  ; toCharArray()[C
    move-result-object  v10
    invoke-direct   {v4,v10,v7,v0,v3},javax/crypto/spec/PBEKeySpec/<init>   ; <init>([C[BII)V
.line 235
    const-string    v10,"PBKDF2WithHmacSHA1"
    invoke-static   {v10},javax/crypto/SecretKeyFactory/getInstance ; getInstance(Ljava/lang/String;)Ljavax/crypto/SecretKeyFactory;
    move-result-object  v2
.line 236
    invoke-virtual  {v2,v4},javax/crypto/SecretKeyFactory/generateSecret    ; generateSecret(Ljava/security/spec/KeySpec;)Ljavax/crypto/SecretKey;
    move-result-object  v10
    invoke-interface    {v10},javax/crypto/SecretKey/getEncoded ; getEncoded()[B
    move-result-object  v1
.line 237
    new-instance    v9,javax/crypto/spec/SecretKeySpec
    const-string    v10,"AES/CTR/NoPadding"
    invoke-direct   {v9,v1,v10},javax/crypto/spec/SecretKeySpec/<init>  ; <init>([BLjava/lang/String;)V
.line 239
    return-object   v9
.end method
```

43 pav. „Dedexter“ funkcijos *generateKey* tyrimas be „ProGuard“ apsaugos

Paveikslėlyje (žr. 43 pav.) pavaizduota raktų generavimo funkcija *generateKey*, kuri atrodo labai panašiai, kaip ir su „Apk-Tool“ programos išgautu kodu (žr. 38 pav.). Ją irgi sunkiau skaityti, nei *Java*

kalba aprašytas funkcijas (žr. 30 pav.) dėl žemesnio lygio programavimo kalbos – *Assemblerio*. Nepaisant žemesnio kalbos lygio pradiniame kode, galima įžvelgti, kokios yra naudojamos funkcijos raktui sugeneruoti , pvz.: *getInstance* ar *generateSecret*.

```
.method private a(Ljava/lang/String;)Ljavax/crypto/SecretKey;
    .locals 5
    const/16 v0, 0x3e8
    const/16 v1, 0x100
    const-string v2, "\u0016?z\u000c\u0013???^xs\u0010I?\u0014??H\u001eH,?\u001e?\u001b9_?"
    new-instance v2, Ljava/security/SecureRandom;
    invoke-direct {v2}, Ljava/security/SecureRandom;-><init>()V
    const-string v2, "\u0016?z\u000c\u0013???^xs\u0010I?\u0014??H\u001eH,?\u001e?\u001b9_?"
    invoke-virtual {v2}, Ljava/lang/String;->getBytes()[B
    move-result-object v2
    new-instance v3, Ljavax/crypto/spec/PBEKeySpec;
    invoke-virtual {p1}, Ljava/lang/String;->toCharArray()[C
    move-result-object v4
    invoke-direct {v3, v4, v2, v0, v1}, Ljavax/crypto/spec/PBEKeySpec;-><init>([C[BII)V
    const-string v0, "PBKDF2WithHmacSHA1"
    invoke-static {v0}, Ljavax/crypto/SecretKeyFactory;->getInstance(Ljava/lang/String;)Ljavax/crypto/SecretKeyFactory;
    move-result-object v0
    invoke-virtual {v0, v3}, Ljavax/crypto/SecretKeyFactory;->generateSecret(Ljava/security/spec/KeySpec;)Ljavax/crypto/SecretKey;
    move-result-object v0
    invoke-interface {v0}, Ljavax/crypto/SecretKey;->getEncoded()[B
    move-result-object v0
    new-instance v1, Ljavax/crypto/spec/SecretKeySpec;
    const-string v2, "AES/CTR/NoPadding"
    invoke-direct {v1, v0, v2}, Ljavax/crypto/spec/SecretKeySpec;-><init>([BLjava/lang/String;)V
    return-object v1
.end method
```

44 pav. „Dedexter“ funkcijos *generateKey* tyrimas su „ProGuard“ apsauga

Paveikslėlyje (žr. 44 pav.) pavaizduota suspaustas ir sumažintas kodas, labai panašus į su „Apk-Tool“ programa gautą kodą. Tai palengvina kodo skaitymą ir analizę, nors tūrėtų būti atvirkščiai. Šiame kode, taip pat kaip ir neapsaugotame kode (žr. 38 pav.), matyti *generateSecret* ir *getInstance* naudojamos funkcijos.

Išvada: kodo atvaizdavimas atrodo kitaip, lyginant su „Apk-Tool“ ir „Dedexer“ programų išgautu pradiniu kodu, bet tai dėl kalbos skirtumo. „Dedexer“ programos visų trijų tyrimo dalių rezultatai yra identiški „Apk-Tool“ programos rezultatams. „Dedexer“ programa išgautame apsaugotame pirminiame kode, inicializacijos vektorius rastas funkcijoje, kurioje yra naudojamas, o neapsaugotame - *BuildConfig.ddx* faile.

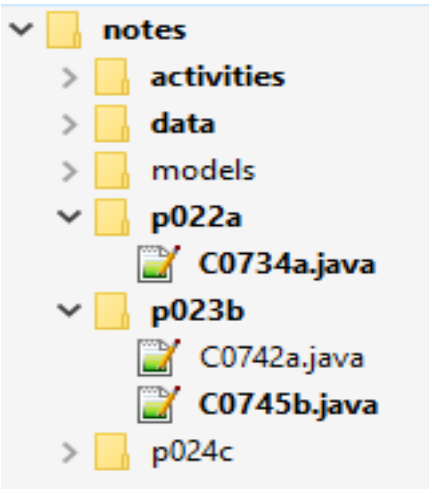
3.2.4.2. Tyrimo atlikimas dekompiliavimo metodu

Norint išgauti iš sukompiliuotos mobiliosios programėlės „Java“ kalba parašytą kodą, reikia naudoti dekompiliatorių. Dekompiliavimo metodas nuo išrinkimo skiriasi tuo, kad gaunamas kodas yra parašytas aukštesnio lygio kalba. Toks kodas yra lengviau skaitomas.

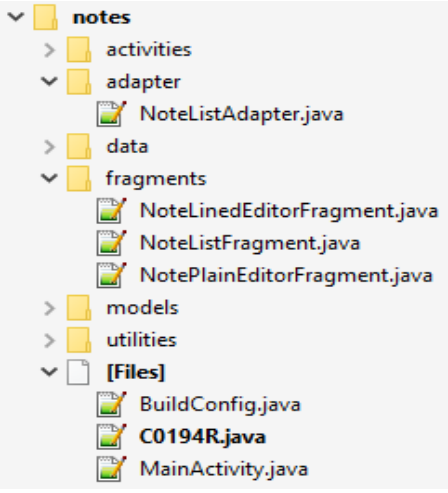
„Jadx“ – tai vienas paprasčiausių būdų išgauti pirminį kodą, panaudojant internetinį įrankį [32], kuris veikia *Jadx* dekompiliatorius principu. Įėjus į šios programos tinklalapį ir įkėlus *Note_one_v2.apk* failą, tinklalapis pateikia suarchyvuotą failą, kurį reiki atsisiųsti ir išarchyvuoti. Išarchyvavus failą, gautuose failuose matyti pirminis kodas.

Rezultatai:

Aplankalų ir failų struktūros tyrimo rezultatai pavaizduoti paveikslėliuose (žr. 45,46 pav.):



45 pav. „Jadx“ aplankalų ir failų struktūros tyrimas be „ProGuard“ apsaugos



46 pav. „Jadx“ aplankalų ir failų struktūros tyrimas su „ProGuard“ apsauga

Paveikslėlyje (žr. 46 pav.) pavaizduota, kaip atrodo pagrindinių failų ir aplankalų struktūra, o paveikslėlyje (žr. 45 pav.) – apsaugota „ProGuard“ programėle. Palyginus šiuos paveikslėlius su originalia aplankalų ir failų struktūra (žr. 31 pav.), matyti, kad išgavus pradinius failus su „Jadx“ programėle, failų sumažėjo. Taip pat matyti, kaip „ProGuard“ apsauga pervadina dalį aplankalų ir pagrindinius failus, kuriuose saugomas pagrindinis funkcionalumas.

Inicializacijos vektoriaus saugojamos vietos radimas (žr. 47, 48 pav.):

```
public static final String hiddenIV = "AZ:\u0017?K?/ \u0015\u001c$K.T-";
public static final String hiddenSalt = "\u0016?z\f\u0013???^xs\u0010I?\u0014??H\u001eH,?\u001e?\u001b9_?";
```

47 pav. „Jadx“ inicializacijos vektoriaus tyrimas be „ProGuard“ apsaugos

Inicializacijos vektorius neapsaugotoje mobiliojoje programėlėje rastas *BuildConfig.java* faile. Jo reikšmė, lyginant su originalu (žr. 32 pav.), atrodo skirtinga, bet taip atrodo dėl koduotės, o reali reikšmė yra tokia pati, kaip originalaus inicializacijos vektoriaus.

```
byte[] bytes = "AZ:\u0017?K?/ \u0015\u001c$K.T-".getBytes();
```

48 pav. „Jadx“ inicializacijos vektoriaus tyrimas su „ProGuard“ apsauga

Inicializacijos vektorius apsaugotoje mobiliojoje programėlėje rastas visose funkcijose, kuriose yra naudojamas. Tai netgi palengvina analizuotojui darbą, padeda išsiaiškinti, kaip veikia programinis kodas, nes jam nereikia ieškoti, kurioje vietoje paslėptas inicializacijos vektorius, jis tiesiog yra priskirtas pačiam kintamajam (žr. 48 pav.).

Funkcijos paieška, kuri sugeneruoja slaptą raktą pagal mobiliojo įrenginio informaciją (žr. 49, 50 pav.):

```
private SecretKey generateKey(String pass) throws NoSuchAlgorithmException, InvalidKeySpecException {
    int saltLength = AccessibilityNodeInfoCompat.ACTION_NEXT_AT_MOVEMENT_GRANULARITY / 8;
    String s = BuildConfig.hiddenSalt;
    SecureRandom random = new SecureRandom();
    return new SecretKeySpec(SecretKeyFactory.getInstance("PBKDF2WithHmacSHA1")
            .generateSecret(new PBEKeySpec(pass.toCharArray(), BuildConfig.hiddenSalt.getBytes(), 1000, AccessibilityNodeInfoCompat.ACTION_NEXT_AT_MOVEMENT_GRANULARITY))
            .getEncoded(), "AES/CTR/NoPadding");
}
```

49 pav. „Jadx“ funkcijos *generateKey* tyrimas be „ProGuard“ apsaugos

Paveikslėlyje (žr. 49 pav.) pateikta, raktų generavimo funkcija *generateKey*. Ji atitinka originalią funkciją (žr. 30 pav.), tik yra kitaip išdėstyta, t. y. nebėra kintamųjų aprašymo, panaikinti tarpai, gražinama reikšmė surašyta į vieną eilutę.

```
private SecretKey m4503a(String str) {
    String str2 = "\u0016?z\f\u0013???^xs\u0010I?\u0014??H\u001eH,?\u001e?\u001b9_?";
    SecureRandom secureRandom = new SecureRandom();
    return new SecretKeySpec(SecretKeyFactory.getInstance("PBKDF2WithHmacSHA1")
        .generateSecret(new PBEKeySpec(str.toCharArray(), "\u0016?z\f\u0013???^xs\u0010I?\u0014??H\u001eH,?\u001e?\u001b9_?".getBytes(), 1000, 256))
        .getEncoded(), "AES/CTR/NoPadding");
}
```

50 pav. „Jadx“ funkcijos *generateKey* tyrimas su „ProGuard“ apsauga

Paveikslėlyje (žr. 50 pav.) pavaizduotas suspaustas ir sumažintas kodas. Vietoje paslėptų kintamųjų atsiranda to kintamojo reikšmės. Tai palengvina kodo skaitymą ir analizę, nors turėtų būti atvirkščiai.

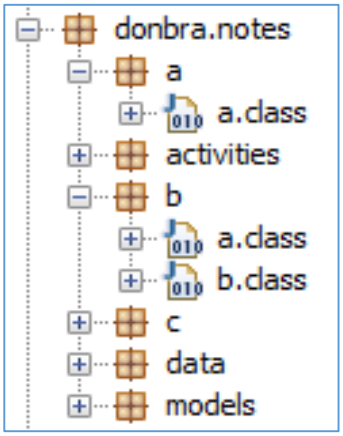
Išvada: aplankalų, failų struktūra atkurta lyginant su originalia struktūra (žr. 31 pav.), nors failų sumažėjo apsaugotoje mobiliojoje programėlėje, bet juose išliko ta pati informacija, kaip ir originale. Tiek apsaugotame pradiniame kode, tiek neapsaugotame, pavyko rasti ieškotos raktų generavimo funkcijos *generateKey*. Aprašyto inicializacijos vektoriaus kodo vieta skiriasi apsaugotoje ir neapsaugotoje mobiliojoje programėlėje. Apsaugotame kode inicializacijos vektorius rastas toje funkcijoje, kurioje yra naudojamas, o neapsaugotame – *BuildConfig.java* faile.

„Dex2jar“ ir „Jd-gui“ – tai programos, reikalingos šiam bandymui atlikti. „Dex2jar“ programa yra skirta *Dex* tipo failui perversti į *Jar* tipo failą. „Jd-gui“ programa nuskaito gautą *Jar* tipo failą ir sugeneruoja *Java* tipo failus.

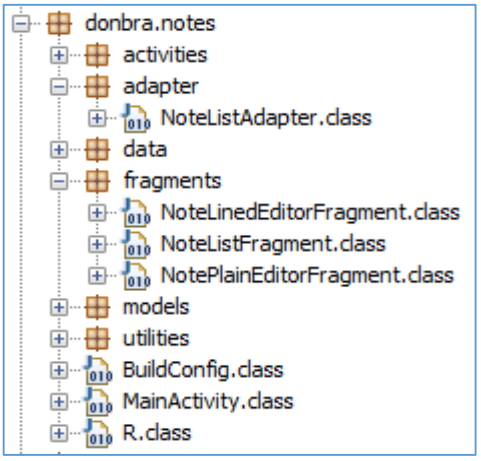
Pirmiausia, atsisiunčiamos abi programėlės, tada – mobiliosios programėlės failo *Note_one_v2.apk* tipas pakeičiamas į *Note_one_v2.zip*. Gautas failas išarchyvuojamas. Aplankale, į kurį išarchyvuotas failas, surandamas classes.dex failas ir nukopijuojamas į „Dex2jar“ programos aplankalą. Tuomet paleidžiama „Command Prompt“ pultas, kurioje nurodoma išsaugotos „Dex2jar“ programos vieta ir panaudojama komanda „d2j-dex2jar.bat classes.dex“. Po šios komandos *classes.dex* failas konvertuojamas į *classes-dex2jar.jar* failą. Atidaroma „Jd-gui“ programa, parenkamas gautas failas ir programa atveria jį kaip *Java* tipo failus.

Rezultatai:

Aplankalų ir failų struktūros tyrimo rezultatai pavaizduoti paveikslėliuose (žr. 51, 52 pav.):



51 pav. „Dex2jar“ ir „Jd-gui“ aplankalų ir failų struktūros tyrimas su „ProGuard“ apsauga



52 pav. „Dex2jar“ ir „Jd-gui“ aplankalų ir failų struktūros tyrimas be „ProGuard“ apsaugos

Paveikslėlyje (žr. 52 pav.) pavaizduota, kaip atrodo pagrindinių failų ir aplankalų struktūra, o paveikslėlyje (žr. 51 pav.) – kaip apsaugota „ProGuard“ programėle. Palyginus šiuos paveikslėlius su originalia aplankalų ir failų struktūra (žr. 31 pav.), matyti, kad išgavus pradinius failus su „Dex2jar“ ir „Jd-gui“ programomis, failų sumažėjo. Taip pat matyti, kad „ProGuard“ apsauga pervadina dalį aplankalų ir pagrindinius failus, kuriuose saugomas pagrindinis funkcionalumas.

Inicializacijos vektoriaus saugojamos vietos radimas (žr. 53, 54 pav.):

```
/* Error */
public void a(a parama, int paramInt)
{
  // Byte code:
  //   0: aconst_null
  //   1: astore 4
  //   3: aload_0
  //   4: getfield 25  com/donbra/notes/a/a:a   Ljava/util/List;
  //   7: iload_2
  //   8: invokeinterface 247 2 0
  //   13: checkcast 249    com/donbra/notes/models/a
  //   16: invokevirtual 252     com/donbra/notes/models/a:e   ()Ljava/lang/Integer;
  //   19: invokevirtual 257     java/lang/Integer:intValue    ()I
  //   22: iconst_1
  //   23: if_icmpne +3 -> 26
  //   26: aload_0
  //   27: invokespecial 259     com/donbra/notes/a/a:e   ()Z
  //   30: ifeq +205 -> 235
  //   33: aload_0
  //   34: invokevirtual 262     com/donbra/notes/a/a:d   ()V
  //   37: ldc_w 264
  //   40: invokevirtual 42 java/lang/String:getBytes     ()[B
  //   43: astore 5
  //   45: aload_0
  //   46: invokevirtual 266     com/donbra/notes/a/a:c   ()Ljava/lang/String;
  //   49: astore_3
  //   50: aload_0
  //   51: aload_3
  //   52: invokespecial 268     com/donbra/notes/a/a:a   (Ljava/lang/String;)Ljavax/crypto/SecretKey;
  //   55: astore_3
  //   56: aload_0
  //   57: aload_3
  //   58: new 270     javax/crypto/spec/IvParameterSpec
  //   61: dup
  //   62: aload 5
  //   64: invokespecial 273     javax/crypto/spec/IvParameterSpec:<init>     ([B)V
  //   67: invokevirtual 275     com/donbra/notes/a/a:a   (Ljavax/crypto/SecretKey;Ljavax/crypto/spec/IvParameterSpec;)Ljava/lang/String;
```

53 pav. „Dex2jar“ ir „Jd-gui“ inicializacijos vektoriaus tyrimas su „ProGuard“ apsauga

Inicializacijos vektorius apsaugotoje mobiliojoje programėlėje nerastas, nes „ProGuard“ apsauga suklaidino „Dex2jar“ ir „Jd-gui“ programas ir joms nepavyko tinkamai atversti pradinio kodo (žr. 53 pav.).

Inicializacijos vektorius neapsaugotoje mobiliojoje programėlėje rastas *BuildConfig.java* faile (žr. 54 pav.):

```
public static final String hiddenIV = "AZ:\027?K?/ \025\034$K.T-";
public static final String hiddenSalt = "\026?z\f\023???^xs\020I?\024??H\036H,?\036?\0339_?";
```

54 pav. „Dex2jar“ ir „Jd-gui“ inicializacijos vektoriaus tyrimas be „ProGuard“ apsaugos

Inicializacijos vektorius reikšmė, lyginant su originalu (žr. 32 pav.), atrodo skirtinga, bet taip atrodo dėl koduotės, o reali reikšmė yra tokia pati, kaip originalaus inicializacijos vektoriaus.

Funkcijos paieška, kuri sugeneruoja slaptą raktą pagal mobiliojo įrenginio informaciją (žr. 55, 56 pav.):

```
private SecretKey generateKey(String paramString)
  throws NoSuchAlgorithmException, InvalidKeySpecException
{
  int i = 'Ä' / 8;
  new SecureRandom();
  byte[] arrayOfByte = "\026?z\f\023???^xs\020I?\024??H\036H,?\036?\0339_?".getBytes();
  paramString = new PBEKeySpec(paramString.toCharArray(), arrayOfByte, 1000, 256);
  return new SecretKeySpec(SecretKeyFactory.getInstance("PBKDF2WithHmacSHA1").generateSecret(paramString).getEncoded(), "AES/CTR/NoPadding");
}
```

55 pav. „Dex2jar“ ir „Jd-gui“ funkcijos *generateKey* tyrimas be „ProGuard“ apsaugos

Paveikslėlyje (žr. 55 pav.) pateikta *generateKey* funkcija. Ji atitinka originalią funkciją (žr. 30 pav.), tik kitaip išdėstytą, t. y. nebėra kintamųjų aprašymo, panaikinti tarpai, taip pat vietoje paslėpto kintamojo atsiranda to kintamojo reikšmės.

```
private SecretKey a(String paramString)
{
  new SecureRandom();
  byte[] arrayOfByte = "\026?z\f\023???^xs\020I?\024??H\036H,?\036?\0339_?".getBytes();
  paramString = new PBEKeySpec(paramString.toCharArray(), arrayOfByte, 1000, 256);
  return new SecretKeySpec(SecretKeyFactory.getInstance("PBKDF2WithHmacSHA1").generateSecret(paramString).getEncoded(), "AES/CTR/NoPadding");
}
```

56 pav. „Dex2jar“ ir „Jd-gui“ funkcijos *generateKey* tyrimas su „ProGuard“ apsauga

Paveikslėlyje (žr. 56 pav.50 pav.) pavaizduotas suspaustas ir sumažintas kodas. Vietoje paslėptų kintamųjų atsiranda to kintamojo reikšmės. Tai palengvina kodo skaitymą ir analizę, nors turėtų būti atvirkščiai.

Išvada: aplankalų, failų struktūra atkurta lyginant su originalia struktūra (žr. 31 pav.), nors failų sumažėjo apsaugotoje mobiliojoje programėlėje, bet juose išliko ta pati informacija kaip ir originale. Tiek apsaugotame, tiek neapsaugotame pradiniame kode pavyko rasti ieškotą raktų generavimo funkciją *generateKey*. Inicializacijos vektoriaus aprašymo kodo vieta apsaugotoje ir neapsaugotoje mobiliojoje programėlėje skiriasi. Neapsaugotos mobiliosios programėlės kode inicializacijos vektorių pavyko rasti *BuildConfig.java* faile, o apsaugotoje rasti nepavyko, nes inicializacijos vektorius yra įrašytas vietoje kintamojo, bet visos funkcijos nepavyko atversti į pradinį kodą.

Bendri tyrimo rezultatai:

Lentelėje atvaizduoti mobiliosios programėlės apsaugojimo nuo apgrąžos inžinerijos rezultatai (žr. 5 lentelė):

5 lentelė Mobiliosios programėlės apsaugojimas nuo apgrąžos inžinerijos tyrimo rezultatai

Programėlė	Apsaugota „ProGuard“				Neapsaugota			
Naudota programa	**APK-Tool**	**Dedexer**	**Jadx**	**Dex2jar ir Jd-gui**	**APK-Tool**	**Dedexer**	**Jadx**	**Dex2jar ir Jd-gui**
Rastas rakto sudarymas	Taip	Taip	Taip	Taip	Taip	Taip	Taip	Taip
Reikalingas laikas rasti rakto sudarymą (min.)	~30	~30	~10	~10	~10	~10	~5	~5
Rastas paslėptas IV	Taip	Taip	Taip	Ne*	Taip	Taip	Taip	Taip
Reikalingas laikas rasti paslėptą IV (min.)	~20	~20	~15	~15	~10	~10	~5	~5
Atkurti visi failai	Taip	Taip	Taip	Taip	Taip	Taip	Taip	Taip
Atkurti originalūs failų, funkcijų pavadinimai	Dalis	Dalis	Dalis	Dalis	Taip	Taip	Taip	Taip

* – inicializacijos vektorius nebuvo rastas, bet rasta vieta, kurioje jis yra naudojamas.

Išvada: kaip matyti iš 5 lentelės, naudojant „ProGuard“ apsaugą, ji pasunkina kodo analizę, bet yra greitai įveikiama. Naudojant vieną iš mokamų apsaugų (žr. 2 lentelė), rezultatai parodė geresnius apsaugos metodus nei „ProGuard“, vadinasi, prireiktų daugiau laiko, kad mobiliosios programos apsauga būtų įveikta.

3.3. Tyrimo išvados

Tyrimas sudarytas iš dviejų dalių: sukurtos mobiliosios programėlės funkcionalumo testavimo ir mobiliosios programėlės apsaugos įveikimo. Tyrimas atliktas žmogaus, turinčio tris metus programavimo patirties (šio metodo kūrėjo). Pirmojoje dalyje atliktas testavimas buvo sėkmingas ir nustatyta, kad programėlės funkcionalumas atitinka išsikeltą reikalavimą – multimedijos failai veiktų tik „savo“ įrenginiuose. Antroji dalis pavyko labai gerai, nes pavyko nustatyti, jog standartinė „Proguard“ apsauga nepakankama ir jos geriau nenaudoti. Ji tik prailgino sugaištą laiką, ieškant funkcijų pirminiame kode. Tačiau jos vis tiek buvo rastos ir išanalizuotos.

„ProGuard“ apsauga sukėlė daugiau problemų norint išanalizuoti pradinį kodą, kadangi pervadino daugumą funkcijų, kintamųjų, failų ir aplankalų pavadinimų, sukeitė vietomis funkcijas bei kai kurias funkcijas pakeitė taip, kad pritaikius apgrąžos inžineriją, dekompiliavimo programos darė klaidą, t. y. funkcijos nebuvo neįmanoma atversti į pradinį kodą. Analizuojant apsaugotą pradinį kodą prireikė nuo penkių iki dvidešimt minučių daugiau laiko, nei tiriant neapsaugotą pradinį kodą.

Greičiausiai funkcijos rastos naudojant „Jadx“ ir „Dex2jar“ su „Jd-gui“ dekompiliavimo programas, nes jos mobiliąją programėlę perverčia į aukštos kalbos *Java* kodą.

Nustatyta, kad standartinė „Android“ OS siūloma „ProGuard“ apsauga yra nepakankama ir gerai neapsaugo mobiliosios programėlės nuo apgrąžos inžinerijos atakų, todėl vietoje jos rekomenduojama naudoti komercinius analogus, naudojančius papildomus apsaugos metodus, tokius kaip pvz. apsauga realiu laiku.

4. IŠVADOS

1. Atlikus skaitmeninio turinio teisių valdymo sistemų analizę nustatyta, kad jos visos nėra patogios vartotojo atžvilgiu, nes nei viena sistema neleidžia perkelti multimedijos failų į kitus to paties vartotojo įrenginius.
2. Pasiūlytas originalus metodas, kurio pagrindu sudaryta skaitmeninio turinio teisių valdymo sistema, atvaizduojanti skaitmeninio turinio failus, veikianti be interneto prieigos ir turinti galimybę perkelti parsisiųstą vaizdo failą į kitą „savo“ įrenginį. Pasiūlytas metodas pagrįstas multimedijos failų užšifravimu *AES-CTR* šifravimo metodu, multimedijos failų iššifravimo raktų saugojimu tarp mobiliosios programėlės failų, užšifruotame tekstiniame faile, failo iššifravimo raktą sugeneruojant pagal mobilaus išmaniojo įrenginio techninius duomenis ir mobilios programėlės reikšmes.
3. Eksperimentiniam pasiūlyto metodo įvertinimui realizuota skaitmeninio turinio teisių valdymo sistemos pagrindinė dalis – multimedijos failų peržiūros programėlė. Programėlė realizuota „Android“ OS, o jos apsaugai nuo piktavalių panaudota standartinė „Android“ OS siūloma „ProGuard“ apsauga.
4. Eksperimentų rezultatai parodė, kad siūlomo metodo pagrindu realizuotas skaitmeninio turinio teisių valdymo sistemos prototipas pilnai atitinka jam iškeltus reikalavimus ir užtikrina multimedijos failų atjungties režimu peržiūrą, tuo pat metu, skirtingai nuo esamų sistemų, leisdamas vieną kartą įsigytą turinį peržiūrėti visuose vartotojo turimuose įrenginiuose.
5. Eksperimentų metu nustatyta, kad „Android“ OS siūloma naudoti standartinė „ProGuard“ programa neužtikrina reikiamos apsaugos nuo apgrąžos inžinerijos atakų: programuotojas su kelių metų programavimo stažu gali nesunkiai įveikti tokiu būdu apsaugotas programas vos per keletą minučių. Todėl rekomenduojama naudoti kitus apsaugos nuo apgrąžos inžinerijos metodus ir programas, turinčias daugiau apsaugos galimybių, pvz. apsauga realiame laike.

5. LITERATŪRA

[1] D. Bosomworth, „Mobile Marketing Statistics 2015,“ 2015. [Tinkle]. Pasiekiama: http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/. [Kreiptasi 10 01 2016].

[2] Z. Ma, J. Huang, M. Jiang and X. Niu, „A Video Watermarking DRM Method Based on H.264 Compressed Domain with Low Bit-Rate Increasement,“ *Chinese Journal of Electronics,* t. 25, nr. 4, pp. 641-647, 07 08 2016.

[3] CastLabs, „Drmtoday,“ CastLabs, 2015. [Tinkle]. Pasiekiama: http://www.drmtoday.com/platforms. [Kreiptasi 10 01 2016].

[4] M. R. Mitashe, A. R. B. Habib, A. Razzaque, I. A. Tanima and J. Uddin,, „An adaptive digital image watermarking scheme with PSO, DWT and XFCM,“ įtraukta *IEEE International Conference on Imaging, Vision & Pattern Recognition (icIVPR),* Dhaka, 2017.

[5] Ensaf Hussein, Mohamed A. Belal, „Digital Watermarking Techniques,“ *International Journal of Engineering Research & Technology,* t. 1, nr. 7, 25 09 2012.

[6] Pooja Dabas and Kavita Khanna, „A Study on Spatial and Transform Domain Watermarking Techniques,“ *International Journal of Computer Applications,* t. 71, nr. 14, 05 2013.

[7] Moulick, Subhayan Roy and Arora, Siddharth and Jain, Chirag and Panigrahi, Prasanta K, „Reliable SVD based Semi-blind and Invisible Watermarking Schemes,“ *eprint arXiv,* p. 11, 03 2015.

[8] Flixster, „Flixster Video FAQ,“ Flixster, [Tinkle]. Pasiekiama: http://support.flixstervideo.com/link/portal/15025/15266/ArticleFolder/141/Playback. [Kreiptasi 02 10 2016].

[9] G. Steirer, „Clouded Visions: UltraViolet and the Future of Digital Distribution,“ *Television & New Media,* t. 16, nr. 2, pp. 180 - 195, 10 03 2014.

[10] „Widevine,“ Google, [Tinkle]. Pasiekiama: http://www.encoding.com/widevine/. [Kreiptasi 11 01 2016].

[11] „expressPlay,“ Intertrust, [Tinkle]. Pasiekiama: http://www.expressplay.com/. [Kreiptasi 12 01 2016].

[12] UltraViolet, „UltraViolet FAQ,“ UltraViolet , [Tinkle]. Pasiekiama: http://www.uvdemystified.com/. [Kreiptasi 01 03 2017].

[13] N. info, „Netflix,“ Netflix, [Tinkle]. Pasiekiama: https://ir.netflix.com/index.cfm. [Kreiptasi 03 03 2017].

[14] Ezequiel Minaya and Amol Sharma, „Netflix Expands to 190 Countries,“ 2016.

[15] Netflix, „Neftlix devices,“ Neftlix, 2017. [Tinkle]. Pasiekiama: https://devices.netflix.com/en/. [Kreiptasi 03 03 2017].

[16] J. Y. Pan and S. H. Ma, „Advertisement removal of Android applications by reverse engineering,“ įtraukta *2017 International Conference on Computing, Networking and Communications (ICNC),* Santa Clara, CA, 2017.

[17] Faruki Parvez, Fereidooni Hossein, Laxmi Vijay, Conti Mauro, and Gaur Manoj, „Android Code Protection via Obfuscation Techniques: Past, Present and Future Directions,“ *eprint arXiv,* 30 11 2016.

[18] GuardSquare, „ProGuard,“ GuardSquar, [Tinkle]. Pasiekiama: https://www.guardsquare.com/proguard. [Kreiptasi 15 11 2016].

[19] GuardSquare, „DexGuard,“ GuardSquare, [Tinkle]. Pasiekiama: https://www.guardsquare.com/dexguard. [Kreiptasi 15 11 2016].

[20] DexProtector, „DexProtector,“ DexProtector, [Tinkle]. Pasiekiama: https://dexprotector.com/. [Kreiptasi 16 11 2016].

[21] Dr. Prerna Mahajan, Abhishek Sachdeva, „A Study of Encryption Algorithms AES, DES and RSA for Security,“ *Global Journal of Computer Science and Technology Network, Web & Security,* t. 13, nr. 15, 2013.

[22] M. Sharma and R. B. Garg, „DES: The oldest symmetric block key encryption algorithm,“ įtraukta *2016 International Conference System Modeling & Advancement in Research Trends (SMART),* Moradabad, 2016.

[23] N. M. S. Iswari, „Key generation algorithm design combination of RSA and ElGamal algorithm,“ įtraukta *016 8th International Conference on Information Technology and Electrical Engineering (ICITEE),* Yogyakarta, Indonesia, 2016.

[24] S. M. Soliman, B. Magdy and M. A. Abd El Ghany, „Efficient implementation of the AES algorithm for security applications,“ įtraukta *2016 29th IEEE International System-on-Chip Conference (SOCC),* Seattle, WA, USA, 2016.

[25] D. Jayasinghe, R. Ragel, J. A. Ambrose, A. Ignjatovic and S. Parameswaran, „Advanced modes in AES: Are they safe from power analysis based side channel attacks?,“ įtraukta *2014 IEEE 32nd International Conference on Computer Design (ICCD),* Seoul, 2014.

[26] „Identifying App Installations,“ Google, 30 03 2011. [Tinkle]. Pasiekiama: https://android-developers.googleblog.com/2011/03/identifying-app-installations.html. [Kreiptasi 12 03 2017].

[27] I. Gepko, „General requirements and security architecture for mobile phone anti-cloning measures,“ įtraukta *IEEE EUROCON 2015 - International Conference on Computer as a Tool (EUROCON),* Salamanca, 2015.

[28] N. P. Tran, M. Lee, S. Hong and S. J. Lee, „Parallel Execution of AES-CTR Algorithm Using Extended Block Size,“ įtraukta *2011 14th IEEE International Conference on Computational Science and Engineering,* Dalian, Liaoning, 2011.

[29] Apktool, „Apktool,“ [Tinkle]. Pasiekiama: https://ibotpeaches.github.io/Apktool/install/. [Kreiptasi 20 04 2017].

[30] „Dedexer,“ [Tinkle]. Pasiekiama: http://dedexer.sourceforge.net/. [Kreiptasi 04 04 2017].

[31] G. Paller, „Dedexer,“ Sourceforge, [Tinkle]. Pasiekiama: http://dedexer.sourceforge.net/. [Kreiptasi 20 04 2017].

[32] A. Rukin, „javadecompilers,“ [Tinkle]. Pasiekiama: http://www.javadecompilers.com/apk. [Kreiptasi 05 04 2017].