KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

Vaidotas Šimaitis

NFC mokėjimo sistema klientas - terminalas

Baigiamasis magistro darbas

Vadovas
Prof. dr. Eligijus Sakalauskas

KAUNAS, 2015

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
KOMPIUTERIŲ KATEDRA

TVIRTINU
Katedros vedėjas
Prof. dr. Algimantas Venčkauskas

NFC mokėjimo sistema klientas - terminalas

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

Vadovas
Prof. dr. Eligijus Sakalauskas

Recenzentas
doc. dr. Stasys Maciulevičius

Projektą atliko
Vaidotas Šimaitis

KAUNAS, 2015

KAUNO TECHNOLOGIJOS UNIVERSITETAS

Informatikos fakultetas

(Fakultetas)

Vaidotas Šimaitis

(Studento vardas, pavardė)

Informacijos ir informacinių technologijų sauga (kodas 621E10003)

(Studijų programos pavadinimas, kodas)

Baigiamojo projekto „NFC mokėjimo sistema klientas - terminalas“

AKADEMINIO SĄŽININGUMO DEKLARACIJA

20 15 m. gegužės 25 d.

Kaunas

Patvirtinu, kad mano **Vaidoto Šimaičio** baigiamasis projektas tema „NFC mokėjimo sistema klientas - terminalas“ 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)*

Šimaitis, V. NFC mokėjimo sistema klientas - terminalas. *magistro* baigiamasis projektas / vadovas prof. dr. Eligijus Sakalauskas; Kauno technologijos universitetas, Informatikos fakultetas, kompiuterių katedra.

Kaunas, 2015. 57 psl.

SANTRAUKA

NFC mokėjimo sistema klientas – terminalas – tai mokėjimų sistema, kuri skirta atlikti mokėjimus išmaniuoju telefonu artimo nuotolio ryšio pagalba. Pagrindinis šios sistemos tikslas yra užtikrinti atliekamus apmokėjimus ir perduoti duomenis pasinaudojant artimo ryšio technologija. Šiuo darbu siekiama išnagrinėti mokėjimo sistemos architektūrą ir specifiką, duomenų saugos bei vartotojo autentifikacijos problemas bei pasiūlyti duomenų konfidencialumą bei autentiškumą užtikrinančius sprendimus.

Analitinėje dalyje yra nagrinėjama mokėjimo sistemų veikimo specifika, reikalavimai. Šioje dalyje analizuojamas NFC-SEC duomenų protokolas, turintis duomenų apsaugos mechanizmą. Yra iškeliami jo privalumai bei trūkumai, kurie aprašomi nagrinėjant reikalavimus mokėjimo sistemoms. Analizės pabaigoje iškeliami darbo tikslai bei uždaviniai, apibrėžiamas siekiamo sprendimo apibrėžimas.

Antroje darbo dalyje yra aprašoma modeliuojamo saugos sprendimo architektūra, sudaromas duomenų apsikeitimo planas. Numatomas raktų generavimo ir apsikeitimo būdas ir jų panaudojimas apsaugant duomenis bei užtikrinant jų autentiškumą. Numatyta, jog sistema naudos papildomą šifravimo ir autentifikavimo įrenginį, kuriame bus saugomi vartotojo privatieji raktai.

Trečioje dalyje atliekamas NFC duomenų perdavimo realizavimas panaudojant išmaniuosius telefonus. Atliekami duomenų šifravimo ir autentifikavimo greitaveikų tyrimai, juos palyginant su kitu protokolu, aprašomi prototipo privalumai.

Darbas užbaigiamas išvadomis, kurios apibendrina darbo rezultatus.

Šimaitis, V. NFC payment system client - terminal. Master work in information technologies / supervisor prof. dr. Eligijus Sakalauskas; Kauno University of technology, Departament of Informatics, Departament of Computers.

Kaunas, 2015. 57 psl.

SUMMARY

NFC payment system client – terminal is the payment system, which is designed to make payments with smartphone using near field communication. The main goal of this system is to ensure that payments are done and to send data using near field communication technology. This project is used to investigate architecture of payment systems and specifics, issues of data safety and user authentication and to offer solution to keep data confidential and authentic.

In the analysis part there is investigations of payment systems specifics, requirements. In this part there is analysis made for NFC-SEC protocol, which provides security mechanisms. There is amplified its advantages and disadvantages, which are described by investigating requirements of electronic payment systems. The main goals and tasks is made in the end of analysis part, there is also description of the solution which will be made.

In the second part there is description of architecture for the security solution, data traffic exchange plan, anticipatory the way of keys generation and exchange and usage to secure data and maintain its authenticity. System should use another device for cryptography and authentication, where will be private keys stored.

In the third part there is realisation of of NFC data traffic using smartphones. Experiments of the performance of cryptography and authentication are made, described and compared with another protocol. Descriptions of advantages are made.

This project ends summarizing conclusions of the results.

TURINYS

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

TERMINŲ IR SANTRUMPŲ ŽODYNAS

NFC – Artimo ryšio komunikacija *(angl. Near Field Communication)*

SSC - Saugumo standartų taryba *(angl. Security Standards Council)*

ESE - Integruotas saugumo elementas *(angl. Embedded Security Element)*

SNEP - Paprastas NDEF apsikeitimo protokolas *(angl. Simple NDEF Exchange Protocol)*

NDEF – NFC duomenų apsikeitimo formatas *(angl. NFC data exchange format)*

PIN – asmeninis identifikacijos numeris *(angl. Personal Identification Number)*

DSA – Skaitmeninio parašo algoritmas *(angl. Digital Signature Algorithm)*

SSE – Bendros paslapties servisas *(angl. Shared Secret Service)*

SCH – Saugaus kanalo servisas *(angl. Secure Channel Service)*

IK – Intelektuali kortelė

ARK – artimo ryšio komunikacija

ĮVADAS

Darbas priklauso informacinių technologijų studijų, informacijos ir informacinių technologijų saugos krypties specializacijai. Darbe nagrinėjami mobiliųjų telefonų programų saugos klausimai, atsižvelgiama į vartotojų poreikius bei būtinybę užtikrinti saugų duomenų apsikeitimą tarp mobiliųjų telefonų ir įrenginių.

Šio darbo tikslas yra užtikrinti mobiliųjų telefonų NFC įrenginių perduodamų duomenų saugumą ir autentiškumą, pritaikant juos mokėjimams. Darbe remiamasi tarptautiniais standartais, kuriais užtikrinamas perduodamų duomenų saugumas.

Plečiantis NFC technologijų panaudojimams, šia technologija paremti mokėjimai taps neišvengiami, todėl vartotojo duomenų apsaugojimas ilgainiui taps vis svarbesnis naudojant šią technologiją.

Darbo problematika ir aktualumas

Darbas aktualus šiuolaikinėms technologijos spartinant įvairius gyvenimiškus procesus, o tai pat ir apmokėjimus. Didesnė apmokėjimo greitaveika palengvins asmenų mokėjimus, pagreitins šį procesą, tuo pačiu įtraukiant naujas technologijas, kurių perduodamus duomenis būtina tinakmai apsaugoti. Tai pat būtinas duomenų autentifikavimas, siekiant užtikrinti jog duomenys nebus suklastoti ar panaudoti pakartotinai.

Darbo tikslas ir uždaviniai

Darbo tikslas:

- Sukurti mokėjimų sistemą, kuri naudotų NFC technologiją atlikti mokėjimus ir pasiūlyti saugumo sprendimus perduodamiems duomenims.

Darbo uždaviniai:

- Išanalizuoti esamas sistemas ir reikalavimus mokėjimo sistemoms
- Suprojektuoti mokėjimų sistemą naudojančią NFC technologiją
- Remiantis tarptautiniais standartais išanalizuoti ir pasiūlyti tinkamą duomenų apsaugos mechanizmą.
- Sukurti sistemos klientas – terminalas prototipą ir palyginti su esamais NFC saugos sprendimais.

Darbo rezultatai ir jų svarba

Pagrindiniai darbo rezultatai yra mokėjimo sistema, naudojanti NFC duomenų perdavimo technologiją ir galinti užtikrinti šių perduodamų duomenų saugumą ir autentiškumą.

Darbo struktūra

Dokumentą sudaro:

- Analitinė dalis, kurioje analizuojama technologija, esami sukurti saugos spendimai šiai technologijai, tarptautinių saugos standartų normos mokėjimo sistemoms.
- Projektinė dalis aprašo kuriamos sistemos modelį, duomenų šifravimo ir pasirašymo mechanizmus, reikalavimus programinei įrangai.
- Prototipo dalyje aprašoma sistemos struktūra, veikimas.
- Tyrimo dalyje aprašomos šifravimo algoritmų greitaveikos bei palyginimas su NFC-SEC protokolu.

1. MOBILIŲJŲ TELEFONŲ MOKĖJIMŲ SRITIES ANALIZĖ

1.1. Analizės tikslas

Analizės tikslas yra surinkti reikalavimus mokėjimo sistemoms, analizuoti galimus sprendimus konkrečiai NFC technologijai pritaikyti saugos klausimais. Analizės metu vienas iš tikslų yra išanalizuoti esamas NFC technologijas ir pasiūlyti sprendimų būdus informacinių duomenų saugai užtikrinti.

1.2. Tyrimo objektas, sritis ir problema

Pagrindinė mokėjimo sistemų problema ir svarbiausias uždavinys yra konfidencialių vartotojo duomenų saugojimas ir saugus jų perdavimas viešaisiais ryšių kanalais [1]. Privalo būti kiek galima labiau sumažinama galimybė perimti šiuos duomenis, iššifruoti ir panaudoti blogiems tikslams: neteisėtiems atsiskaitymams, elektroninių pinigų vagystėms, vartotojo duomenų panaudojimui be jo žinios siekiant gauti asmeninę naudą ar tiesiog pakenkti.

Šiuo metu naudojama populiariausia elektroninių atsiskaitymų sistema paremta elektroninėmis kortelėmis [2]. Ši sistema atsiskaitymams ar grynųjų pinigų išėmimo operacijoms naudoja dviejų faktorių autentifikavimą: reikalinga turėti pačią kortelę (ką turi) ir žinoti tos kortelės PIN (angl. personal identification number) kodą (ką žinai). Šie du faktoriai sustiprina vartotojo pinigų apsaugą ir sumažina piktavalių galimybes perimti duomenis.

Projektuojant ir kuriant mobiliąją NFC mokėjimų sistemą svarbu įvertinti tai, jog vieno faktoriaus vartotojo identifikacijai nepakanka. Svarbu dabartiniams mobiliesiems įrenginiams sukurti tokią programinę įrangą, kuri neleistų nuskaityti, perimti ar kitu būdu sužinoti vartotojo vieną iš faktorių : identifikavimo kodo arba slaptažodžio. Pagal Saugumo Standartų Tarybos siūlymus, mokėjimo sistemos turi atitikti tam tikrus reikalavimus, siekiant sumažinti galimybę neteisėtai perimti duomenis ir nepalikti kuriamoje programinėje įrangoje spragų.

1.3. NFC technologijos pritaikymo mokėjimo sistemoms analizė

Kuriant mokėjimų sistemą, ypač tokią, kuri naudoja bevielį ryšį, svarbu atkreipti dėmesį ne tik į pačios programinės įrangos saugumą, tačiau ir fizinę apsaugą.

NFC technologija yra ypač jautri bevielio ryšio duomenų nuskaitymui ar perėmimui. Reikalinga nustatyti, kokį fizinį apsauginį barjierą galima naudoti, norint jog duomenys nebūtų nuskaityti ir panaudoti piktavališkoms paskatoms. Tai vadinamojo “žmogaus viduryje” ataka, kuomet tarp siuntėjo ir

gavėjo gali būti perimti, nuskaityti, arba pakeisti duomenys. Tam būtina naudoti vartotojo autentifikavimo funkcijas, kurios užtikrins duomenų autentiškumą.

Duomenų apsaugos atvejų visi siunčiami arba priimami duomenys turės būti šifruoti, panaudojant saugius kriptografinius metodus bei plačiai taikomus protokolus, tokius kaip SSL. Informacijos perdavimo atveju, ji turės būti šifruojama panaudojant viešąjį raktą, o iššifruojama panaudojant privatųjį raktą, kuris turės būt saugomas nuo priėjimo prie jo [3].

Kuriant programinę įrangą, reikės paisyti Saugumo Tarybos rekomendacijų [4], siekiant sumažinti riziką jog duomenys bus perimti, nuskaityti šifruoti pranešimai arba šifravimo/iššifravimo raktai.

Tai pat turės būti numatoma kaip turės būti apsaugota programinė įranga ir duomenų bazė, kurioje saugomi vartotojų duomenys. Šie duomenys turi būti itin griežtai prižiūrimi, kadangi bet koks duomenų nutekinimas turės didelių nuostolių.

1.4. NFC mokėjimo sistemos naudotojų analizė

Pagrindiniai objekto naudotojai yra vartotojai, atsiskaitantys už prekes ar paslaugas, tai pat įmonės ar privatūs asmenys, kurie siekia gauti šiuos apmokėjimus.

Reikalavimas vartotojui yra turėti mobilųjį telefoną, kuris gali būti naudojamas NFC duomenų perdavimui. Tai pat būtina programinė įranga, skirta duomenis per NFC perduoti, kurios sprendimas ir yra kuriamas.

1.5. Esamų mokėjimo sistemų sprendimo metodų analizė

Vienas iš dabartinių panašios paslaugos teikėjų yra „Master Card“. Jie siųlo atsiskaitymus mobiliaisiais telefonais panaudojant NFC funkciją [5]. Norėdamas atsiskaitinėti už prekes ar paslaugas vartotojas turi padaryti šiuos žingsnius:

1. Rasti banką, kuris siųlo šią paslaugą.
2. Įsigyti telefoną su integruota NFC funkcija.
3. Užsiregistruoti „MasterCard PayPass“ banke.
4. Atsisiųsti mokėjimo kortelės detales į telefoną.
5. Aktyvuoti „MasterCard PayPass“ programą.

Šios paslaugos tiekėjas leidžia atsiskaitinėti tiesiai iš bankinės sąskaitos, tačiau neužtikrina informacijos saugumo papildomais saugumo faktoriais. Realiai nupirktas telefonas jau gali būti užkrestas kompiuteriniu virusu ir panaudoti vartotojo duomenis blogiems tikslams.

Šis atsiskaitymo būdas nereikalauja jokio PIN kodo ar parašo, jeigu atsiskaitoma Jungtinėse Amerikos Valstijose už mažiau negu 25 dolerius.

„Google Wallet“ [6]

Sparčiai populiarėjanti atsiskaitymų sistema, naudojanti NFC technologiją. Sistema palaikoma mobiliosios aplikacijos, neapribota išleidžiama pinigų suma. Sistema apsaugota galingu kriptografiniu mechanizmu, o pats prisijungimas prie „Google Wallet“ surištas su kitom paskyromis, naudojamomis jungtis prie Google. Ši mokėjimo sistema naudoja „Google Wallet Fraud Protection“ sistemą“, kuri garantuoja šimtaprocentinį pinigų grąžinimą už patvirtintus bet neautorizuotus mokėjimus, tačiau tam būtina būti Jungtinių Amerikos Valstijų piliečiu.

Visa „Pay Wave“ [7]

Visa bankinių kortelių sistema leidžia atsiskaitinėti NFC kortelėmis, tačiau nepalaiko mobiliųjų aplikacijų atsiskaitymų. Naudojantis NFC technologija galima atsiskaityti už prekes ar paslaugas tik priliečiant kortelę prie skaitytuvo. Didelė problema yra ta, jog nereikalaujamas joks saugos kodas, kuris užtikrintų, jog niekas kitas negalės atsiskaityti ne savo kortele. Kadangi kortelės yra pasyviniai elementai, jos neturi viduje kriptografinių mechanizmų, galinčių šifruoti perduodamą informaciją, todėl nuskaičius siunčiamą signalą yra tikimybė, jog duomenys gali būti panaudojami blogiems tikslams.

Express Pay

NFC sistemą naudojanti Jungtinių Amerikos Valstijų mokėjimo sistema. Panašiai kaip ir kiti šios paslaugos teikėjai, leidžia atsiskaitinėti priliečiant kortelę prie skaitytuvo. Neapsaugota kelių faktorių saugumo sistema, ir leidžianti atsiskaityti tik už mažiau negu 25 dolerius.

Lietuvoje realiai yra dvi galimybės atsiskaityti NFC technologija. Viena iš jų yra MokiPay [8] mokėjimo sistema. Naudojama kartu su mobiliuoju telefonu, tačiau jam reikalingas papildomas NFC adapteris, beto, reikalingą sumą reikia turėti pervestą į tam skirtą sąskaitą, o tai kelia nepatogumų. Apie apsaugos sistemas informacija nėra pateikiama, tačiau yra patvirtinimas, jog duomenų apsaugos licenziją yra išdavęs Lietuvos bankas.

Kita sistema yra Tapsis [9]. Realizuota moksleivių ir studentų pažymėjimuose. Galima atsiskaitinėti už smulkias paslaugas.

1.5.1. NFC - SEC

NFC – SEC [10] Saugus duomenų apsikeitimo protokolas standartizuotas kaip NFCIP-1. Šis standartas aprašo signalo perdavimą ir protokolą artimo nuotolio komunikacijos technologijoje. Papildoma serija NFC apsaugos standartų aprašo protokolo steką kuris suteikia galimybę aplikacijai nepriklausomai naudoti kriptografines funkcijas duomenų perdavimo lygyje, virš NFCIP-1 lygio.

NFC apsaugos standartas bus naudojamas visuose NFC komunikacijos priemonėse, kur reikalaujama apsaugos nuo duomenų perėjimo ir nebūtinai reikalaujama specialų programinės įrangos kriptografijos priemonių.

Tipinis pavyzdys panaudojant artimo nuotokio šifruotą komunikaciją būtų dviejų arba daugiau įrenginių susiejimas (įrenginių poravimas) tolimesniam jų veikimui didesniu atstumu naudojant bevielio ryšio technologiją. „Bluetooth“ ir „WiFI“ poravimo protokolai naudos NFC saugumo standartus apsikeisti iš saugumo pusės jautria informacija, o tuomet jau galės veikti jiems įprastu rėžimu. Praktiškai numatoma, jog įrenginiai apsikeis slaptažodžiais ar identifikavimo priemonėmis prieš juos panaudojant pagal jų paskirtį.

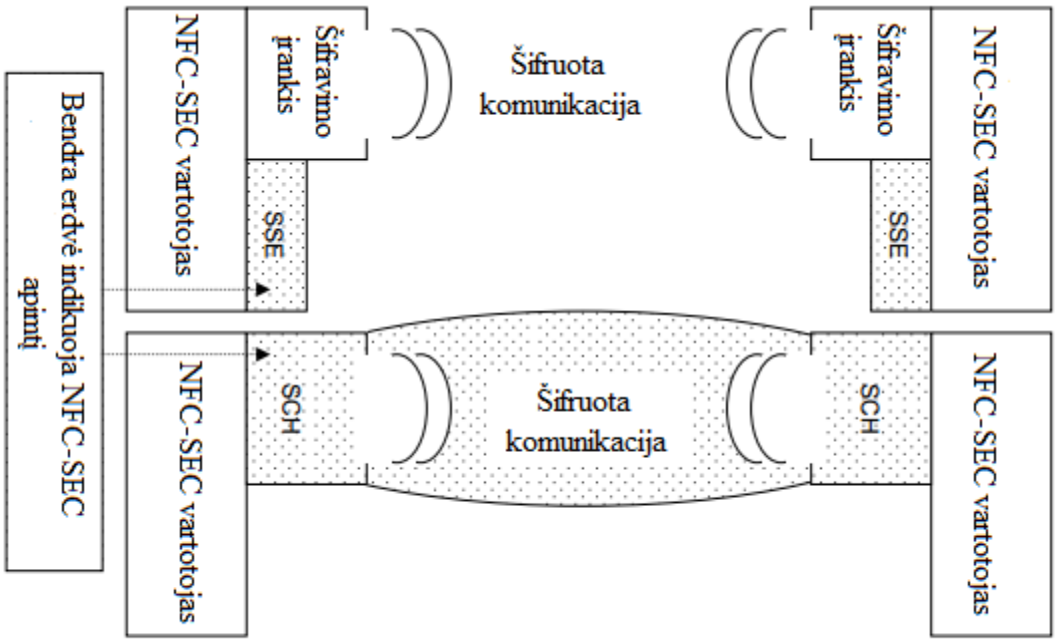
Koncepsinis NFC apsaugos modulis supaprastins pačias šio įrenginio apsaugos specifikacijas ir leis laisviau bei patogiau pritaikyti jį ateities plėtros planuose. Bendroji struktūra, kuri apibūdinama šį modulį yra aprašyta ECMA-385 NFC-SEC: NFCIP-1 Saugumo Paslaudų ir Protokolo standartu [11].

NFC saugumo standartas ECMA-385 NFC-SEC: NFCIP-1 aprašo šifravimo mechanizmus panaudojant ECDH ir AES. Elipsinių kreivių Difi-Helmano raktų apsikeitimo protokolas ir AES šifravimo protokolas užtikrina duomenų saugumą ir integralumą.

NFC-SEC protokolas aprašo du servisus, kurie pateikiami schemoje (1.1 pav.) [12].

Bendros paslapties servisas (angl. Shared Secret Service, toliau SSE) aprašo bendrosios paslapties apsikeitimą tarp dviejų vartotojų ar įrenginių, kurie pasikeičia duomenimis panaudodami kriptografinius mechanizmus.

Sekančiuose žingsniuose Saugaus Kanalo Servisas (angl. Secure Channel Service, toliau SCH) panaudoja bendrąją paslaptį sukurti saugią duomenų perdavimo terpę apsaugojant visus toliau perduodamus duomenis. Visi saugumo mechanizmai aprašomi kriptografiniais standartais.

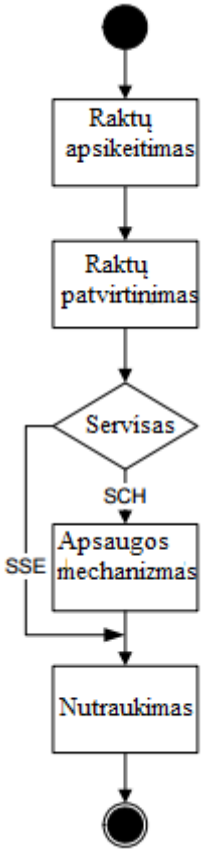


1.1 pav. NFC-SEC schema

1.5.1.1. Protokolas

Duomenų perdavimo protokolas atvaizduotas 1.2 paveikle.

Žingsniai „raktų apsikeitimas“ ir „raktų patvirtinimas” yra būtini tiek SSE, tiek SCH. “Apsaugos mechanizmas” suteikia konkretų duomenų užšifravimą ir apsaugą, kuri yra reikalaujama SCH. Paskutiniame žingsnyje SSE ir SCH ryšiai ir duomenų šifravimo raktai yra sunaikinami.



1.2 pav. Pagrindiniai NFC-SEC protokolo žingsniai

NFC-SEC-01 protokolas suteikia apsaugą siunčiamiems pranešimams ir kriptografinius metodus užtikrinant saugią komunikaciją tarp NFC įrenginių, kurie nepasikeičia jokia, saugumo prasme, jautria informacija prieš pradėdami apsaugotą duomenų perdavimo sesiją.

Viešojo rakto kriptografija, labiau specifinė elipsinių kreivių Difi-Helmano raktų apsikeitimo schema panaudojama apsikeisti slaptuoju raktu tarp šių įrenginių. Bendroji paslaptis naudojama pradėti SSE ir SCH ryšio įslaptinimą. Saugumo mechanizmo parametras šiuo atveju yra 192 bitai.

Tačiau šis apsaugos mechanizmas neapsaugo nuo "Žmogus viduryje" (angl. Man in the middle) atakos, kadangi duomenys nėra pasirašomi, tai yra nepanaudojamas autentifikavimo mechanizmas pries užmezgant sesiją tarp šių dviejų įrenginių. Praktinė šio tipo ataka vertinama kaip sunkiai įgyvendinama, kadangi įrenginiai duomenis perduoda itin mažu atstumu, tačiau rizika išlieka, jeigu panaudojant pažanges ryšio perėmimo priemones šis ryšys būtų sukompromituotas.

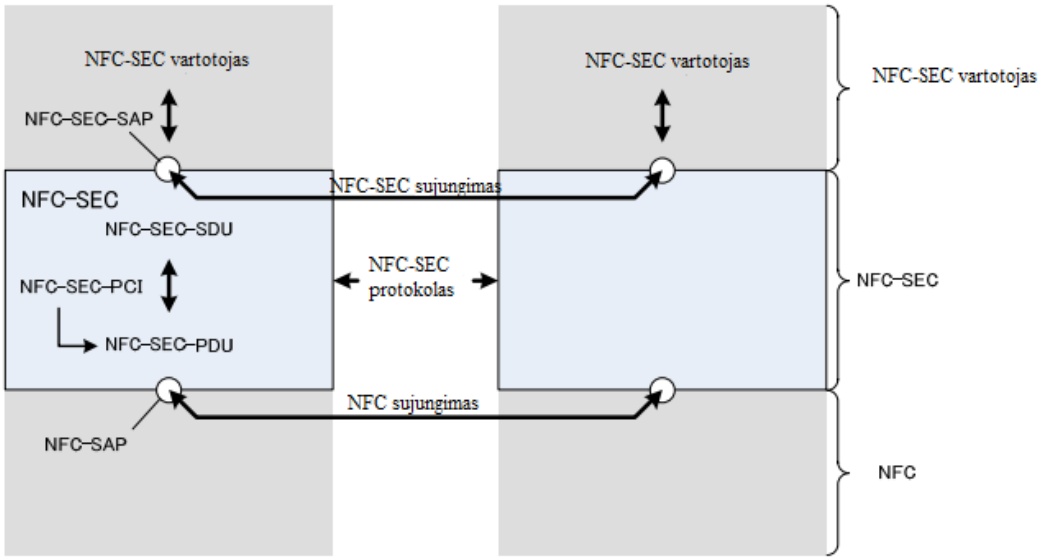
Specifikuota rakto išvedimo funkcija, raktų patvirtinimas, duomenų integralumo užtikrinimas ir duomenų šifravimo mechanizmai yra paremti AES. Duomenų apsauga užtikrinama 128 bitų AES raktu veikiant CTR režimu, kuris laikomas saugiu ir tinkamuryšio apsaugai.

1.5.1.2. Architektūra

NFC-SEC vartotojai sužadina ir atidaro šį servisą per NFC-SEC serviso atidarymo tašką (angl. Service Access Point). NFC-SEC esybės gauna NFC-SEC-SDUs (užklausas) iš kito NFC-SEC vartotojo ir gražina NFC-SEC-SDUs (atsakymą).

Ši architektūra tai pat aprašo ir SCH ir SSE. Tam, jog tiekti NFC-SEC servisą, susijungusios NFC-SEC esybės apsikeičia NFC-SEC-PDUs taip patvirtindamos NFC-SEC protokola esant NFC sujungimui.

Abi NFC-SEC esybės siunčia ir priima NFC-SEC-PDUs per NFC-SAP (1.3 pav.).



1.3 pav. NFC-SEC architektūra

Sistema nepalaiko duomenų autentifikacijos protokolų, siunčiami duomenys gali būti perimami ir klastojami. Nors NFC ir dirba itin mažu atstumu, “Žmogus viduryje” atakos atveju duomenys gali būti klastojami ir perimama vartotojų siunčiama informacija.

1.5.2. Teorinis pasirinktos problemos nagrinėjimas

Pagrindinė mokėjimo sistemų problema ir svarbiausias uždavinys yra konfidencialių vartotojo duomenų saugojimas ir saugus jų perdavimas viešaisiais ryšių kanalais. Privalo būti kiek galima labiau sumažinama galimybė perimti šiuos duomenis, iššifruoti ir panaudoti blogiems tikslams: neteisėtiems atsiskaitymams, elektroninių pinigų vagystėms, vartotojo duomenų panaudojimui be jo žinios siekiant gauti asmeninę naudą ar tiesiog pakenkti.

Šiuo metu naudojama populiariausia elektroninių atsiskaitymų sistema paremta elektroninėmis kortelėmis. Ši sistema atsiskaitymams ar grynųjų pinigų išėmimo operacijoms naudoja dviejų faktorių autentifikavimą: reikalinga turėti pačią kortelę (ką turi) ir žinoti tos kortelės PIN (angl. personal identification number) kodą (ką žinai). Šie du faktoriai sustiprina vartotojo pinigų apsaugą ir sumažina piktavalių galimybes perimti duomenis.

Projektuojant ir kuriant mobiliąją NFC mokėjimų sistemą svarbu įvertinti tai, jog vieno faktoriaus vartotojo identifikacijai nepakanka [13]. Svarbu dabartiniams mobiliesiems įrenginiams sukurti tokią programinę įrangą, kuri neleistų nuskaityti, perimti ar kitu būdu sužinoti vartotojo vieną iš faktorių : identifikavimo kodo arba slaptažodžio. Pagal Saugumo Standartų Tarybos siųlymus, mokėjimo sistemos turi atitikti tam tikrus reikalavimus, siekiant sumažinti galimybę neteisėtai perimti duomenis ir nepalikti kuriamoje programinėje įrangoje spragų.

1.5.2.1. Saugumo reikalavimai mokėjimo sistemai

Remiantis Saugumo Standartų Tarybos (angl. Security Standards Council) pateikiamomis rekomendacijomis [14], kurios aprašo saugumo reikalavimus apmokėjimo programinei įrangai, bus formuluojamos būtinos priemonės, užtikrinant didelį duomenų saugumo lygį. Tai pat atsižvelgiama į šios organizacijos rekomendacijas Duomenų Apsaugos (angl. Data Security Standard) bei PIN Kodo Apsaugos Priemonių [15] (angl. PIN Security Requirements) pateikiamuose dokumentuose.

1.5.2.2. Nesaugoti autentifikavimo duomenų

Pirmasis reikalavimas programinei įrangai yra nesaugoti, nelaikyti autentifikacijos duomenų darbinėje atmintyje. Gauti duomenys iš darbinės atminties turi būti panaikinti iškarto po panaudojimo. Slaptažodžiai, identifikacijos žymės, visa slapta informacija turi būti ištrinta iškarto, kai tik šių duomenų nebereikia. Tai neleis kenkėjiškai programinei įrangai nuskaityti vertingos

informacijos ir panaudoti jos blogais tikslais. Svarbu užtikrinti pilną duomenų sunaikinimą, jog po ištrinimo nebūtų galima jų atstatyti . Rekomendacija nelaikyti svarbių duomenų galioja ir šifruotai informacijai.

Vienintelis atvejis išlaikyti šią informaciją yra leistinas testavimo arba programinės įrangos derinimo tikslais, tačiau turi būti laikomasi šių punktų:

Duomenys gali būti neištrinami sprendžiant konkrečią problemą, ir turi būti pritaikomi būtent šiai problemai.

Duomenys turi būti saugomi žinomoje, apsaugotoje duomenų saugojimo vietoje, neprieinamoje nepageidautiniems asmenims arba programinei įrangai kuri gali panaudoti informaciją negerais tikslais.

Saugomos informacijos kiekis turi būti minimalus - tik tiek, kiek jos reikia problemai išspręsti.

Saugoma informacija turi būti šifruota stipriu kriptografiniu metodu.

Duomenys turi būti ištrinti tuojau pat, kai tik jų nebereikia. Tai pat turi būti ištrinami įrašai (angl. Log File), derinimo informacija, ir kiti duomenys, panaudoti sprendžiant problemą.

1.5.2.3. Užtikrinti vartotojų duomenų apsaugą

Vartotojų duomenų saugojimo apibrėžimas apima saugumą vartotojo pusės programinės įrangos, tai pat duomenų apsaugą produkto kūrėjo pusėje, kur saugomi identifikaciniai duomenys bei kita informacija.

1.5.2.4. Suteikti saugų autentifikavimo mechanizmą

Mokėjimo programa turi užtikrinti vartotojo autentifikavimą unikaliu identifikacijos numeriu (ID).

Mokėjimo sistema negali naudoti prisijungimo vardų ir slaptažodžių, sukurtų pagal nutylėjima (angl. Delaults), kadangi šie prisijungimai dažniausiai būna viešai žinomi.

Sistema privalo identifikuoti vartotoją mažiausiai pagal vieną iš identifikacijos tipų:

- Ką žinai (slaptažodis)
- Ką turi (kortelė ar papildomas įrenginys)
- Kas esi (biometriniai duomenys)

Mokėjimo sistemos autentifikacijos ID ir slaptažodis (ar kitas autentifikacijos įrankis) turi būti skirti tik vienam asmeniui, negali būti skirtas keliems ar grupei asmenų.

Jeigu autentifikavimui naudojamas slaptažodis, jis turi būti ne mažiau negu 7 simbolių ilgio, tai pat turi būti panaudoti tie skaitmenys, kiek raidės (Pagal NIST SP 800-63-1 elektroninės autentifikacijos standartą). Kaip alternatyvą galima naudoti kitokį slaptažodžio mechanizmą, tačiau jis turi užtikrinti ne mažesnį stiprumą, negu aprašyta prieš tai. Silpni slaptažodžiai suteiks galimybę juos atspėti ir pasinaudoti vartotojo duomenimis neteisėtiems veiksmams.

Esant keletui nesėkmingų prisijungimų, vartotojas turi būti blokuojamas mažiausiai 30-čiai minučių, arba mažiau, jeigu administratorius šį blokavimą panaikiną. Ši apsaugos priemonė saugo nuo bandymų atspėti slaptažodį.

Jeigu vartotojui prisijungus, sistema nėra naudojama 15 minučių, turi būti reikalaujama prisijungti iš naujo. Taip apsaugoma, jog vartotojas nepaliko savo mobiliojo įrenginio ir nesuteikia galimybės kitiems asmenis neteisėtai pasinaudoti jo duomenimis.

1.5.2.5. Mokėjimų registravimas

Kaip ir daugeliui kitų programų, įvykių registravimas yra svarbus stebint programos veiklą, bei registruojant vartotojų veiksmus. Šiuo atveju svarbus yra piniginių operacijų stebėjimas, jų išsaugojimas.

1.5.2.6. Reikalavimai kuriant mokėjimų programinę įrangą

Programinės įrangos kūrimo procese tai pat svarbu užtikrinti programos saugą: nepalikti spragų, neleisti jog programinis kodas būtų nutekintas arba būtų išduoti apsaugą užtikrinantys kriptografiniai mechanizmai. Tai pat negalima testavimo metu naudoti realių prisijungimo duomenų, reikalinga atlikti tik simuliacijas su duomenimis, kurie vėliau būtinai bus sunaikinti.

Turi būti užtikrintas programinės įrangos atnaujinimas, kuomet tik yra atliekami svarbūs pakeitimai ar pataisymai saugumo dalyje.

Reikia užtikrinti, jog kode neliktų dažnai pasitaikančių saugumo spragų, tokių kaip duomenų bazių injekcijos, buferio perpildymai, nesaugūs duomenų, kriptografinių mechanizmų saugojimo būdai, nesaugūs komunikavimo protokolai, tai pat vidinių veikimo principų, nustatymų (konfiguracijos) nutekėjimai.

1.5.2.7. Apsaugoti bevielį duomenų perdavimą

Vienas iš mokėjimo programos reikalavimų yra užtikrinti, jog duomenys, perduodami bevieliu ryšiu, būtų perduodami saugiai. NFC mokėjimo sistema iš principo kuriama kaip bevielio ryšio pagrindu veikiantis duomenų perdavimo būdas, skirtas atsiskaityti už prekes ar paslaugas.

Svarbiausias nurodymas mokėjomo sistemoms yra visų pirma pakeisti nustatymus pagal nutylėjimą (angl. Defaults). Šie nustatymai dažniausiai yra žinomi ir viešai prieinami, todėl palengvina galimybes duomenims perimti. Aišku, NFC technologija veikia tik artimu nuotoliu (1-5cm), tačiau iš principo yra nešifruotas duomenų perdavimo būdas, todėl reikės užtikrinti saugumo mechanizmus duomenų pasikeitimo metu.

Vartotojų duomenys negali būti laikomi ten pat, kur veikia mokėjimo sistemos programinė įranga. Tai yra dėl priežasties, jog vartotojų duomenų apsaugai yra didesni saugumo reikalavimai negu programinės įrangos, kuri yra prieinama daug viešiau. Jeigu programinė įranga taptų prieinama asmenims, norintiems panaudoti pačią programą ar duomenis blogiems tikslams (mokėjimo metu), jie lygiai tai pat galėtų gauti priėjimą ir prie vartotojų informacijos.

1.5.2.8. Slaptos informacijos šifravimas perduodant informaciją viešais tinklais

Nors kaip jau minėta NFC technologija veikia tik itin mažu atstumu [16], lyginant su kitais (WiFi, Bluetooth) bevielio informacijos perdavimo būdais. Tačiau būtina užtikrinti, jog šios informacijos tarp mobiliojo telefono ir terminalo neperims, nenuskaitys, ir savo tikslais nepanaudos nepageidaujami asmenys.

Mokėjimo programos, perduodančios informaciją viešaisiais tinklais, turi naudoti stiprią kriptografinę sistemą ir saugius protokolus (pavyzdžiui: SSL/TLS IPSEC, SSH).

1.5.2.9. Vartotojo duomenų apsauga

Pirmiausia norint užtikrinti duomenų apsaugą, reikia būti tikram, jog tiek mobilusis įrenginys, tiek atsiskaitymo terminalas nėra ir nebus panaudoti neteisėtiems veiksmams. Sekantis dalykas, padidinantis saugumą, yra gamintojų pradinių nustatymų keitimas: tiek programinės įrangos, tiek tinklo įrenginių slaptažodžiai turi būti pakeičiami į saugius ir viešai neprieinami.

Programinės įrangos kūrime nereikia palikti nenaudojamų programinio kodo fragmentų, papildomų tvarkyklių, visa tai gali turėti spragų, kas leis piktavaliams pasinaudoti tuo ir išgauti slaptą informaciją.

Vartotojo slaptasis raktas turi būti saugomas pagal kriterijus:

- Užšifruotas raktu, ne mažiau saugiu negu vartotojo raktas.
- Vartotojo slaptasis raktas negali būti pakeičiamas kitu be vartotojo ar kūrėjo žinios.

Tai pat vartotojų duomenys turi būti saugomi principu, jog kuo mažiau asmenų turėtų priėjimą prie jų duomenų. Tą galima įgyvendinti skirstant prisijungimą prie sistemos turinčius asmenis į administratorius, vartotojus, personalą ir taip toliau. Priėjimo prie duomenų draudimas užtikrins, jog nebus leista prisijungti asmenims, kurie to daryti negali, ir nebus leidžiama pasinaudoti informacija.

Duomenų apsaugos srityje tai pat svarbu panaudoti apsaugos priemones įrenginių, kurie dalyvauja priimant, apdorojant, perduodant su vartotojo mokėjimu susijusią informaciją. Nusikaltėliai siekia nuskaityti vartotojo identifikacinius duomenis, imituodami mokėjimų sistemų terminalus, pavyzdžiui pavogdami pačius įrenginius siekiama išsiaiškinti jų veikimo principai ir kaip juos galima panaudoti prieš pačius vartotojus. Taip sukuriami modifikuoti įrenginiai, skirti duomenų perdavimui į piktavalių rankas. Kitas atvejis, kuomet įmontuojama papildoma techninė įranga,

tiesiog skenuojanti perduodamus duomenis. Šis tipas pavojingesnis tuo, kad pinigų mokėjimo transakcijos vykdomos be jokių uždelsimų ir negali būti pastebėtos duomenų siuntėjo ar gavėjo.

Prie fizinės įrenginių apsaugos tai pat priskiriamas įrenginių programinės įrangos atnaujinimas, jų protokolavimas profilaktiniams patikrinimams.

1.6. Darbo tikslas, uždaviniai, planas ir siekiami privalumai

Darbo tikslas:

- Sukurti mokėjimų sistemą klientas - terminalas, kuri naudotų NFC technologiją atlikti mokėjimus ir pasiūlyti saugumo bei autentifikavimo sprendimus perduodamiems duomenims.

Darbo uždaviniai:

- Išanalizuoti esamas sistemas ir reikalavimus mokėjimo sistemoms
- Suprojektuoti mokėjimų sistemą naudojančią NFC technologiją
- Remiantis tarptautiniais standartais išanalizuoti ir pasiūlyti tinkamą duomenų apsaugos mechanizmą.

Darbo privalumas – NFC technologijos panaudojimas mokėjimams atlikti, tai pat perduodamų duomenų saugumo bei autentiškumo užtikrinimas.

1.7. Siekiamo sprendimo apibrėžimas

Darbe siekiama sukurti duomenų perdavimo sistemą, leidžiančią saugiai perduoti duomenis NFC technologija iš mobiliųjų telefonų į kitus įrenginius, tai pat užtikrinant duomenų autentiškumą.

1.8. Analizės išvados

Darbo analizės dalyje apibūdinta elektroninių mokėjimų vystymasis ir šių atsiskaitymų tendencijos didėjimas. Augant šių operacijų paklausai, didėja būtinybė pasiųlyti paprastesnes, greičiau veikiančias bei taupančias vartotojų laiką atsiskaitymo priemones.

Aptartos programų saugos rekomendacijos, skirtos mokėjimo sistemų saugai užtikrinti tiek jų kūrimo, testavimo etapuose, tiek realizuojant jau sukurtą produktą. Į šias rekomendacijas būtina atsižvelgti kuriant mokėjimo sistemą.

Išanalizuotos dabar plačiai naudojamos mokėjimo sistemos, skirtos atsiskaityti internet už prekes ar paslaugas. Aprašyti jų veikimo algoritmai, pateiktos principinės schemos.

Iš to tolimesniuose darbuose reikia atlikti:

- Pagrindinis uždavinys yra suformuoti NFC mokėjimo sistemos struktūrą, kuri leistų saugiai keistis duomenimis tarp vartotojo ir terminalo. Struktūra turi apibrėžti perdavimo protokolus, pačią programinę įrangą.
- Parinkti duomenų apsikeitimo protokolą, leidžiantį saugiai perdavinėti informaciją tarp vartotojų, neatskleidžiant informacijos asmenims, kurie neturi jos gauti.
- Suformuoti programinės įrangos struktūrą, jos veikimo principus.
- Parinkti NFC įrenginius darbui realizuoti.
- Parinkti vartotojo autentifikavimo funkcijas.
- Užtikrinti perduodamos informacijos konfidencialumą, integralumą ir autentiškumą.

2. NFC MOKĖJIMO SISTEMOS KLIENTAS - TERMINALAS SPRENDIMO KŪRIMAS

Pirmoje dalyje buvo smulkiai išanalizuota NFC technologija bei mokėjimų sistemos, tai pat techniniai reikalavimai programinei įrangai. Elektroninių mokėjimų sistemų reikalavimai bei rekomendacijos buvo pasirinktos remiantis Saugumo Standartų Tarybos (angl. Security Standards Council) pateiktais dokumentais, kuriuose aiškiai apibrėžtos normos šioms sistemoms, testavimo algoritmai bei smulkiai paaiškinamas kiekvienos rekomendacijos svarba.

Viena pagrindinių užduočių mokėjimo sistemoje yra neleisti jog slapti vartotojo duomenys, tokie kaip privatusis raktas, slaptažodis, patektų į piktavalių rankas. Tokiu atveju pasinaudojus perimtais duomenimis būtų galima atsiskaityti kito žmogaus vardu. Užtikrinant asmens duomenų apsaugą bus kreipiamas privataus rakto saugojimo galimybes, tai pat įvedamo slaptažodžio saugojimą atmintyje bei saugų pašalinimą iš jos.

Pagal Saugumo Standartų Tarybos reikalavimus vartotojui skirtą privatųjį DSA raktą bus siekiama saugoti imituojant intelektualiosios kortelės veikimą, kurioje bus atliekami kriptografiniai mechanizmas pasirašant siunčiamus pranešimus.

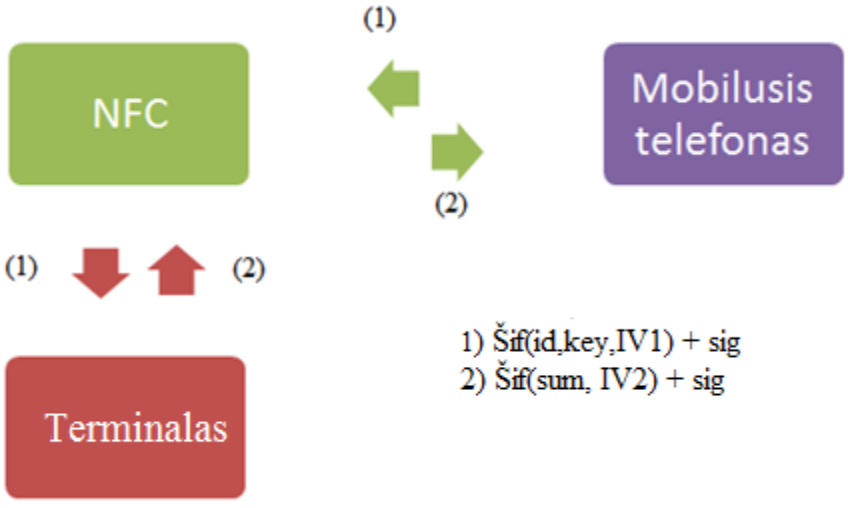
2.1. Mokėjimo sistemos architektūros sudarymas

Mokėjimo sistemos pagrindas šio projekto atveju bus NFC technologiją palaikantis mobilusis telefonas bei NFC terminalo funkciją atliekantis kitas mobilusis telefonas (terminalas). Šiuo atveju mobilusis telefonas bus skirtas kelioms procedūroms atlikti. Viena iš procedūrų – privataus rakto saugojimas ir realizavimas apmokėjimuose, duomenų pasirašymas patvirtinant savo tapatybę ir leidimą naudotis sąskaita. Privatusis raktas naudojamas kaip vienas iš dviejų faktorių asmens identifikavimo procedūroje, užtikrinant jog be šio mobilaus telefono (arba mažiausiai be šios imituojamos atminties kortelės) nebus galima atsiskaitinėti asmens sąskaita.

Mokėjimo veiksmo procesą pradės užmegsta sąsaja tarp mobilaus telefono ir ir NFC terminalo, tai yra kito mobilaus telefono su NFC funkcija, kuriame galima bus atvaizduoti perduodamus duomenis. Vartotojas, užmezgęs ryšį savo telefonu, panaudojant tam skirtą programinę įrangą, gaus informaciją kokią sumą reikia susimokėti. Šiuo atveju tai bus šifruota informacija, kuriai ketinama panaudoti bendrą AES raktą, siunčiant apmokėjimo sumą. Vartotojas išsiųs užšifruotą informaciją AES raktu, kurioje bus jo identifikaciniai duomenys bei sekančio pranešimo inicializavimo vektorius.

Serveris, arba bankas (šiuo atveju tai ketinama realizuoti mobilaus telefono su NFC funkcija panaudojimu) į mobilųjį telefoną atsakys siųsdamas sumą skaičiais, bei kitą inicializacijos vektorių, taip padidindamas AES šifrogramos saugumą. Ši informacija bus pasirašoma terminalo DSA raktu.

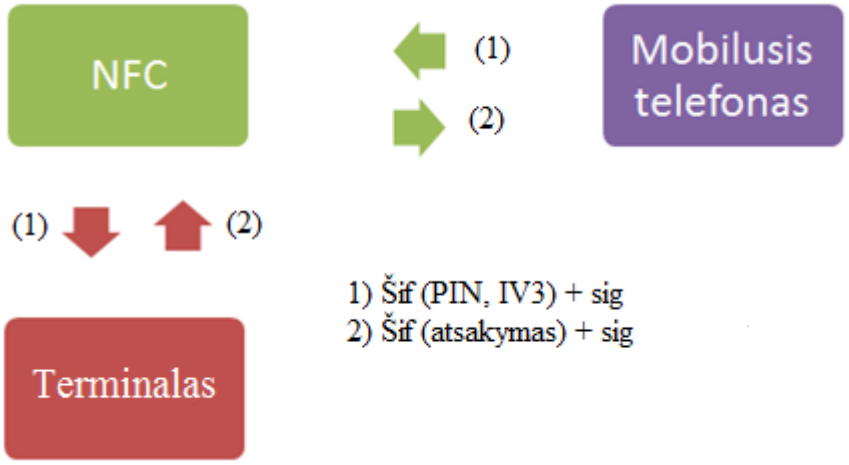
Toliau pateikiama mokėjimo inicializavimo schema bei trumpinių paaiškinimai (2.1 pav.)



2.1 pav. Mokėjimų inicializavimo schema. Šif() – skliausteliuose šifruota informacija, id – vartotojo identifikacinis numeris, sum – mokėjimo suma, IV1 ir IV2 – inicializavimo vektoriai, sig – elektroniniai parašai

Po mokėjimo inicializavimo vartotojas iššifruoja informaciją ir reikalingą mokėti sumą mato savo ekrane. Inicializavimo vektorius, atsiųstas serverio, laikinai saugoma atmintyje, kadangi to prireiks sekančio šifravimo metu.

Vartotojas šiuo metu jau mato sumą ekrane ir gali pasirinkti, mokėti ją ar ne. Jeigu pasirenkamas mokėjimo variantas, vartotojas šį mokėjimą patvirtina įvesdamas PIN kodą ir išsiųsdamas atsakymą į serverį. Atsakymo žinutę sudaro vartotojo PIN kodas, sekančios šifrogramos inicializavimo vektorius bei elektroninis vartotojo parašas, sukurtas pasinaudojant jo privačiuoju raktu. Pateikiama vartotojo atsakymo apie apmokėjimą schema (2.2 pav.)



2.2 pav. Apmokėjimo atsakymas, čia Šif() – šifruotas tekstas, esantis skliausteliuose, PIN – saugos kodas, IV3 – inicializavimo vektorius, atsakymas – rezultatas ar mokėjimas patvirtintas,sig – elektroninis parašas.

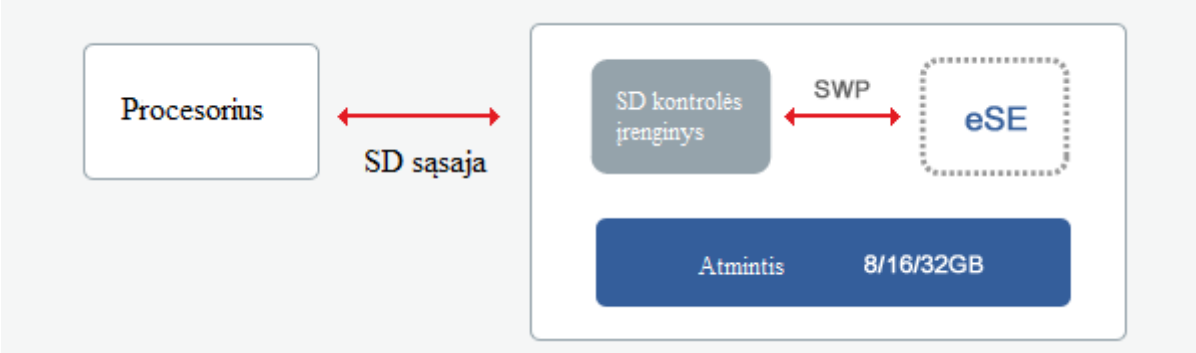
Tuo atveju, jeigu sutampa atėjusios žinutės duomenys su duomenimis esančiais duomenų saugykloje, tuomet vartotojui siunčiamas paprastas patvirtinimo apie mokėjimą pranešimas, priešingu atveju, žinutė būna apie apmokėjimo atmetimą. Iš vartotojo atėjusioje žinutėje turės sutapti vartotojo PIN saugos kodas bei šios informacijos elektroninis parašas.

Atlikus mokėjimą, pinigai iš vartotojo sąskaitos bus nuskaičiuojami, tačiau tai bus atliekama vidinėje sistemoje, arba pačioje banko duomenų bazėje, į ką nebus koncentruojamas dėmesys šiame darbe.

Toliau pateikiama struktūrinė schema, kurioje pavaizduotas mobiliojo telefono bei terminalo (kito mobilaus telefono su NFC funkcija) komunikacijos bei loginio veikimo principas per NFC duomenų perdavimo sąsają (2.3 pav).

2.3 pav. Struktūrinė mokėjimo sistemos mobilusis telefonas – NFC – kompiuteris schema

Šioje sistemoje ypatingas dėmesys skiriamas mobiliajame telefone esančiai programinei įrangai, kadangi ji gali būti labiausiai pažeidžiama mokėjimo sistemos dalis. Mobilieji telefonai gali tapti vienais labiausiai pažeidžiamų įrenginių šioje dalyje, todėl reikia užtikrinti privataus rakto saugojimo galimybes bei užtikrinti šio neprieinamumą tretiems asmenims. Šioje vietoje bus naudojama intelektualiosios atminties kortelės imitacija, tarsi viduje turinti kriptografinį mechanizmą. Šis panaudojimas neleis piktavaliams pasiekti vartotojo privataus rakto, priešingai negu tai būtų įmanoma padaryti, jeigu jis būtų saugomas telefono atmintyje atba tiesiog kortelėje. Nors mobiliuose telefonuose naudojama "Android" programinė įranga neturėtų leisti priėjimo prie duomenų kitoms programoms, išskyrus tai, kuriai jie yra skirti, tačiau kasdien atrandama daug apėjimo būdų. Intelektualiosios kortelės veikimo su integruotu saugos elementu (angl. *Embedded Security Element, eSE*) schema 2.4 pav.



2.4 pav. Intelektualiosios kortelės su integruotu saugos element struktūrinė schema

Programinė įranga kreipsis į imituojamą integruotą saugos elementą tik tuo atveju, jeigu bus būtinybė šifruoti pranešimą ar pasirašyti informaciją.

Programinė įranga kuriama panaudojant Android OS programinės įrangos kūrimo įrankį, kuris integruojamas į programą Eclipse. Kūrimo įrankis palaiko visus būtinus šios sistemos elementus: NFC duomenų perdavimo technologinį palaikymą bei duomenų skaitymą iš intelektualiosios kortelės.

Projektuojant programinės įrangos kodą, galima išskirti kelias klases, kurios atliks tam tikrus vaidmenis mobilioje aplikacijoje:

Šifravimo klasė – užtikrins duomenų šifravimą bei dešifravimą atėjusios informacijos. Tai pat ištrins iš operatyvinės atminties įvestus saugos PIN kodus, bet tokius šifravimo ar dešifravimo raktus, kad jų negalėtų perimti piktavaliai. Klasė kreipsis į imituojamą intelektualiąją kortelę dėl šifravimo rakto panaudojimo, tai pat dėl elektroninio parašo generavimo siunčiamai apmokėjimo informacijai.

Vartotojo sąsajos klasė – ši klasė bus naudojama iš vartotojo sąsajos įvedamiems duomenims apdoroti, tai pat atsiųstus duomenis pateikiant į vartotojo sąsają. Didelės reikšmės ši klasė programinėje įrangoje neužima, kadangi pati sąsaja planuojama minimali siekiant neapkrauti pačios programos ir iki minimum sumažinti galimų klaidų bei pažeidžiamumų skaičių.

Kuriant programinę įrangą Android operacinei sistemai, būtina suteikti tam tikras teises pačiai programai naudotis telefono resursais (angl. *permissions*). Šiuo atveju yra būtini du resursai:

NFC – leidimas programinei įrangai komunikuoti su kitais NFC įrenginiais.

WRITE_EXTERNAL_STORAGE – leidimas rašyti ir skaityti iš intelektualiosios atminties kortelės.

Be viso šito daugiau leidimų, tokių kaip naudojimasis internetu, programoje nebus leidžiami, dėl to sumažėja rizika jog šią aplikaciją pažeis piktavalis.

2.2. Realizacijos aprašymas panaudojant intelektualiąją kortelę ir NFC

Vartotojo privatusis DSA raktas bus saugomas imituojamoje intelektualioje kortelėje esančioje kriptografinio mechanizmo atmintyje [17]. Siekiant panaudoti šią sistemą mokėjimams, privačiojo rakto saugojimas turi atitikti FIPS 140 saugumo standarto reikalavimus. Šis standartas turi 4 saugumo lygius, iš kurių mokėjimo sistemai palaikyti reikalingas ne mažesnis kaip 3 lygis. Daugelis intelektualių kortelių palaiko šį standartą.

Šiuo atveju šios kortelės palaiko tokias funkcijas kaip šifravimas, raktų generaviamas, sertifikatų palaikymas bei importavimas ir t.t. Tai daugiau negu pakanka mokėjimo sistemai suprojektuoti. Atlikus tyrimą išsiaiškinta, jog dauguma kriptografiją palaikančių intelektualiųjų kortelių naudoja šiuos kriptografinius mechanizmus:

AES – 254 bit
Triple-DES – 112 ir 168 bit
RSA PKCS #1.5 su parašo generavimu ir patvirtinimu – 1024 ir 2048 bit
DSA PKCS #1.5 su parašo generavimu ir patvirtinimu – 1024 ir 2048 bit

Realiai raktai saugomi specialioje microSD kortelės atmintyje SST (angl. Security Storage). Šioje atmintyje saugomi tokie elementai kaip raktų poros, nuolatiniai simetriniai raktai, konfigūracijos informacija, PIN kodai.

Intelektualiajai kortelei palaikyti mobiliojo telefono operacinės sistemos versijos reikalavimai nėra keliami, todėl praktiškai visos versijos gali palaikyti šias atminties korteles. Nėra ir specialių reikalavimų mikro SD kortelėse esančiu kriptografinių mechanizmų palaikymui, todėl naudojama “Android” operacinės sistemos versija v4.0.3 bus pakankama.

NFC technologija mobiliajame telefone bus realizuojama “Android ” programinėje įrangoje. Minimalūs reikalavimai palaikyti NFC technologiją yra SDK v9 [18], tačiau norint pilnai palaikyti funkcionalumą, reikalinga SDK v10 versija mobiliajame telefone.

Mobilusis telefonas, kuriam bus kuriama sistema, turi Android 5.0.0 versiją, kas atitinka SDK v15 versiją, todėl šis telefonas pilnai palaikys NFC technologiją. Terminalo funkciją atliekančiame telefone veikia 4.0.3 Android versija.

2.3. Raktų generavimas “OpenSSL” pagalba

“OpenSSL” atviro kodo programinė įranga [19] siūlo programinės įrangos apsaugos sprendimus. Šie sprendimai kuriami savanoriškai prisidedančių prie šio projekto programuotojų.

Yra siūlomi 2 variantai asimetrinės kriptografijos mechanizmų:

RSA – naudojamas pasirašymui bei šifravimui. Generuojant raktus galima pasirinkti privataus rakto slaptažodį, tam, jog nebūtų įmanoma jo perskaityti net perėmus, tačiau nėra rekomenduojama jeigu bus naudojama kartu su sertifikatu. Privataus rakto slaptažodis gali kliudyti pačio rakto panaudojime. Komandinėje eilutėje RSA raktus galima sugeneruoti:

```
openssl genrsa -out privkey.pem 2048
```

Rakto ilgis pagal rekomendacijas mokėjimo sistemoms yra mažiausiai 2048 bitai, trumpesnio ilgio raktai laikomi nesaugiais arba bus nesaugūs artimiausioje ateityje (priklauso nuo interpretavimo).

DSA – naudojamas elektroninių dokumentų pasirasymui. Skirtumas nuo RSA yra pastebimas greitaveikoje: informacijos parašo generavimas atliekamas ilgiau negu RSA atveju, tačiau parašo patvirtinimas yra daug greitesnis, todėl galima naudoti atskirą raktą. Kadangi serverio dalyje galimi dideli informacijos srautai, todėl parašo tikrinimo laiko sutrumpinimas yra itin svarbus, iš kitos pusės vartotojas mažai pajus greitaveikos sumažėjimą, kadangi apmokėjimai neatliekami dažnai. Beto, dviejų skirtingų raktų kombinacija šifravimui ir pasirašymui daug labiau užtikrins saugumą ir sumažins informacijos klastojimo galimybes.

Pirmiausia gauti DSA raktų parametrus:

```
openssl dsaparam -out dsaparam.pem 2048
```

Tai pat kaip ir RSA raktų atveju, rekomenduojamas ne mažesnis negu 2048 bitų raktų ilgis. Toliau iš sugeneruotų parametrų gaunami raktai:

```
openssl gendsa -out privkey.pem dsaparam.pem
```

Tai pat nerekomenduojama privačiojo rakto apsaugoti papildomu slaptažodžiu, kadangi tai komplikuos šifravimo mechanizmus.

Pagal mokėjimo sistemos koncepciją, vartotojas negalės pats sau generuoti raktų porų ir taip pasirašinėti ir šifruoti informacijos. Dėl papildomo saugumo, vartotojas gaus jau paruoštą intelektualiąją kortelę, kurioje bus sugeneruoti privatūs bei viešieji raktai. Vartotojas gaudamas intelektualiąją kortelę patvirtins savo tapatybę.

2.4. Intelektualiosios kortelės konfigūravimas ir raktų kėlimas

Kaip ir minėta prieš tai, raktų įkėlimo procesas bus vykdomas prieš vartotojui gaunant atminties intelektualiąją kortelę. Šio projekto atveju imituojant intelektualiąją kortelę raktai bus įrašomi į specialius metodus, kurie šifruos ir pasirašys paduodamus parnešimus. Priežastis to yra ta, jog mokėjimus prižiūrintis juridinis vienetas pilnai turėtų pačio sugeneruotas raktų poras ir niekas kitas negalėtų jų pasiekti ar kuriant suklastoti, tai pat dėl to jog būtų patvirtinama vartotojo tapatybė šią kortelę atsiimant.

Kortelės šifravimo algoritmas pirmajam pranešimui iš vartotojo puses nustatomas kaip ir anksčiau numatyta RSA 2048 bit, elektroniniui parašui sugeneruoti DSA 2048 bit rakto formatas [20].

2.5. NFC terminalo dalies programos veikimo aprašymas aprašymas

Terminalas, kurio funkciją atliks kitas, NFC technologiją palaikantis telefonas, bus skirtas komunikuoti su mobiliuoju telefonu perduodant informaciją NFC technologija. Technologija palaiko 2 galimus komunikavimo tipus [21]:

1. skaitymo ir rašymo, kuomet aktyvus skaitytuvas komunikuoja su pasyvia NFC kortele.
2. P2P (angl. Peer-to-peer) duomenų perdavimo abiejomis kryptymis komunikavimo tipas.

Šiuo atveju naudojamas tik P2P komunikavimo tipas, kadangi reikia priimti ir išsiųsti informaciją abiejomis kryptimis. NFC naudos SNEP protokolą [22] (angl. *Simple NDEF Exchange*), kurio pagalba duomenys paprastai gali būti perduodami ir priimami iš abiejų įrenginių, palaikančių NFC technologiją.

Mobilusis telefonas bus naudojamas kaip atsiskaitymo priemonė, skirta mokėjimams atlikti. Jame esantis metodas, imituojantis intelektualios kortelės veikimą su kriptografiniu mechanizmu funkcionuos kaip vartotojo autentifikavimo įrankis. Tai pat mobiliojo telefono atsiskaitymų programinė įranga leis parodyti pinigų sumą, kurią reikia apmokėti, bei įvesti saugos PIN kodą, patvirtinant vieną iš autentifikavimo faktorių.

Šiuo atveju terminalas atliks tik asmens identifikavimo funkciją, parodys ar suvesti duomenys yra teisingi, ar galima atlikti mokėjimą ir siųs atsakymo pranešimus į mobiliojo telefono programinę įrangą. Sujungtas per NFC terminalas užmegs ryšį su telefonu.

2.6. Išvados

Suprojektuota NFC mokėjimų sistema, kuris leis prilietus telefoną atlikti mokėjimą panaudojant mobilųjį telefoną bei jame esančią programinę įrangą. Programinė įranga naudos dviejų faktorių autentifikavimą: vartotojo įvedamas PIN kodas bei imituojamoje intelektualioje kortelėje esantis kriptografinis mechanizmas, kuris leis šifruoti bei pasirašyti informaciją bei apsaugos ją nuo piktavališko panaudojimo.

Numatyta, jog imituojamoje intelektualioje kortelėje esantys vartotojo duomenys bus sukurti ir įkelti dar prieš jam gaunant pačią kortelę, o atsiimti galės tik pateikęs savo asmens dokumentą. Taip užtikrinama, jog duomenų siųsti bendro naudojimo tinklais nereikės, todėl eliminuojamas piktavalio žmogus viduryje (angl. “man-in-the-middle”) atakos tipas, kuris leistų perimti ir panaudoti duomenis savais tikslais.

NFC technologija pati neapsaugo siunčiamų duomenų, todėl sistema suprojektuota taip, jog siųstų šifruotus duomenis ir juos panaudoti galėtų tik ta šalis, kuriai jie yra siunčiami. Bendrojos paslapties apsikeitimui pasirinkta RSA 2048 bitų viešojo rakto šifravimo technologija, šiuo metu pilnai užtikrinanti duomenų saugumą ir galėsianti užtikrinti tai artimiausioje ateityje. Kiti siunčiami duomenys bus šifruojami AES 256 bitų raktu.

Duomenų skaitmeninio parašo generavimui bus naudojamas DSA 2048 bitų raktas, kuris pasižymi didesne greitaveika nei RSA ir yra pakankamai patikimas, tai yra atitinka mokėjimo sistemoms skirtus reikalavimus.

Mobiliojo telefono programa su terminale (šiuo atveju kitame mobiliajame telefone) esančia programine įranga komunikuos apsikeisdamos šifruotais pranešimais, kurių viduje bus tam tikra identifikacinė informacija.

Iš programinės įrangos projektavimo pusės bus atsižvelgta į programinės įrangos skirstymą į klases, kurių kiekviena turės savo konkrečius tikslus ir veikimo principo apibrėžimus. Programinė įranga bus kuriama naudojant integruotą “Android” įrankį “Eclipse” programoje, Java programavimo kalba.

Programinė įranga naudos 2 leidimus, kurie gali būti apibrėžti kaip leidimai naudotis mobilaus telefono resursais: leidimas naudotis NFC technologija bei leidimas naudotis išorine atmintini, šiuo atveju tai yra imituojama intelektualioji kortelė su integruoti kriptografiniu mechanizmu. Visi kiti leidimai bus draudžiami dėl saugumo sumetimų.

Mobilusis telefonas ir NFC skaitytuvas komunikuos naudojant vieną iš NFC technologjos siūlomų komunikacijos tipų, šiuo atveju naudojamas tik P2P komunikavimo tipas, kadangi reikia priimti ir išsiųsti informaciją abiejomis kryptimis. NFC naudos NDEF protokolą, kurio pagalba duomenys paprastai gali būti perduodami ir priimami iš abiejų įrenginių, palaikančių NFC technologiją.

3. NFC MOKĖJIMO SISTEMOS KLIENTAS - TERMINALAS PROTOTIPO REALIZAVIMAS IR TYRIMAS

3.1. Įvadas

Išpopuliarėjus mobiliesiems telefonams ir kitiems nešiojamiesiems įrenginiams, tai pat sparčiai plintant mobilioms aplikacijoms bei pritaikant kitus įrenginius tinkančiais bendrauti su telefonais, atsiranda būtinybė juos pritaikyti vis reiklesniems verslininkų bei pačių vartotojų norams. Pradėjus telefonus naudoti naršant internete, skaitant elektroninį paštą arba tiesiog bendraujant socialiniuose tinkluose, galimybės panaudoti išmaniuosius telefonus prasiplėtė iki tiesioginių apkomėjimų už prekes ar paslaugas, nenaudojant atsiskaitymo kortelių ar grynųjų pinigų.

Vieni pirmųjų atsiskaitymų buvo paremti tiesiog internetinėmis mokėjimo sistemomis, kuriomis prisijungus per telefone veikiančias naršykles buvo galima apmokėti. Tačiau tai vistiek reikalavo turėti mokėjimo kortelę, kurios duomenis reikia suvesti, arba būti užsiregistravus kokioje nors mokėjimo sistemoje.

Šiais atvejais pats mokėjimo procesas nebuvo greitas, bei plintant piktavališkoms mokėjimo programoms buvo galimybė prarasti duomenis arba pinigus. Tobulėjant mobiliesiems telefonams į juos buvo diegiama vis daugiau įvairių įrenginių, palengvinančių darbą. Vienas jų – NFC, skirtas greitai perduoti duomenis itin mažu atstumu tarp dviejų įrenginių.

Ši technologija greitai paplito, ją tai pat imta naudoti ir atsiskaitymams. Keletas analizės dalyje išvardintų technologijų veikia panašiai, tačiau turi skirtumų. Išanalizavus šių sistemų apsaugos mechanizmus, tiek, kiek galima jų rasti internete ir kiek jie yra paviešinti, sugalvota pasiųlyti patobulinimus informacinių technologijų duomenų saugojimo srityje.

Išanalizuoti galimi vartojo autentifikacijos būdai panaudojant mobiliuosius telefonus, ir pasirinkta imituoti intelektualią atminties kortelę su specialia kriptografine sistema jos viduje. Panaudojant ši mechanizmą, galima užtikrinti jog vartotojo privatusis raktas bus saugiai paslėptas ir neprieinamas išorinėms telefono aplikacijoms. Šiuo atveju galima pasirašinėti duomenis nebijant, jog parašas bus suklastotas, o gaunama informacija bus perskaityta kažkur anksčiau.

Projektinėje dalyje suprojektuotas informacijos šifravimo ir pasirašymo mechanizmas, aprašyti reikalavimai programinei įrangai, tai pat sudaryta mokėjimo duomenų apsikeitimo struktūrinė schema.

Prototipo realizavimo dalyje numatoma suprogramuoti mobiliają aplikaciją, kuri leistų perdavinėti šifruotus duomenis NFC technologija. Visi duomenų perdavimo schema ir programinė įranga turi atitikti tarptautinius informacinių technologijų saugumo standartus.

3.2. Programinės įrangos kūrimas

“Android” operacinės sistemos programinė įranga kuriama naudojant “Android Studio” programavimo programą, skirtą veikti Windows operacinėje sistemoje. Visa programinė įranga kuriama nuo pradžių, panaudojant tik standartines ir kūrėjų platinamas nemokamas bibliotekas, kurios praplėčia aplikacijos funkcionalumą.

Tai pat įtraukiamos būtinos “android.nfc” bibliotekos, tokios kaip NdefMessage, NfcRecord, NfcAdapter, NfcAdapter. CreateNdefMessageCallback, NfcEvent [23] . Jos visos yra būtinos užmezgant ryšį tarp mobilaus telefono ir terminal bei perdavinėjant duomenis per NFC įrenginį.

Kuriant programinę įrangą Android operacinei sistemai, būtina suteikti tam tikras teises pačiai programai naudotis telefono resursais (angl. *permissions*):

Android.permission.NFC – leidimas programinei įrangai naudoti NFC įrenginį esantį telefone.

WRITE_EXTERNAL_STORAGE – leidimas rašyti ir skaityti iš intelektualiosios atminties kortelės.

Nustatoma minimali SDK versija kurią programinė įranga palaikys. API 9 versija palaiko ribotą NFC funkcionalumą ir suteikia teises į tik į NDEF perduodamų žinučių tipą. Jokie kiti nustatymai įėjimo/išėjimo operacijose nėra prieinami. API 10 suteikia skaitymo/rašymo galimybę, o API 14 versija suteikia galimybę nesunkiai siųsti NDEF pranešimus į kitus įrenginius naudojant Android Beam metodus NDEF pranešimams.

3.3. Klasės

Visa programinė įranga išskirtyta į kelias pagrindines klases, kurios yra išskaidytos į atsakingas už tam tikras veiklas. Tuo siekiama supaprastinti pačios įrangos kūrimą bei tolimesnę jos priežiūrą bei plėtojimą [24].

Realizuojant programinės įrangos veikimą, galima išsiskiria kelios klasės, kurios atlieka tam tikrą vaidmenį mobilioje aplikacijoje:

Šifravimo klasės – užtikrina duomenų šifravimą bei iššifravimą atėjusios į mobilųjį telefoną informacijos. Tai pat ištrins iš operatyvinės atminties įvestus PIN kodus, bet tokius šifravimo ar dešifravimo raktus, slaptą informaciją, kad jų negalėtų perimti piktavaliai. Klasė kreipsis į imituojamą intelektualiąją kortelę dėl šifravimo raktų panaudojimo, tai pat dėl elektroninio parašo generavimo siunčiamai apmokėjimo informacijai.

Vartotojo sąsajos klasė – pagrindinė programinės įrangos klasė. Jos pagalba bus išvedami duomenys į vartotojo telefono ekraną. Dėl “Android” operacinės sistemos saugumo, duomenis

galima atvaizduoti ekrane, keisti grafinės sąsajos elementus ir kitaip koreguoti vaizdinę informaciją tik pačioje grafinės sąsajos klasėje, arba specialiai tam iškvietus vartotojo sąsajos procesą. Ši klasė tai pat atlieka vaidmenį užmezgant ryšį NFC duomenų perdavimo būdu tarp mobilaus telefono ir NFC terminalą atstojančio mobilaus telefono. Uždavinys yra perduoti informaciją abiejomis kryptimis tarp komunikuojančių šalių, tai pat sudėlioti priimtą informaciją į tam skirtus kintamuosius, kurie vėliau bus panaudoti vartotojui pateikiant informaciją ar ją toliau naudojant

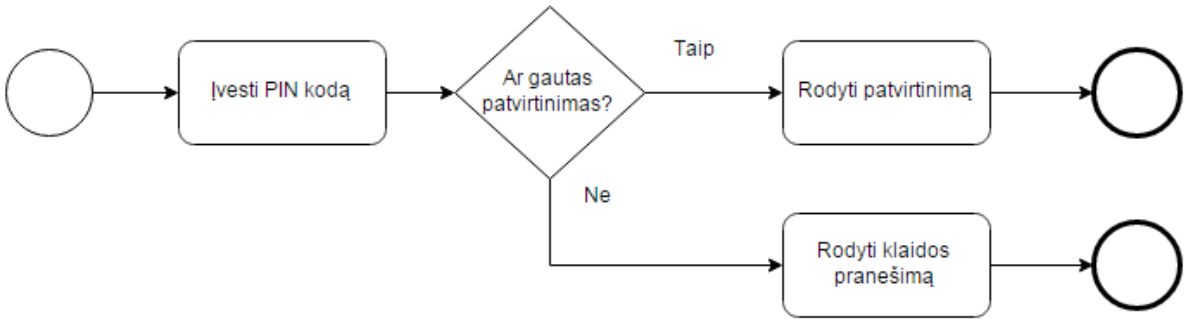
Pačią sąsajos klasę sudaro funkcija, nuskaitanti ir siunčianti PIN kodą į šifravimo klasę. Nėra leidžiama įvesti daugiau negu 6 simbolius. Tai pat rezultato (apmokėta, neapmokėta) atvaizdavimas ekrane, kuriame vartotojas gali matyti, koks yra pačio mokėjimo rezultatas.

3.4. Grafinė vartotojo sąsaja

Grafinė vartotojo sąsaja sumodeliuota kaip minimalistinių funkcijų turinti programa, realiai leidžianti tik įvesti vartotojui žinomą PIN kodą bei NFC srauto iniciavimu patvirtinti įvestą informaciją, kuri vidinių funkcijų pagalba apdorojama, užšifruojama vidiniais apsaugos mechanizmais bei perduodama per NFC įrenginį.

Priklausomai nuo gauto rezultato, kuris apdorojus viską serverio dalyje yra siunčiamas atgal vartotojui į programinę įrangą, skirtingai atvaizduojama ekrane gaunama informacija. Esant klaidos, negalimo ryšio, ryšio sutrikimo atveju arba trukstant kitų duomenų ar net pinigų vartotojui bus parodoma, jog operacija yra negalima atlikti. Priežingu atveju, jeigu mokėjimas įvykdomas sėkmingai, ekrane pasirodo pranešimas jog apmokėjimas yra įvykdymas sėkmingai.

Pagrindinis grafinės sąsajos funkcijos bei galimi vartotojo veiksmai ir skirtingos informacijos pateikimo diagram pateikiama:



3.1 pav. Grafinės sąsajos veiksmai po patvirtinimo

Minimalistinis vartotojo veiksmų scenarijus leidžia greitai bei aiškiai pateikti, kokie veiksmų rezultatai gaunami. Esant klaidos pranešimo veiksmui vartotojas ekrane gali pamatyti standartinį klaidos pranešimą, nepasakantį dėl ko negalima atlikti mokėjimo. Kitu atveju galima

klaidos pranešimus išskaidyti į grupes, kurios nurodys, ar dėl pinigų trūkumo, dėl ryšio praradimo ar dėl duomenų paketų iškraipymo arba informacijos nevientisumo įvyko klaida.

Pati grafinė sąsaja yra paprasta: vos įjungus programą, ji iškarto laukia NFC susijungimo. Matomas besisukantis laukimo indikatorius bei vartotojui pateikiama tekstinė informacija jog programa laukia.

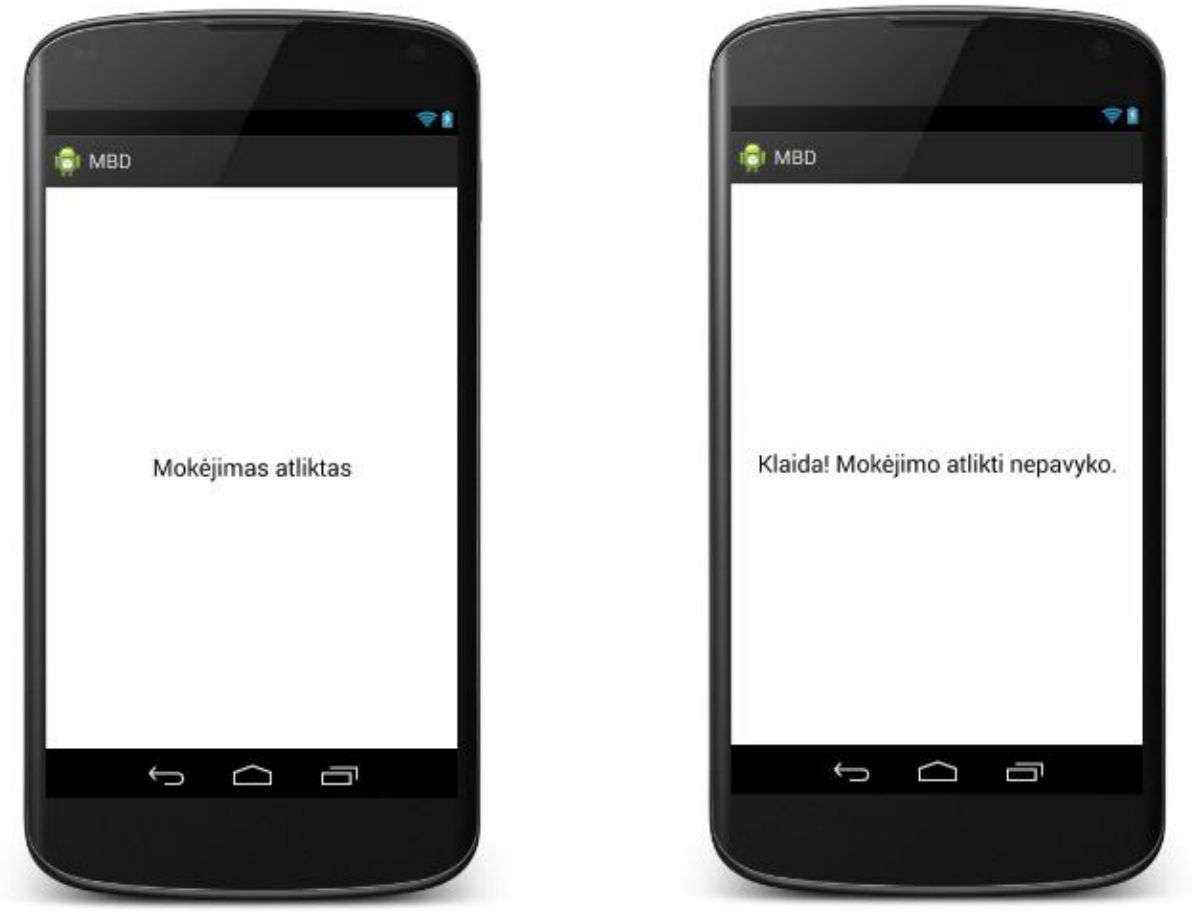
3.2 pav. Programos laukimo indikacija

Kuomet užmezgamas programinės įrangos ryšys su NFC skaitytuvu, nusiunčiama šifruota informacija, kurioje yra vartotojo identifikacinis numeris, AES 256 bitų raktas bei inicializavimo vektorius. Jeigu serverio dalyje patikrinta informacija yra teisinga, ši yra gražinama su mokėjimo suma, kuria pateikia vartotojo mobiliojo telefono ekrane. Vartotojas mato informaciją ir gali atšaukti mokėjimą tiesiog išjungdamas programą, taip nutraukdamas sudarytą sesiją tarp programos ir serverio dalies. Atliekant sekantį mokėjimą, būtų generuojamas kitas AES 256 bitų raktas, taip užtikrinant jog piktavalis arba net pats vartotojas negalės klastoti mokėjimų.

3.3 pav. Mokėjimo sumos atvaizdavimas ir patvirtinimo laukimas

Jeigu mokėjimo suma tenkina vartotoją, jis įvesdamas PIN kodą ir iniciavus NFC duomenų perdavimą sutinka, jog mokėjimas būtų vykdomas ir iš jo saskaitos būtų nuskaičiuojama atvaizduojama pinigų suma. Šiuo atveju visa informacija yra šifruojama kartu su PIN kodu, pasirašoma ir per NFC siunčiama į terminalo dalį, kur patikrinamas informacijos autentiškumas.

Esant teisingiems duomenims vartotojas išvys pranešimą jog mokėjimas atliktas, priešingu atveju pasirodys pranešimas jog įvyko klaida ir mokėjimas nebus atliktas.



3.4 pav. Atlikto mokėjimo patvirtinimas arba klaidos atvaizdavimas

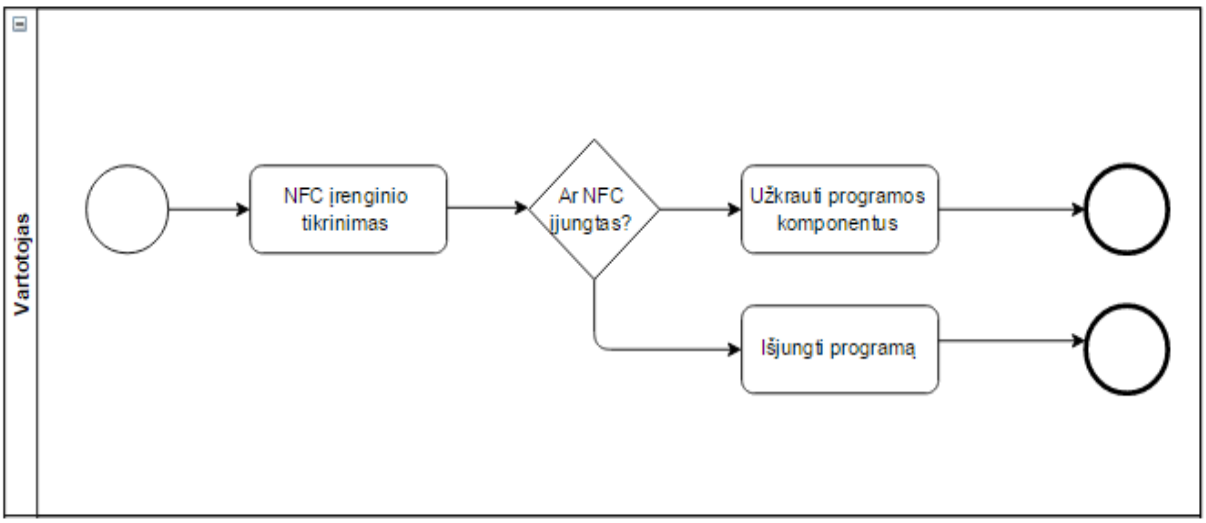
Ši paprasta grafinė sąsaja leidžia paprastai vykdyti mokėjimus, neapkraunant vartotojo bereikalinga informacija ar įvairių veiksmų, siekiant atlikti mokėjimą, gausa. Minimalistinė sąsaja tai pat sumažina klaidų ar programos spragų tikimybę.

3.5. Programos veikimo mechanizmas

3.5.1. NFC įrenginio aptikimas

Be vartotojo grafinės sąsajos programoje yra vykdomos vidinės funkcijos, atsakingos už NFC įrenginio veikimo tikrinimą, duomenų šifravimą ir iššifravimą, tai pat įvedimo ir išvedimo funkcijoms atlikti.

NFC įrenginys yra būtinas šioje programinėje įrangoje. Realiai programa neveiks tol, kol NFC nebus aktyvuotas mobiliajame telefone. Šiuo atveju programa tik ją įjungus patikrins, ar įrenginys veikia. Jeigu jis nebus aptiktas, programa bus priverstinai išjungta. NFC aktyvavimo patikrinimo schema pavaizduota 3.5 pav.



3.5 pav. NFC įrenginio aptikimo schema

NFC gali neįsijungti dėl dviejų priežasčių: pačio įrenginio gedimo arba tiesiog jo nebuvimo telefone. Pačią programą įrašyti į telefoną be NFC galima, tačiau jos nebus įmanoma naudoti tuo tikslu, kokiam ji buvo sukurta.

3.5.2. Duomenų saugumo užtikrinimo mechanizmai

Duomenų apsaugai nuo jų perėmimo, nuskaitymo ir klastojimo užtikrinti yra naudojama hibridinė duomenų apsaugos sistema. Šiuo atveju žodis hibridinė panaudotas tam, jog sistema apima dviejų faktorių autentifikavimo sistemos sudarymą bei elektroninio parašo panaudojimą siunčiamiems duomenims. Visa duomenų apsaugos esmė yra išlaikyti duomenis konfidencialius ir autentiškus perduodant juos iš “Vartotojo” į “Terminalą” ir atvirkščiai.

Vartotojas savo mobiliajame telefone inicijuoja mokėjimo procedūą. Po mokėjimo inicializavimo vartotojas gautą informaciją iššifruoja ir reikalingą mokėti sumą mato savo ekrane.

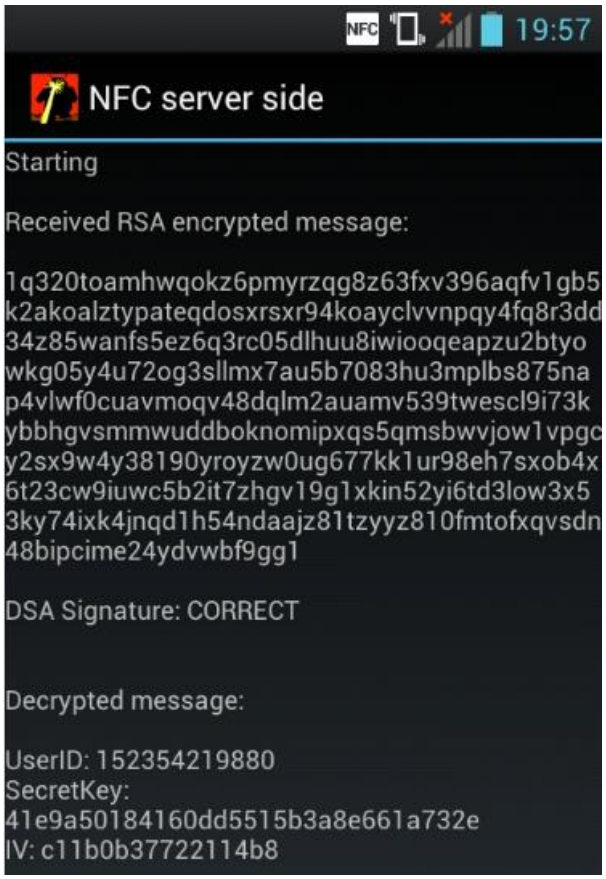
Vartotojas šiuo metu jau mato sumą ekrane ir gali pasirinkti, mokėti ją ar nemokėti. Jeigu pasirenkamas apmokėjimo variantas, vartotojas jį patvirtina įvesdamas saugos PIN kodą ir inicijuodamas kitą NFC duomenų perdavimo sesiją persiunčia informaciją į terminalą. Atsakymo žinutę sudaro vartotojo saugos kodas bei elektroninis vartotojo parašas, sukurtas pasinaudojant jo privačiuoju DSA raktu.

Tuo atveju, jeigu sutampa atėjusios žinutės duomenys su duomenimis esančiais duomenų saugykloje (šiuo atveju mobiliajame telefone - terminale), tuomet vartotojui siunčiamas paprastas patvirtinimo apie mokėjimą pranešimasarba pranešimas apie atmestą mokėjimą. Iš vartotojo atėjusioje žinutėje turės sutapti vartotojo saugos kodas bei patikrinamas šios informacijos parašas.

Kriptografijai ir elektoniniam pasirašymui yra naudojami RSA ir DSA raktai kurie atitinkamai pasižymi savo greitaveika šifruojant arba pasirašant duomenis.

3.5.3. NFC terminalo dalies realizacija

NFC terminalo (su mokėjimo mobiliąja programėle) dalis realizuota kitame mobiliajame telefone. Šiuo atveju to visiškai pakanka atvazduojant priimtą NFC duomenų srautą. Gauti duomenys, tyrimo tikslais, atvaizduojami ekrane (3.6 pav).



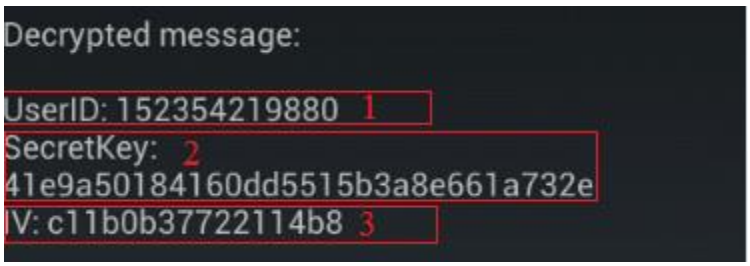
3.6 pav. Duomenys, atvaizduoti kitame mobiliajame telefone

Terminalo funkciją imituojančiame mobiliajame telefone visų pirma atvaizduojama RSA viešuoju raktu užšifruota šifrograma, pasirašyta vartotojo DSA raktu. Pirmiausia ši šifrograma yra iššifruojama, atrenkama jos viduje esanti informacija. Pagal vartotojo identifikacinį numerį patikrinamas visos šifrogramos autentiškumas.

Jeigu šifrograma yra autentiška, galima laikyti, jog vartotojas yra identifikuotas ir galima testi mokėjimo procesą. Visa užšifruoto pranešimo informacija yra (3.7 pav.):

- Vartotojo identifikacinis numeris (angl. User ID) (1)

- Slaptasis raktas (angl. SecretKey) (2)
- Inicializavimo vektorius (IV) (3)

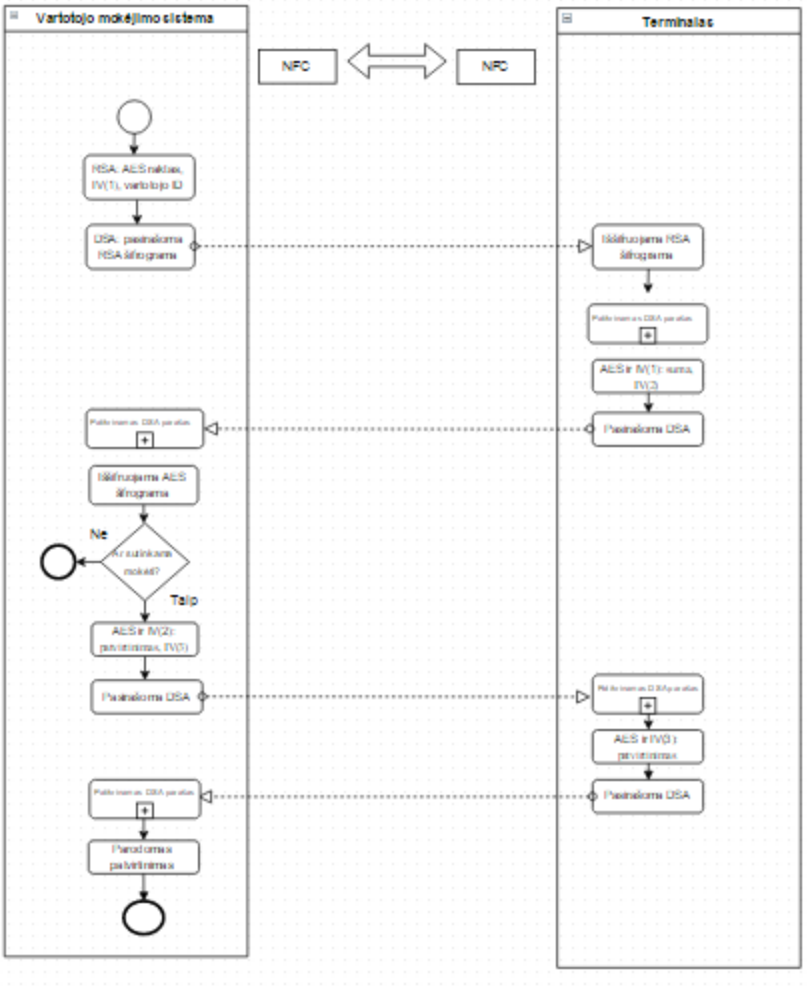


3.7 pav. RSA šifrogramos viduje esanti informacija

Perduodamoje informacijoje yra simetrinis raktas, kuris bus naudojamas tolesniai AES šifrogramų komunikacijai [25]. Tai pat siunčiamas ir inicializavimo vektorius, kurį teminalo dalis galės panaudoti siunčiamam pranešimui. Nors teoriškai nėra būtina, tačiau paslėpus inicializavimo vektorių, tikimybė jog šifrograma bus atskleista, ženkliai sumažėja.

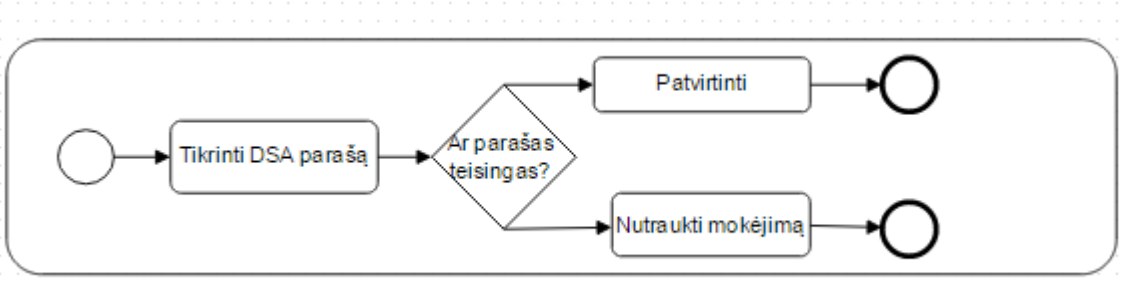
3.5.4. Duomenų perdavimo per NFC diagrama

Diagramoje (3.8 pav.) pavaizduota, kokie duomenys yra perduodami NFC kanalu. Kiekvienas pranešimas yra šifruojamas ir pasirašomas skaitmeniniu parašu.



3.8 pav. NFC kanalu perduodamų duomenų diagrama

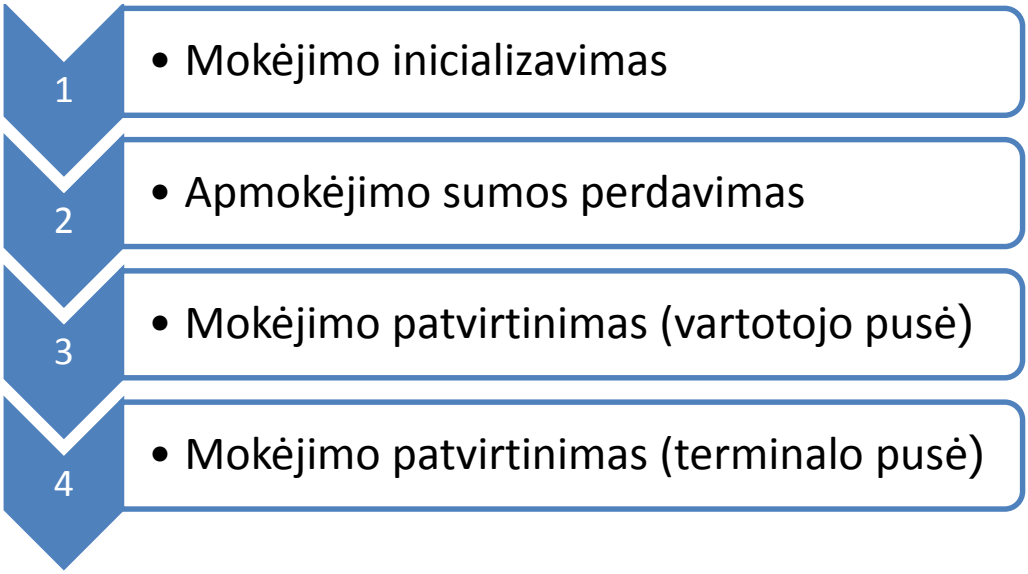
Diagramoje (3.9 pav.) pateikiami sub-procesai veikimo principu yra vienodi, skiriasi tik kuriuo DSA raktu yra tikrinamas parašas (priklausomai vartotojo ar terminalo).



3.9 pav. DSA rakto tikrinimo procesas

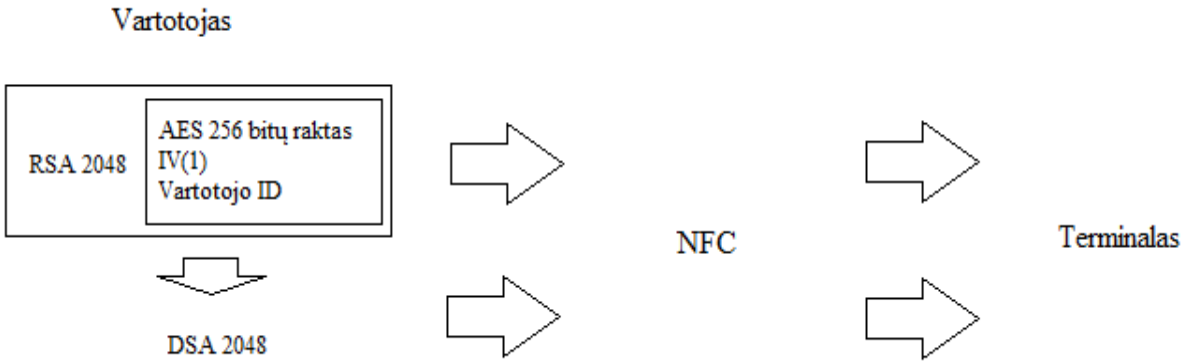
Šiuo atveju, jeigu DSA parašai įrodo siuntėjo tapatybę ir informacija yra iššifruojama, mokėjimas gali būti patvirtintas. Priešingu atveju, mokėjimas yra nutraukiamas.

Išskaidžius NFC duomenų srautus į atskirus blokus, gaunama supaprastinta schema, parodanti kiekvieno iš duomenų srautų reikšme duomenų perdavime (3.10 pav.).



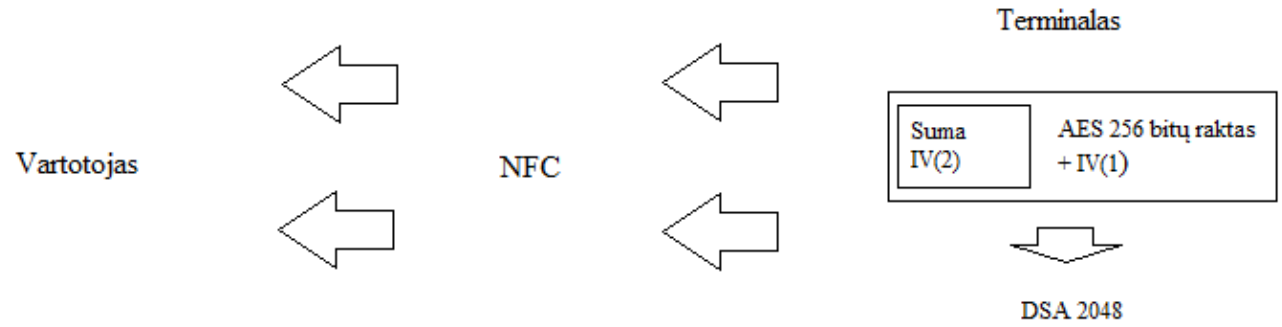
3.10 pav. Supaprastinta duomenų perdavimo schema

Mokėjimo realizacijoje pirmuoju žingsniu vartotojas inicijuoja mokėjimą, siųsdamas savo identifikacinį numerį, sugeneruotą AES 256 bitų raktą bei inicializavimo vektorių, kuris bus panaudotas sekančiame žingsnyje, terminalui siunčiant informaciją į vartotojo telefoną (3.11 pav).



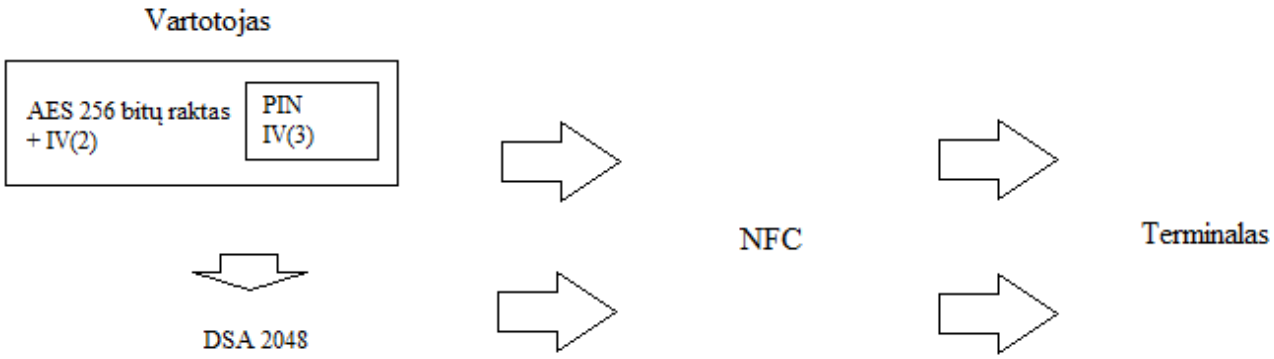
3.11 pav. Vartotojo siunčiama identifikacinė informacija, parašas ir bendroji paslaptis

Terminalui gavus informaciją, ji yra iššifruojama, patikrinamas šifrogramos autentiškumas, tuomet mokėjimo suma su kitu inicializavimo vektoriumi siunčiami vartotojui (3.12 pav.). Šis pranešimas pasirašomas terminalo (arba serverio).



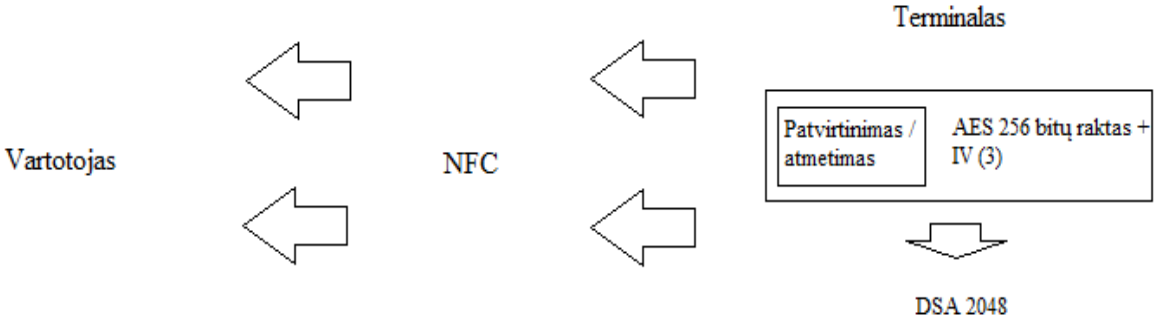
3.12 pav. Terminalo siunčiama suma ir sekantis inicializavimo vektorius

Vartotojas gavęs informaciją, pamato sumą ekrane. Jeigu yra nusprendžiama mokėti, suvedamas PIN kodas ir inicijuojamas NFC perdavimas. Tai įvykdoma priglaudžiant telefoną prie NFC terminalo. Į terminalą siunčiamas vartotojo PIN kodas ir incializavimo vektorius (3.13 pav).



3.13 pav. Vartotojo siunčiamas PIN kodas su inicializavimo vektoriumi

Jeigu vartotojo PIN kodas teisingas, terminalas šifrograma atsiunčia patvirtinimą. Priešingu atveju, atsiunčia pranešimą apie klaidos pranešimą (3.14 pav.).



3.14 pav. Terminalo mokėjimo patvirtinimas/atmetimas

3.6. NFC mokėjimo sistemos klientas - terminalas tyrimas

Tyrimo metu skaičiuojama ir analizuojama kriptografinių mechanizmų greitaveika. Kadangi darbo metu sukurtas prototipas naudoja RSA 2048 šifravimą bendrosios paslapties apsikeitimui, AES 256 duomenų šifravimui ir DSA 2048 elektroniniam pasirašymui, palyginimui naudojamas analizės dalyje aprašytas NFC-SEC protokolas. NFC-SEC protokolas naudoja elipsinių kreivių Diffi-Hellman 192 bitų raktų apsikeitimo mechanizmą, eksperimento metu realizuotą panaudojus "Spongy Castle" biblioteką [26], o šifravimui AES 128 bitų raktą.

Tyrimui sukurta atskira programos dalis, skirta skaičiuoti laikus kuomet kuriami šifravimo raktai, atliekami kriptografijos veiksmai, generuojami ir tikrinami elektroniniai parašai.

Pirmuoju atveju atliekamas tyrimas kriptografinių mechanizmų, naudojamų šiame projekte:

- Kriptografijai - AES 256 bitų
- Bendros paslapties apsikeitimui - RSA 2048 bitų
- Elektroninio parašo generavimui - DSA 2048 bitų

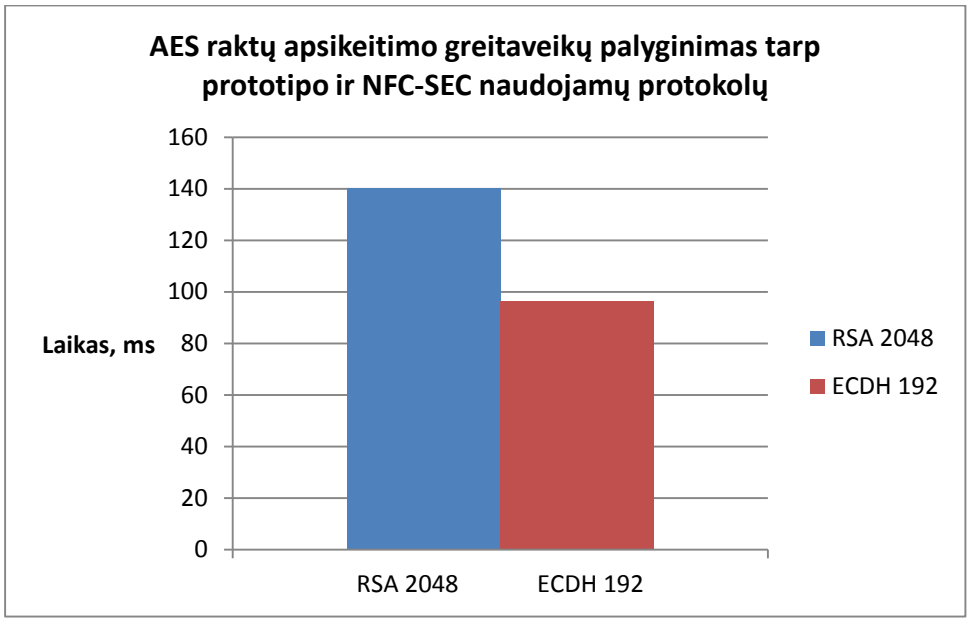
Antruoju atveju išmėginami kriptografiniai mechanizmai, naudojami NFC-SEC protokole:

- Kriptografijai – AES 128 bitų
- Bendros paslapties apsikeitimui – ECDH 192 bitų

Gauti rezultatai pateikiami 1-oje lentelėje.

1 lent. Prototipo ir NFC-SEC protokolo greitaveikų apskaičiavimas ir palyginimas

		Laikas, skirtas operacijai atlikti, ms	
		Prototipas	NFC-SEC
Bendrosios paslapties apsikeitimas	RSA 2048 bitų	140.23	-
	ECDH 192 bitų	-	96.2
Duomenų užšifravimas	AES 256 bitų	2.36	-
	AES 128 bitų	-	1.49
Duomenų iššifravimas	AES 256 bitų	5.31	-
	AES 128 bitų	-	2.73
Elektroninio parašo generavimas	DSA 2048	73.57	-
Elektroninio parašo tikrinimas	DSA 2048	3.13	-



3.15 pav. AES raktų apsikeitimo greitaveikų palyginimas

Tyrimo metu apskaičiuotos greitaveikos apsikeičiant AES raktus (3.15 pav). Šiame darbe sukurtas prototipas naudoja RSA 2048 raktą, kuriuo perduoda AES 256 bitų bendrąją paslaptį. Rezultatai palyginami su NFC-SEC protokole naudojamais kriptografiniais mechanizmais apsikeičiant numatytąjį 128 bitų raktą.

Prototipo atveju skaičiuojant bendrą rakto apsikeitimo laiką naudotos tokios dedamosios:

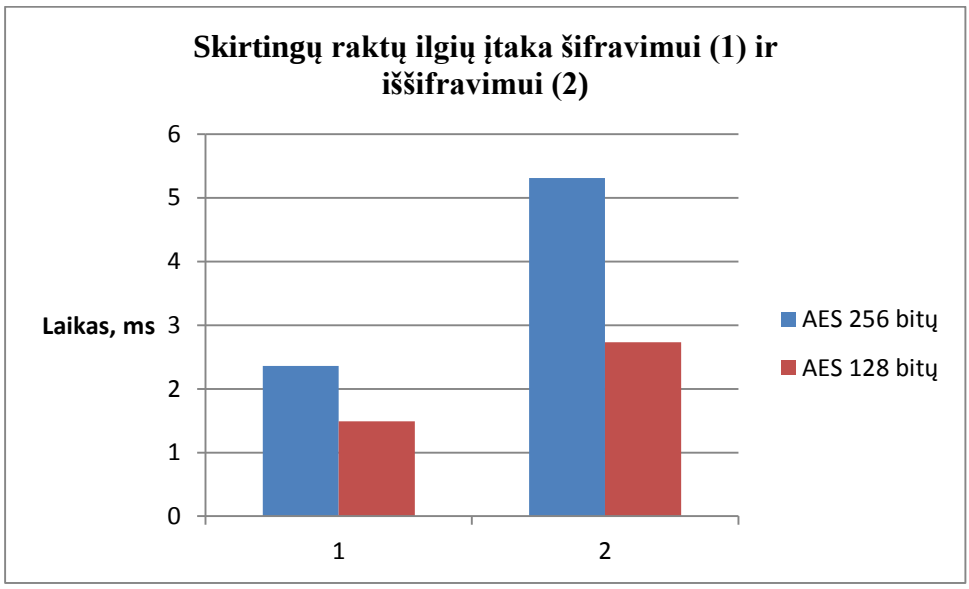
- AES rakto generavimas
- Šifravimas RSA viešuoju raktu
- Iššifravimas RSA privačiuoju raktu
- Elektroninio parašo generavimas
- Elektroninio parašo tikrinimas

2 lent. Prototipo AES simetrinio rakto apsikeitimo operacijų laikų dedamosios

Operacija	Laikas, ms
AES rakto generavimas	22.1
Šifravimas RSA viešuoju raktu	16.9
Iššifravimas RSA privačiuoju raktu	24.53
Elektroninio parašo generavimas	73.57
Elektroninio parašo tikrinimas	3.13
Viso:	140.23

Iš šių rezultatų matyti (2 lent), jog prototipo atveju bendras AES rakto apsikeitimas užtrunka ilgiau negu NFC-SEC protokolo atveju, tačiau terminalo dalyje atliekamos tik iššifravimo ir elektroninio parašo tikrinimo operacijos, todėl serveris apkraunamas mažiau.

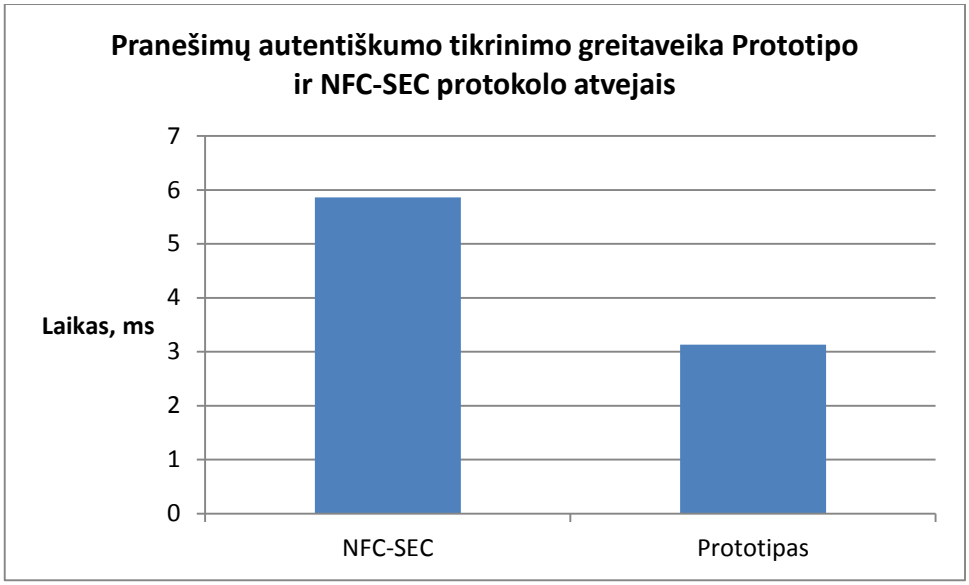
Duomenų šifravimo ir iššifravimo laikai naudojant skirtingus AES raktus (128 bitų NFC-SEC atveju ir 256 bitų prototipo) pateikiami 3.16 pav. Grafiko horizontalioje ašyje „1“ reiškia duomenų užšifravimo laiko trukmę, „2“ – duomenų iššifravimo laiko trukmę.



3.16 pav. Skirtingų AES raktų palyginimas šifruojant ir iššifruojant informaciją

Iš rezultatų matyti, jog skirtumas nedidelis užšifruojant informaciją, tačiau jis atsiranda kuomet pranešimą reikia iššifruoti, tuomet AES 128 bitų raktas turi akivaizdų pranašumą.

Sukurto prototipo atveju elektroninis parašas siunčiamas kartu su šifrograma, o naudojant NFC-SEC protokolą, elektroninį parašą tektų įdėti į patį pranešimą, kurį prieš tikrinant jo autentiškumą, reikėtų iššifruoti. Pateikiamas grafikas skaičiavimų, kiek laiko truktų patikrinti pranešimų autentiškumus abiem atvejais (3.17 pav.). Kadangi NFC-SEC vartotojo autentifikacijos nenaudoja, palyginimui bus pasitelktas tas pats DSA 2048 raktu sugeneruotas parašas.



3.17 pav. Pranešimų autentiškumo tikrinimo greitaveika prototipo ir NFC-SEC protokolo atvejais

Iš grafiko matosi, jog patikrinti gauto pranešimo autentiškumą prototipo atveju užtrunka trumpiau. Tai yra todėl, jog pirmiausia tikrinamas elektroninis parašas, o tik nustačius jog jis teisingas, iššifruojamas pranešimas. NFC-SEC atveju pirmiausia pranešimą tektų iššifruoti, o tik poto tikrinti jo autentiškumą.

Bendroji sistemos greitaveika gali būti paskaičiuojama įvertinus šifravimo, iššifravimo, elektroninio parašo generavimo bei jo patikrinimo laikus, tai pat duomenų perdavimo laikus pagal NFC ryšio greitaveiką atskiriems duomenų blokams. Visų kriptografinių funkcijų atlikimas užtrunka 163,24 ms, tuo tarpų šių duomenų perdavimas NFC ryšio kanalu bendrai užtrunka 1140 ms. Šios sistemos bendra greitaveika yra 1303,12 ms.

Į bendrosios greitaveikos skaičiavimą neįtrauktas asmens PIN kodo įvedimo laikas, kadangi tai labai subjektyvus dalykas ir priklauso nuo pačio asmens. Tai pat greitaveika tarp terminalo ir bankinės sistemos ar mokėjimo paslaugos tiekėjo.

3.7. Išvados

Realizuotas mokėjimo sistemos klientas – terminalas prototipas, veikiantis panaudojus 2 mobiliuosius telefonus su Android operacinėmis sistemomis. Viename iš telefonų realizuota vartotojo mobilioji aplikacija, skirta atlikti mokėjimams, kitame – terminalo darbą atliekanti programinė įranga.

Realizuotas duomenų apsikeitimo per NFC protokolas ir duomenų saugumą bei autentiškumą užtikrinančios kriptografinės funkcijos.

Programinėje įrangoje šifravimui ir duomenų pasirašymui naudojamos kriptografinės funkcijos:

RSA 2048 – simetrinio rakto apsikeitimui

AES 256 – siunčiamų duomenų šifravimui

DSA 2048 – elektroniniam parašui

Informacijos saugos sprendimo prototipo realizavimo dalyje apibrėžta programinės įrangos veikimo schema pagal kurią vykdomi mokėjimai. Pateikiamos visos galimų vartotojų veiksmų baigtys – esant mokėjimui arba atmetus jį. Pateikiamas grafinis mokėjimo scenarijaus atvaizdavimas.

Aprašoma programinės įrangos dalies struktūra, naudojamos klasės. Jos skirtos išskirti atskiras programos funkcionalumo dalis, atskirti kriptografinius mechanizmus, vartotojo grafinės sąsajos funkcijas, įvesties ir išvesties perdavimo mechanizmus.

Aprašomas NFC įrenginio aptikimo ir valdymo programinėje įrangoje algoritmas. Nurodomi keli variantai, kaip programa turi elgtis neesant NFC įrenginiui arba kuomet jis yra išjungtas.

Tyrimo metu apskaičiuoti ir palyginti procedūrų laikai tarp siūlomos sistemos ir NFC-SEC protokolo:

- Apskaičiuota, jog raktų apsikeitimui siūlomos sistemos metu užtrunka ilgiau negu naudojantis NFC-SEC raktų apsikeitimui naudojamomis kriptografinėmis funkcijomis.
- Šifravimas naudojant NFC-SEC protokole naudojamą AES 128 bitų raktą užtrunka trumpiau negu siūlomoje sistemoje AES 256 bitų raktu.
- Tikrinant siunčiamo pranešimo autentiškumą siūlomos sistemos atveju elektroninio parašo tikrinimas užtrunka trumpiau, negu tokias pačias elektroninio parašo funkcijas panaudojus NFC-SEC protokole

4. IŠVADOS

Darbo analizės metu ištirti reikalavimai elektroninėms mokėjimo sistemoms, aprašyti reikalavimai jų programinei įrangai, slaptažodžių saugojimui, vartotojo autentifikacijos funkcijos integruojant papildomus elementus.

Integruojant IK kartu su ARK protokolu suprojektuota terminalinė mokėjimo sistema, realizuota mobiliuosiuose telefonuose ir užtikrinanti informacijos konfidencialumą, integralumą ir autentiškumą.

Suprojektuota sistema yra patogi vartotojui, nes reikalauja tiktai PIN kodo įvedimo vartotojo mobiliajame telefone.

Pasirašytas ir realizuotas metodas, kaip su ARK realizuoti vartotojo autentifikaciją, panaudojant IK autentifikacijos funkcijas.

Ištirta raktų apsikeitimo protokolo tarp vartotojo mobilaus telefono ir terminalo greitaveika, simetrinio šifravimo greitaveika ir pranešimų autentiškumo tikrinimo greitaveika. Nustatyta, jog siūlomos terminalinės mokėjimo sistemos greitaveika yra 1,303 sekundės ir nesudaro nepatogumų vartotojams.

Tyrimo metu nustatyta, jog NFC-SEC protokolu atveju simetrinių raktų apsikeitimas bei šifravimas šiais raktais yra greitesni negu naudojantis siūlomos mokėjimo sistemos naudojamų kriptografinių funkcijų metodais.

Siūloma autentifikavimo sistema yra greitesnė nei NFC-SEC protokolo autentifikacijos sistema, palyginimui jai pritaikius tas pačias autentifikacijos funkcijas.

LITERATŪRA

[1] W. K. Chung-wei Lee, Advances in Security and Payment Methods for Mobile Commerce, University of North Dakota, USA, 2004.

[2] S. Kungpisdan, Securing Mobile Payments: Modelling, Design, and Analysis, 2010.

[3] B. S. a. T. K. Niels Ferguson, Cryptography Engineering, Wiley, 2012.

[4] „Official PCI Security Standards Council," PCI Security Standards Council , 2006. [Tinkle]. Available: https://www.pcisecuritystandards.org/. [Kreiptasi 05 11 2013].

[5] „PayPass Tap & Go," MasterCard, [Tinkle]. Available: http://www.mastercard.ca/paypass-tapgo.html. [Kreiptasi 12 12 2013].

[6] „Google Wallet," Google, [Tinkle]. Available: http://www.google.lt/wallet/#. [Kreiptasi 12 12 2013].

[7] „Contactless," Visa Europe, [Tinkle]. Available: http://www.visaeurope.com/receiving-payments/contactless. [Kreiptasi 12 12 2013].

[8] „Mokipay," Mokipay Europe, 2014. [Tinkle]. Available: http://www.mokipay.com/. [Kreiptasi 14 12 2013].

[9] „Tapsis E-klasteris," Tapsis, 2013. [Tinkle]. Available: www.e-klasteris.lt/tag/tapsis/. [Kreiptasi 12 2013].

[10] „NFC-SEC, NFCIP-1 Security Services and Protocol, Cryptography Standard using ECDH and AES," ECMA, 2008.

[11] „NFC-SEC NFCIP-1 Security Services and Protocol," ECMA, Geneva, 2013.

[12] „NFC-SEC-01, NFC Cryptography, Standard using ECDH and AES," ECMA, Geneva, 2010.

[13] S. C. Allience, The Mobile Payments and NFC Landscape: A U.S. Perspective, Princeton Junction, NJ, 2011.

[14] E. Technologies, „PCI Mobile Payment Acceptance Security Guidelines for Developers," PCI Security Standards Council, 2012.

[15] P. S. S. Council, „Payment Card Industry (PCI)," PCI Security Standards Council, 2014.

[16] J. Zaloker, „Near Field Communication," Arrow Electronics, 2012.

[17] Corsec, „TrustedFlash v1.0 - microSD FIPS 140-2," Fairfax, 2009.

[18] „NFC Basics," Google, [Tinkle]. Available: https://developer.android.com/guide/topics/connectivity/nfc/nfc.html. [Kreiptasi 25 03 2015].

[19] O. SSL, „www.openssl.org," [Tinkle]. Available: www.openssl.org. [Kreiptasi 1 9 2014].

[20] P. Dhawan, „Performance Comparison," Microsoft, 10 2002. [Tinkle]. Available: https://msdn.microsoft.com/en-us/library/ms978415.aspx. [Kreiptasi 15 2 2014].

[21] Google, „Near Field Communication," Google, [Tinkle]. Available: https://developer.android.com/guide/topics/connectivity/nfc/index.html. [Kreiptasi 2 10 2014].

[22] M. Clark, „Java Library implements SNEP," 2012.

[23] N. Semiconductors, „How to build a NFC Application on Android," 2013.

[24] A. Q. Alliance, „Best Practice Guidelines," 2013.

[25] navneet88, 04 2015. [Tinkle]. Available: https://github.com/Pakhee/Cross-platform-AES-encryption/blob/master/Android/CryptLib.java.

[26] S. C. rtyley, 04 2015. [Tinkle]. Available: https://rtyley.github.io/spongycastle/.

5. PRIEDAI

5.1. Vartotojo sąsajos programinis kodas

```
package com.example.android.beam;

import android.app.Activity;
import android.content.Intent;
import android.nfc.NdefMessage;
import android.nfc.NdefRecord;
import android.nfc.NfcAdapter;
import android.nfc.NfcAdapter.CreateNdefMessageCallback;
import android.nfc.NfcEvent;
import android.os.Bundle;
import android.os.Parcelable;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

import java.io.UnsupportedEncodingException;
import java.nio.charset.Charset;
import java.security.InvalidAlgorithmParameterException;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.Signature;
import java.security.SignatureException;

import javax.crypto.BadPaddingException;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;


public class Beam extends Activity implements CreateNdefMessageCallback {
    NfcAdapter mNfcAdapter;
    TextView textView;
    int i=0;
    Signature signature;
    PrivateKey userPrivateKey;
    PublicKey serverPublicKey;
    String userID = "152354219880", AESkey, IV;
    boolean sent1 = false, sent2  = false;
    static String kaina;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        View suma = findViewById(R.id.suma);
        View sum = findViewById(R.id.sum);
```

```
    View ed = findViewById(R.id.editText);
    TextView textView = (TextView) findViewById(R.id.textView);

    suma.setVisibility(View.GONE);
    sum.setVisibility(View.GONE);
    ed.setVisibility(View.GONE);

    // CRYPTO

    try
    {
        String x = CryptLib.generateRandomIV(32);
        AESkey = CryptLib.SHA256(x, 32); //32 bytes = 256 bit
        IV = CryptLib.generateRandomIV(16); //16 bytes = 128 bit
    } catch (Exception e) {
        e.printStackTrace();
    }

    mNfcAdapter = NfcAdapter.getDefaultAdapter(this);
    if (mNfcAdapter == null) {
        Toast.makeText(this, "NFC is not available", Toast.LENGTH_LONG).show();
        finish();
        return;
    }
    mNfcAdapter.setNdefPushMessageCallback(this, this);
}

@Override
public NdefMessage createNdefMessage(NfcEvent event) {
    i++;
    String text = "Z2 sends: " +  i + "\n\n";
    crypto c = new crypto();
    if (!sent1)
    {
        String message = "";
        String encr = c.EncRsa(userID + ";;;;;" + AESkey + ";;;;;" + IV,serverPublicKey);
        message = encr;
        text = (
                message +
                "//////////" +
                c.signDSA(message, userPrivateKey, signature)
        );
        sent1 = true;
    }

    NdefMessage msg = new NdefMessage(
            new NdefRecord[] { createMimeRecord(
                    "application/com.example.android.beam", text.getBytes())
    });
    return msg;
}

@Override
```

```
public void onResume() {
   super.onResume();
   if (NfcAdapter.ACTION_NDEF_DISCOVERED.equals(getIntent().getAction())) {
      processIntent(getIntent());
   }
}

@Override
public void onNewIntent(Intent intent) {
   setIntent(intent);
}

void processIntent(Intent intent) {
   textView = (TextView) findViewById(R.id.textView);

   Parcelable[] rawMsgs = intent.getParcelableArrayExtra(
         NfcAdapter.EXTRA_NDEF_MESSAGES);
   NdefMessage msg = (NdefMessage) rawMsgs[0];
   textView.append(new String(msg.getRecords()[0].getPayload()));
   String mess = new String(msg.getRecords()[0].getPayload());
   String[] splitedsig = mess.split(";;;;;");
   String ciphersig = splitedsig[1];
   try {
      signature=Signature.getInstance("DSA");
   } catch (NoSuchAlgorithmException e) {
      e.printStackTrace();
   }
   try {
      signature.initVerify(serverPublicKey);
   } catch (InvalidKeyException e) {
      e.printStackTrace();
   }
   boolean sig = true;
   try {
      signature.update(mess.getBytes());
      if (!signature.verify(ciphersig.getBytes()))
         textView.append("\nSignature failed");
         sig = false;
   } catch (SignatureException e) {
      e.printStackTrace();
   }
   if (i==0 && sig == true)
   {
   setContentView(R.layout.main);
   View suma = findViewById(R.id.suma);
   View sum = findViewById(R.id.sum);
   View ed = findViewById(R.id.editText);
   TextView textView = (TextView) findViewById(R.id.textView);
   suma.setVisibility(View.VISIBLE);
   sum.setVisibility(View.VISIBLE);
   ed.setVisibility(View.VISIBLE);
      View te = findViewById(R.id.textView);
```

```
    String[] splited = mess.split(";;;;;");
    String cipher = splited[0];
    String encrypted = "";

    try {
       CryptLib c = new CryptLib();
       try {
          encrypted = c.decrypt(cipher,AESkey,IV);
       } catch (InvalidKeyException e) {
          e.printStackTrace();
       } catch (UnsupportedEncodingException e) {
          e.printStackTrace();
       } catch (InvalidAlgorithmParameterException e) {
          e.printStackTrace();
       } catch (IllegalBlockSizeException e) {
          e.printStackTrace();
       } catch (BadPaddingException e) {
          e.printStackTrace();
       }
    } catch (NoSuchAlgorithmException e) {
       e.printStackTrace();
    } catch (NoSuchPaddingException e) {
       e.printStackTrace();
    }
    String[] split = encrypted.split(";;;;;");
    kaina = split[0];
    i++;
}
if(i==1 && sig == true)
{
    setContentView(R.layout.main);
    View sum = findViewById(R.id.sum);
    View ed = findViewById(R.id.editText);

    sum.setVisibility(View.GONE);
    ed.setVisibility(View.GONE);
    String[] splited = mess.split(";;;;;");
    String cipher = splited[0];
    String encrypted = "";
    try {
       CryptLib c = new CryptLib();
       try {
          encrypted = c.decrypt(cipher,AESkey,IV);
       } catch (InvalidKeyException e) {
          e.printStackTrace();
       } catch (UnsupportedEncodingException e) {
          e.printStackTrace();
       } catch (InvalidAlgorithmParameterException e) {
          e.printStackTrace();
       } catch (IllegalBlockSizeException e) {
          e.printStackTrace();
       } catch (BadPaddingException e) {
          e.printStackTrace();
```

```
            }
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        } catch (NoSuchPaddingException e) {
            e.printStackTrace();
        }
        String[] splited2 = mess.split(";;;;;");
        String cipher2 = splited[0];
        if ( cipher2 == "1")
        {
            TextView textView = (TextView) findViewById(R.id.suma);
            textView.setText("Mokejimas atliktas");
        }
        else
        {
            TextView textView = (TextView) findViewById(R.id.suma);
            textView.setText("Klaida! Mokejimo atlikti nepavyko.");
        }
      }
   }

   public NdefRecord createMimeRecord(String mimeType, byte[] payload) {
      byte[] mimeBytes = mimeType.getBytes(Charset.forName("US-ASCII"));
      NdefRecord mimeRecord = new NdefRecord(
            NdefRecord.TNF_MIME_MEDIA, mimeBytes, new byte[0], payload);
      return mimeRecord;
   }
}
```

5.2. Serverio dalies programinis kodas

```
package com.example.android.beam;

import android.app.Activity;
import android.content.Intent;
import android.nfc.NdefMessage;
import android.nfc.NdefRecord;
import android.nfc.NfcAdapter;
import android.nfc.NfcAdapter.CreateNdefMessageCallback;
import android.nfc.NfcEvent;
import android.os.Bundle;
import android.os.Parcelable;
import android.widget.TextView;
import android.widget.Toast;
import java.nio.charset.Charset;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.Signature;
import java.security.SignatureException;


public class Beam extends Activity implements CreateNdefMessageCallback {
```

```
NfcAdapter mNfcAdapter;
TextView textView;
int i = 0 ;
PrivateKey ServerPrivateKey;
PublicKey UsersPublicKey;
String AESkey,IV;
String generatedIV, encrypted;
boolean received1 = false;
String message1, message2;
Signature signature;

@Override
public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.main);
   TextView textView = (TextView) findViewById(R.id.textView);
   mNfcAdapter = NfcAdapter.getDefaultAdapter(this);
   if (mNfcAdapter == null) {
      Toast.makeText(this, "NFC is not available", Toast.LENGTH_LONG).show();
      finish();
      return;
   }
   mNfcAdapter.setNdefPushMessageCallback(this, this);
}

@Override
public NdefMessage createNdefMessage(NfcEvent event) {
   i++;
   String text = message1 + ";;;;;Z2";
   if (i ==2)
   {
      text = "Z3" + message2;
   }

   NdefMessage msg = new NdefMessage(
         new NdefRecord[] { createMimeRecord(
               "application/com.example.android.beam", text.getBytes())
   });
   return msg;
}

@Override
public void onResume() {
   super.onResume();
   if (NfcAdapter.ACTION_NDEF_DISCOVERED.equals(getIntent().getAction())) {
      processIntent(getIntent());
   }
}

@Override
public void onNewIntent(Intent intent) {
   setIntent(intent);
}
```

```
void processIntent(Intent intent) {
   textView = (TextView) findViewById(R.id.textView);
   Parcelable[] rawMsgs = intent.getParcelableArrayExtra(
         NfcAdapter.EXTRA_NDEF_MESSAGES);
   NdefMessage msg = (NdefMessage) rawMsgs[0];
   if (!received1)
   {
      String message = new String(msg.getRecords()[0].getPayload());
      String[] splited = message.split("//////////");
      crypto c = new crypto();
      textView.append("Received RSA encrypted message:\n\n");
      textView.append(splited[2] + "\n");
      String decr = c.DecRsa(splited[2], ServerPrivateKey);
      String[] splited2 = decr.split(";;;;;");
      boolean sig = true;
      try {

         signature.update(splited[2].getBytes());
         if (!signature.verify(splited[1].getBytes()))
            textView.append("\nSignature failed");
         sig = false;
      } catch (SignatureException e) {
         e.printStackTrace();
      }
      textView.append("\nDSA Signature: CORRECT\n\n");
      textView.append("\nDecrypted message:\n\n");
      textView.append("UserID: " + splited2[0] +"\n");
      textView.append("SecretKey: " + splited2[1] +"\n");
      AESkey = splited2[1];
      textView.append("IV: " + splited2[2] +"\n\n");
      IV = splited2[2];
      textView.append("DSA Signature:\n");
      textView.append(splited[3]+"\n");
      received1 = true;

      generatedIV = CryptLib.generateRandomIV(16);
      message = "95.99" + ";;;;;" + generatedIV;

      textView.append("\n\nGenerated payment: " +splited[4]);
      textView.append("\nGenerated IV: " + generatedIV + "\n");
      try {
         CryptLib _crypt = new CryptLib();
         encrypted = _crypt.encrypt(message, AESkey, splited2[2]);
      } catch (Exception e) {
         // TODO Auto-generated catch block
         e.printStackTrace();
      }
      textView.append("\nEncrypted message: " + encrypted + "\n");
      message1 = encrypted;
      textView.append("\nPranesimas, kuris bus siunciamas i KLIENTA\n");
   }
}
```

```
    public NdefRecord createMimeRecord(String mimeType, byte[] payload) {
        byte[] mimeBytes = mimeType.getBytes(Charset.forName("US-ASCII"));
        NdefRecord mimeRecord = new NdefRecord(
                NdefRecord.TNF_MIME_MEDIA, mimeBytes, new byte[0], payload);
        return mimeRecord;
    }
}
```

5.3. Bendros programinių įrangų klasės

```
package com.example.android.beam;

import android.util.Base64;

import java.math.BigInteger;
import java.nio.charset.Charset;
import java.security.InvalidKeyException;
import java.security.KeyFactory;
import java.security.NoSuchAlgorithmException;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.SecureRandom;
import java.security.Signature;
import java.security.SignatureException;
import java.security.spec.InvalidKeySpecException;
import java.security.spec.PKCS8EncodedKeySpec;
import java.util.Arrays;

import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;

public class crypto {


    //AES

    private static final byte[] salt = { (byte) 0xA4, (byte) 0x0B, (byte) 0xC8,
            (byte) 0x34, (byte) 0xD6, (byte) 0x95, (byte) 0xF3, (byte) 0x13 };

    private static int BLOCKS = 128;

    //RSA
    Cipher cipher, cipher1;
    byte[] encryptedBytes, decryptedBytes;
    String encrypted, decrypted;

    public String EncryptRSA (String plain, PublicKey publicKey) throws
NoSuchAlgorithmException, NoSuchPaddingException, InvalidKeyException,
IllegalBlockSizeException, BadPaddingException
```

```
{
    cipher = Cipher.getInstance("RSA");
    cipher.init(Cipher.ENCRYPT_MODE, publicKey);
    encryptedBytes = cipher.doFinal(plain.getBytes());

    encrypted = bytesToString(encryptedBytes);
    return encrypted;

}

public String DecryptRSA (String result, PrivateKey privateKey) throws
NoSuchAlgorithmException, NoSuchPaddingException, InvalidKeyException,
IllegalBlockSizeException, BadPaddingException
{

    cipher1= Cipher.getInstance("RSA");
    cipher1.init(Cipher.DECRYPT_MODE, privateKey);
    decryptedBytes = cipher1.doFinal(stringToBytes(result));
    decrypted = new String(decryptedBytes);
    return decrypted;

}

public String EncRsa(String mess, PublicKey publicKey)
{
    String x = null;
    try {
        x = EncryptRSA(mess, publicKey);
    } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
    } catch (NoSuchPaddingException e) {
        e.printStackTrace();
    } catch (InvalidKeyException e) {
        e.printStackTrace();
    } catch (IllegalBlockSizeException e) {
        e.printStackTrace();
    } catch (BadPaddingException e) {
        e.printStackTrace();
    }
    return x;
}

public String bytesToString(byte[] b) {
    byte[] b2 = new byte[b.length + 1];
    b2[0] = 1;
    System.arraycopy(b, 0, b2, 1, b.length);
    return new BigInteger(b2).toString(36);
}

public  byte[] stringToBytes(String s) {
    byte[] b2 = new BigInteger(s, 36).toByteArray();
    return Arrays.copyOfRange(b2, 1, b2.length);
}
```

```
    public PrivateKey convertStringtoRSAPprivate(String key){

        byte [] clear = Base64.decode(key, Base64.DEFAULT);
        PKCS8EncodedKeySpec keySpec = new PKCS8EncodedKeySpec(clear);
        KeyFactory fact = null;
        try {
            fact = KeyFactory.getInstance("RSA");
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }
        PrivateKey priv = null;
        try {
            priv = fact.generatePrivate(keySpec);
        } catch (InvalidKeySpecException e) {
            e.printStackTrace();
        }
        Arrays.fill(clear, (byte) 0);
        return priv;
    }

    public String signDSA (String text, PrivateKey privateKey, Signature signature)
    {
        try {
            signature = Signature.getInstance("SHA1withRSA");
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }
        try {
            signature.initSign(privateKey, new SecureRandom());
        } catch (InvalidKeyException e) {
            e.printStackTrace();
        }
        byte[] message = text.getBytes();
        try {
            signature.update(message);
        } catch (SignatureException e) {
            e.printStackTrace();
        }
        byte[] sigBytes=null;
        try {
            sigBytes = signature.sign();
        } catch (SignatureException e) {
            e.printStackTrace();
        }
        String s = new String(sigBytes, Charset.forName("UTF8"));
        return s;
    }
}

package com.example.android.beam;

import android.util.Base64;
```

```
import java.io.UnsupportedEncodingException;
import java.security.InvalidAlgorithmParameterException;
import java.security.InvalidKeyException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.security.SecureRandom;

import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;

public class CryptLib {

    private enum EncryptMode {
        ENCRYPT, DECRYPT;
    }

    Cipher _cx;

    byte[] _key, _iv;

    public CryptLib() throws NoSuchAlgorithmException, NoSuchPaddingException {
        _cx = Cipher.getInstance("AES/CBC/PKCS5Padding");
        _key = new byte[32];
        _iv = new byte[16];
    }

    private String encryptDecrypt(String _inputText, String _encryptionKey,
                                  EncryptMode _mode, String _initVector) throws
UnsupportedEncodingException,
            InvalidKeyException, InvalidAlgorithmParameterException,
            IllegalBlockSizeException, BadPaddingException {
        String _out = "";
        int len = _encryptionKey.getBytes("UTF-8").length;

        if (_encryptionKey.getBytes("UTF-8").length > _key.length)
            len = _key.length;

        int ivlen = _initVector.getBytes("UTF-8").length;

        if(_initVector.getBytes("UTF-8").length > _iv.length)
            ivlen = _iv.length;

        System.arraycopy(_encryptionKey.getBytes("UTF-8"), 0, _key, 0, len);
        System.arraycopy(_initVector.getBytes("UTF-8"), 0, _iv, 0, ivlen);

        SecretKeySpec keySpec = new SecretKeySpec(_key, "AES");

        IvParameterSpec ivSpec = new IvParameterSpec(_iv);
```

```
        if (_mode.equals(EncryptMode.ENCRYPT)) {

            _cx.init(Cipher.ENCRYPT_MODE, keySpec, ivSpec);
            byte[] results = _cx.doFinal(_inputText.getBytes("UTF-8"));

            _out = Base64.encodeToString(results, Base64.DEFAULT);
        }
        if (_mode.equals(EncryptMode.DECRYPT)) {
            _cx.init(Cipher.DECRYPT_MODE, keySpec, ivSpec);

            byte[] decodedValue = Base64.decode(_inputText.getBytes(),
                    Base64.DEFAULT);
            byte[] decryptedVal = _cx.doFinal(decodedValue);

            _out = new String(decryptedVal);
        }
        System.out.println(_out);
        return _out;
    }
    public static String SHA256 (String text, int length) throws NoSuchAlgorithmException,
UnsupportedEncodingException {

        String resultStr;
        MessageDigest md = MessageDigest.getInstance("SHA-256");

        md.update(text.getBytes("UTF-8"));
        byte[] digest = md.digest();

        StringBuffer result = new StringBuffer();
        for (byte b : digest) {
            result.append(String.format("%02x", b));
        }
        if(length > result.toString().length())
        {
            resultStr = result.toString();
        }
        else
        {
            resultStr = result.toString().substring(0, length);
        }

        return resultStr;

    }
    public String encrypt(String _plainText, String _key, String _iv)
            throws InvalidKeyException, UnsupportedEncodingException,
            InvalidAlgorithmParameterException, IllegalBlockSizeException,
            BadPaddingException {
        return encryptDecrypt(_plainText, _key, EncryptMode.ENCRYPT, _iv);
    }

    public String decrypt(String _encryptedText, String _key, String _iv)
```

```
            throws InvalidKeyException, UnsupportedEncodingException,
            InvalidAlgorithmParameterException, IllegalBlockSizeException,
            BadPaddingException {
        return encryptDecrypt(_encryptedText, _key, EncryptMode.DECRYPT, _iv);
    }

    public static String generateRandomIV(int length)
    {
        SecureRandom ranGen = new SecureRandom();
        byte[] aesKey = new byte[32];
        ranGen.nextBytes(aesKey);
        StringBuffer result = new StringBuffer();
        for (byte b : aesKey) {
            result.append(String.format("%02x", b));
        }
        if(length> result.toString().length())
        {
            return result.toString();
        }
        else
        {
            return result.toString().substring(0, length);
        }
    }
}
```

5.4. Eksperimento metu naudota programa greitaveikos skaičiavimams

```
package com.example.app;

import android.os.Bundle;
import android.support.v7.app.ActionBarActivity;
import android.widget.TextView;

import java.io.UnsupportedEncodingException;
import java.security.InvalidAlgorithmParameterException;
import java.security.InvalidKeyException;
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.NoSuchAlgorithmException;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.Signature;
import java.security.SignatureException;
import java.text.DecimalFormat;
import java.text.NumberFormat;

import javax.crypto.BadPaddingException;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;

public class MainActivity extends ActionBarActivity {
```

```
Signature signature;
PrivateKey userPrivateKey, serverPrivateKey;
PublicKey userPublicKey, serverPublicKey;
String password = "123456",suma;
String userID = "152354219880", serverPrivate, userPublic, AESkey, IV;
boolean sent1 = false, sent2  = false;
TextView textView;

@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main);
   textView = (TextView) findViewById(R.id.textView);
   textView.setText("\nStarting\n\n");
   //RSA
   KeyPairGenerator keyPairGenerator = null;
   try {
      keyPairGenerator = KeyPairGenerator.getInstance("RSA");
   } catch (NoSuchAlgorithmException e) {
      e.printStackTrace();
   }
   crypto c = new crypto();
   keyPairGenerator.initialize(2048);
   KeyPair keyPair1 = keyPairGenerator.genKeyPair();
   serverPrivateKey = keyPair1.getPrivate();
   serverPublicKey = keyPair1.getPublic();
   serverPrivate = c.convertPrivateRSAtoString(serverPrivateKey);

   keyPairGenerator.initialize(2048);
   KeyPair keyPair = keyPairGenerator.genKeyPair();
   userPrivateKey = keyPair.getPrivate();
   userPublicKey = keyPair.getPublic();
   //DSA
   keyPairGenerator = null;
   try {
      keyPairGenerator = KeyPairGenerator.getInstance("RSA");
   } catch (NoSuchAlgorithmException e) {
      e.printStackTrace();
   }
   crypto c1 = new crypto();
   keyPairGenerator.initialize(2048);
   KeyPair keyPair2 = keyPairGenerator.genKeyPair();
   serverPrivateKey = keyPair2.getPrivate();
   serverPublicKey = keyPair2.getPublic();
   serverPrivate = c1.convertPrivateRSAtoString(serverPrivateKey);

   keyPairGenerator.initialize(2048);
   KeyPair keyPair3 = keyPairGenerator.genKeyPair();
   userPrivateKey = keyPair3.getPrivate();
   userPublicKey = keyPair3.getPublic();
   //
   String message = "zinute kuri bus nautojama matuojant laikus";
```

```
textView.append("\nMessage: "+ message + "\n\n");
String AESkeygentime = "";
try
{
   String x = CryptLib.generateRandomIV(16);

   long start = System.nanoTime();
   AESkey = CryptLib.SHA256(x, 16); //32 bytes = 256 bit
   long end = System.nanoTime();
   long estimate = (end - start) / 1000 ;
   NumberFormat formatter = new DecimalFormat("#0.00000");
   AESkeygentime = formatter.format((end - start));
   textView.append("\nAES keygen time: "+ estimate + "\n\n");
   IV = CryptLib.generateRandomIV(16); //16 bytes = 128 bit
} catch (Exception e) {
   e.printStackTrace();
}
String AESencrypttime = "";
String AESdecrypttime = "";
try {
   CryptLib cryptLib = new CryptLib();
   try {
      long start = System.nanoTime();
      String encrypted = cryptLib.encrypt(message,AESkey,IV);
      long end = System.nanoTime();
      long estimate = (end - start) / 1000 ;
      NumberFormat formatter = new DecimalFormat("#0.00000");
      AESencrypttime = formatter.format((end - start));
      textView.append("\nAES encrypt time: "+ estimate + "\n\n");

      start = System.nanoTime();
      String decrypted = cryptLib.decrypt(encrypted, AESkey, IV);
      end = System.nanoTime();
      estimate = (end - start) / 1000 ;
      formatter = new DecimalFormat("#0.00000");
      AESdecrypttime = formatter.format((end - start));
      textView.append("\nAES decrypt time: "+ estimate + "\n\n");

   } catch (InvalidKeyException e) {
      e.printStackTrace();
   } catch (UnsupportedEncodingException e) {
      e.printStackTrace();
   } catch (InvalidAlgorithmParameterException e) {
      e.printStackTrace();
   } catch (IllegalBlockSizeException e) {
      e.printStackTrace();
   } catch (BadPaddingException e) {
      e.printStackTrace();
   }
} catch (NoSuchAlgorithmException e) {
   e.printStackTrace();
} catch (NoSuchPaddingException e) {
   e.printStackTrace();
```

```
if (savedInstanceState == null) {
}
}

crypto rsa = new crypto();

long start = System.nanoTime();
String encRSA = rsa.EncRsa(message,serverPublicKey);
long end = System.nanoTime();
long estimate = (end - start) / 1000 ;
NumberFormat   formatter = new DecimalFormat("#0.00000");
String RSAencrypttime = formatter.format((end - start));
textView.append("\nRSA encrypt time: "+ estimate + "\n\n");

start = System.nanoTime();
String decRSA = rsa.DecRsa(encRSA,serverPrivateKey);
end = System.nanoTime();
formatter = new DecimalFormat("#0.00000");
estimate = (end - start) / 1000 ;
String RSAdecrypttime = formatter.format((end - start));
textView.append("\nRSA decrypt time: "+ estimate + "\n\n");


try {
   signature=Signature.getInstance("SHA1withRSA");
} catch (NoSuchAlgorithmException e) {
   e.printStackTrace();
}
try {
   signature.initSign(keyPair.getPrivate());
} catch (InvalidKeyException e) {
   e.printStackTrace();
}
try {
   signature.update(message.getBytes());
} catch (SignatureException e) {
   e.printStackTrace();
}


start = System.nanoTime();
String sigRSA = rsa.signRSA(message,serverPrivateKey,signature);
end = System.nanoTime();
estimate = (end - start) / 1000 ;
formatter = new DecimalFormat("#0.00000");
String signTime = formatter.format((end - start));
textView.append("\nDSA sign time: "+ estimate + "\n\n");

try {
   start = System.nanoTime();
   signature=Signature.getInstance("DSA");
   signature.initVerify(serverPublicKey);
```

```
    try {
        signature.update(message.getBytes());
        if (!signature.verify(sigRSA.getBytes()))
            textView.append("\nSignature failed");
        end = System.nanoTime();
        formatter = new DecimalFormat("#0.00000");
        estimate = (end - start) / 1000 ;
        String checksignTime = formatter.format((end - start));
        textView.append("\nDSA sign check time: "+ estimate + "\n\n");


    } catch (SignatureException e) {
        e.printStackTrace();
    }
} catch (InvalidKeyException e) {
    e.printStackTrace();
} catch (NoSuchAlgorithmException e) {
    e.printStackTrace();
}
```