KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

Vytautas Berankis

ATVIROS PLATFORMOS KOMUNIKACIJŲ UNIFIKUOTOS ARCHITEKTŪROS INTEGRACIJA Į VIRTUALŲ PRIVATŲ TINKLĄ

Baigiamasis magistro darbas

Vadovas
Prof. dr. Eligijus Sakalauskas

KAUNAS, 2015

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
KOMPIUTERIŲ KATEDRA

TVIRTINU
Katedros vedėjas
(parašas) Prof. dr. Algimantas Venčkauskas
(data)

ATVIROS PLATFORMOS KOMUNIKACIJŲ UNIFIKUOTOS ARCHITEKTŪROS INTEGRACIJA Į VIRTUALŲ PRIVATŲ TINKLĄ

Baigiamasis magistro darbas

Informacijos ir informacinių technologijų sauga (kodas 621E10003)

Vadovas
(parašas) Prof. dr. Eligijus Sakalauskas
(data)

Recenzentas
(parašas) Doc. dr. Kęstutis Paulikas
(data)

Projektą atliko
(parašas) Vytautas Berankis
(data)

KAUNAS, 2015

KAUNO TECHNOLOGIJOS UNIVERSITETAS

Informatikos fakultetas

(Fakultetas)

Vytautas Berankis

(Studento vardas, pavardė)

Informacijos ir informacinių technologijų sauga (kodas 621E10003)

(Studijų programos pavadinimas, kodas)

Baigiamojo projekto

„Atviros platformos komunikacijų unifikuotos architektūros integracija į virtualų privatų tinklą“

AKADEMINIO SĄŽININGUMO DEKLARACIJA

2015 m. gegužės 18 d.

Kaunas

Patvirtinu, kad mano **Vytauto Berankio** baigiamasis projektas tema „Atviros platformos komunikacijų unifikuotos architektūros integracija į virtualų privatų tinklą“ 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)*

Berankis, V. Atviros platformos komunikacijų unifikuotos architektūros integracija į virtualų privatų tinklą. Magistro baigiamasis projektas / vadovas Prof. dr. Eligijus Sakalauskas; Kauno technologijos universitetas, Informatikos fakultetas, Kompiuterių katedra.

Kaunas, 2015. 54 psl.

SANTRAUKA

Gamybai plečiantis vis didėja poreikis perteikti gamykloje surinktą informaciją, apie pagamintus produktus, bet kur internete. Vienas iš sprendimo būdų yra naudojant atviros platformos komunikacijų unifikuotą architektūrą (OPC UA), kuri leidžią perduoti duomenis iš programuojamų loginių valdiklių (PLC) į internetą.

Šio darbo praktinė realizacija yra OPC UA sistemos integracija į virtualų privatų tinklą (VPN). Atliekamas duomenų perdavimo greičių palyginimas tarp OPC UA serverio ir OPC UA kliento naudojant OPC UA sistemoje integruotais šifravimo algoritmais su VPN tinklo šifravimo algoritmais atitinkamai.

Sistema realizuota naudojant Java programavimo kalbą, Android SDK, Prosys OPC UA Java SDK. Virtualaus privataus tinklo sudarymui naudota programinę įrangą OpenVpn. Po tyrimo gauti rezultatai parodė, kad duomenų perdavimo greitis yra per VPN tinklą yra mažesnis ~20%.

Berankis, V. Open platform communications unified architecture‘s integration into virtual private network. Master’s thesis / supervisor Prof. Dr. Eligijus Sakalauskas; Department of Computer Science, Faculty of Informatics, Kaunas University of Technology.

Kaunas, 2015. 54 p.

SUMMARY

Production expansion is a growing need to convey the factory collected information about products manufactured anywhere on the Internet. One solution is to use open platform unified communications architecture (OPC UA), which allows you to transfer data from programmable logic controllers (PLC) to the Internet.

This work is a practical realization of OPC UA system integration into a virtual private network (VPN). The data transmission speed comparison between the OPC UA server and OPC UA client using the OPC UA system integrated encryption algorithms to the VPN encryption algorithms respectively.

The system is implemented using the Java programming language, the Android SDK Pros OPC UA Java SDK. Virtual private network used to establish software OpenVPN. After the investigation led to the result that the data rate is to a VPN network it is less ~ 20%

TURINYS

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

TERMINŲ IR SANTRUMPŲ ŽODYNAS

PLC	Programuojamas loginis valdiklis (Programming logic controller)
DCS	Paskirstytos valdymo sistemos (Distributed Control Systems)
WS	Žiniatinklio paslaugos (Web Service)
VPN	Virtualus privatus tinklas (Virtual Private Network)
URI	Suvienodintas resursų identifikatorius (Uniform Resource Identifier)
SC	Sertifikavimo centras
TCP	Transporto valdymo protokolas (Transport Control Protocol)
IP	Interneto protokolas (Internet Protocol)
OPC UA	Atviros platformos komunikacijų unifikuota architektūra (Open platform communications Unified Architecture)
XML	XML kalba duomenų struktūroms aprašyti
HTTP	Hipertekstų persiuntimo protokolas žiniatinklio duomenims persiųsti.
SOAP	Taisyklių rinkinys XML formato struktūrizuotiems pranešimams siųsti kompiuterių tinklais.

1. ĮVADAS

Technologijos vystosi labai sparčiai šiais laikais. Atsiranda vis daugiau ir daugiau galimybių. Su didesnėmis galimybėmis kyla vis daugiau pavoju, dėl to yra keliami vis didesni reikalavimai. Automatinės valdymo sistemos leidžia užtikrintai pasiekti greitą produkciją ir geriausią produkto kokybę.

Gamybai plečiantis 1996 metais buvo sukurta bendra automatikos sistema pavadinta „Objektų sujungimas ir įterpimas skirtas procesų valdymui („OLE (Object Linking and Embedding) for process control“). Tai buvo naujas standartas leidžiantis vartotojams ne tik dalintis informacija realiu laiku tarp programuojamų loginių valdiklių pagamintų skirtingų gamyklų, bet taip pat išsaugoti visą įvykių ir pavojų signalų istoriją išoriniame serveryje. Ši galimybė vartotojams leidžia stebėti visą gamybos proceso istoriją ir detaliai ją išanalizuoti. Ne tik matyti istoriją, bet taip pat suteikiama galimybė stebėti realiu laiku gaunamą produkciją vietiniame IT biure.

Po „OLE for process control“ standarto sukūrimo buvo įkurta „OPC konsorciumas“ („OPC Foundation“). OPC konsorciumas turi kelias pagrindines užduotis. Pirmoji užduotis išlaikyti „Objektų sujungimas ir įterpimas skirtas procesų valdymui“ standartą, o taip pat ir atlikti su jais tyrimus. Po atliktų tyrimų buvo sukurti nauji pavadinimai ir nauji standartai. Pavadinimas “(„OLE (Object Linking and Embedding) for process control“ buvo pervardintas į OPC. Antroji užduotis buvo sukurti principus ir taisykles naujam OPC standartui. Paskutiniai OPC grupės standartai buvo paskelbtas 2006 metais. Šios standartinės specifikacijos aprašo visus procesų valdymo sistemos taikymo aspektus tarp loginių programuojamų valdiklių (PLC) ir mažų paskirstyto valdymo sistemų (DCS) ir nesvarbu ar ši sistema bus nuolatinio veikimo ar dalinio veikimo. Ne taip kaip programuojamo loginio valdiklio – kompiuterio sistemos (PLC/PC), kurios turi keletą duomenų bazių ir jos tarpusavyje turi būti sujungtos, kad veiktų užtikrintai, ši sistema naudoja vieną bendrą duomenų bazę ir pažangiausius duomenų integravimo metodus, o tai leidžia realizuoti programines įrangas tokias kaip „pajunk ir paleisk“ („plug and play“), kurios leidžia taikomosioms programoms parašytoms skirtingomis kalbomis, įrašytoms į skirtingas platformas keistis įvairia informacija ne tik tarp PLC, bet taip pat ir tarp tokių taikomųjų programų, kaip MS Word, MS Excel ir panašių. Praeityje visos biuro taikomosios programos siūlydavo tik vieną paprastą funkciją nukopijuoti/įdėti tam, kad būtų nukopijuota informacija iš vieno dokumento į kitą. Be to šis veiksmas turėjo būti atliekamas pastoviai tam, kad duomenys būtų pastoviai atnaujinami.

Visos programuojamo loginio valdiklio – kompiuterio sistemose (PLC/PC) valdymo sistemos tinklas fiziškai nebuvo prijungtas prie interneto tam, kad užtikrinti saugumą ir niekas iš išorės negalėtų padaryti jokios įtakos procesų valdymui o taip padaryti žalos visai gamyklai. Sukūrus OPC, atsirado paklausa, kad visa gamyklos sistema būtų prijungta prie interneto. Su internetu atsirado ir naujų pavojų. Todėl vienas iš OPC specifikacijos standartų sukurtų 2006 buvo saugumui užtikrinti.

Pavadintas OPC saugumas (OPC Security). Be šio saugumo, būtų galima, nesunkiai pakenkti gamyklai: praradimas produkcijos, praradimas konfidencialios informacijos, įrangos sugadinimas ar net žmonių sužalojimai.

Po paskutinių OPC specifikacijos standartų paskelbimo OPC konsorciumas pristatė visiškai naują OPC unifikuotą architektūrą (OPC Unified Architecture). Po trijų metų plėtojimo ir specifikavimo OPC konsorciumas pristatė OPC UA specifikacijų standartus padalintus į dalis, kurios buvo padalintos į tris dalis: pagrindines, prieigos tipo ir naudingumo dalis.

OPC UA taikomosios programos gali buti vykdomos skirtingose vietose: tame pačiame kompiuteryje, ar per itin stipriai apsaugotą valdymo tinklą. Taip pat OPC UA klientai ir serveriai gali bendrauti žymiai atviresnėse aplinkose, tokiuose kaip internetas. Su interneto pagalba mes galima sumažinti IT kainas, taip pat gali būti padarytos patikimos atsarginės informacijos kopijos.

OPC UA antroji dalis yra OPC UA saugumo modelis, kuris apibrėžia kaip visi komponentai turėtų veikti vieni su kitais, kad būtų galima sumažinti atakų žalos riziką. Tai gali būti atliekama nustatant sistemos protektorius, nustatant sistemų pažeidžiamumą ir numatant atsakomąsias priemones. Taip pat greitas sistemos atstatymas, po sėkmingos atakos. OPC UA turėdamas daugiau saugumo savybių gali būti daug saugesnis nei OPC saugumas.

2. OPC UA SAUGUMO ANALIZĖ

2.1. Tyrimo objektas

Analizės metu bus susipažinta su OPC UA sistemos architektūra, išanalizuotas sistemos tinklo modelis, sistemos saugumo modelis, galimos saugumo grėsmės ir jų prevencija, viešoje erdvėje. Bus susipažinta su OPC UA sistemos funkcionalumu ir jos aplinka.

2.2. Tinklo architektūra

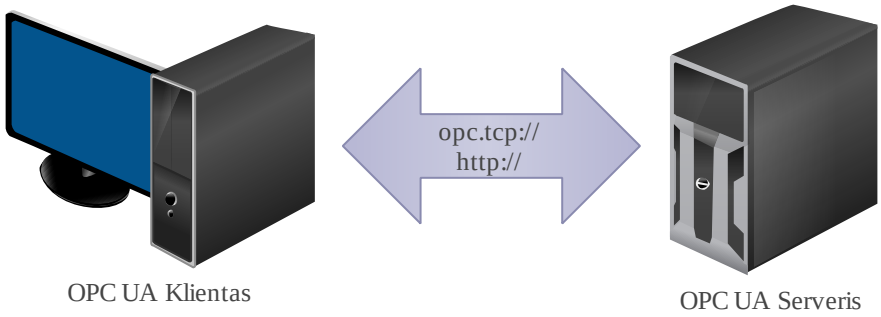
Atviros platformos komunikacijų unifikuota architektūra (OPC UA) yra architektūra skirta naudoti pramonėje. Ji susidedantis iš dviejų pagrindinių OPC UA dalių serverio ir kliento.

OPC UA serveris – techninė ir programinė įranga skirta rinkti duomenims iš klientų, juos saugoti, perduoti kitiems klientams, informuoti apie informacijos pasikeitimus. OPC UA klientas – techninė ir programinė įranga galinti tiek rinkti konkrečią informaciją ir ją atiduoti serveriui, tiek pasiimti iš serverio informacija ir ją panaudoti konkrečiam uždaviniui atlikti. Tiek klientas tiek serveris yra nuo platformos nepriklausomi t.y. kiekvienas iš jų gali būti programuojamas naudojant įvairias programavimo kalbas: Java, C#, C/C++, .NET, JavaScript ir kt., taip pat valdiklių programavimo kalbos kaip: funkcinių blokų diagramos (function block diagram), struktūrizuotas tekstas (structured text) ir kt. [1].

OPC UA palaiko du transporto protokolus *(pav 1.)*:

1. *UA TCP (opc.tcp://Server)* – dvejetainis protokolas.
2. *SOAP/HTTP (http://Server)* – žiniatinklio protokolas.

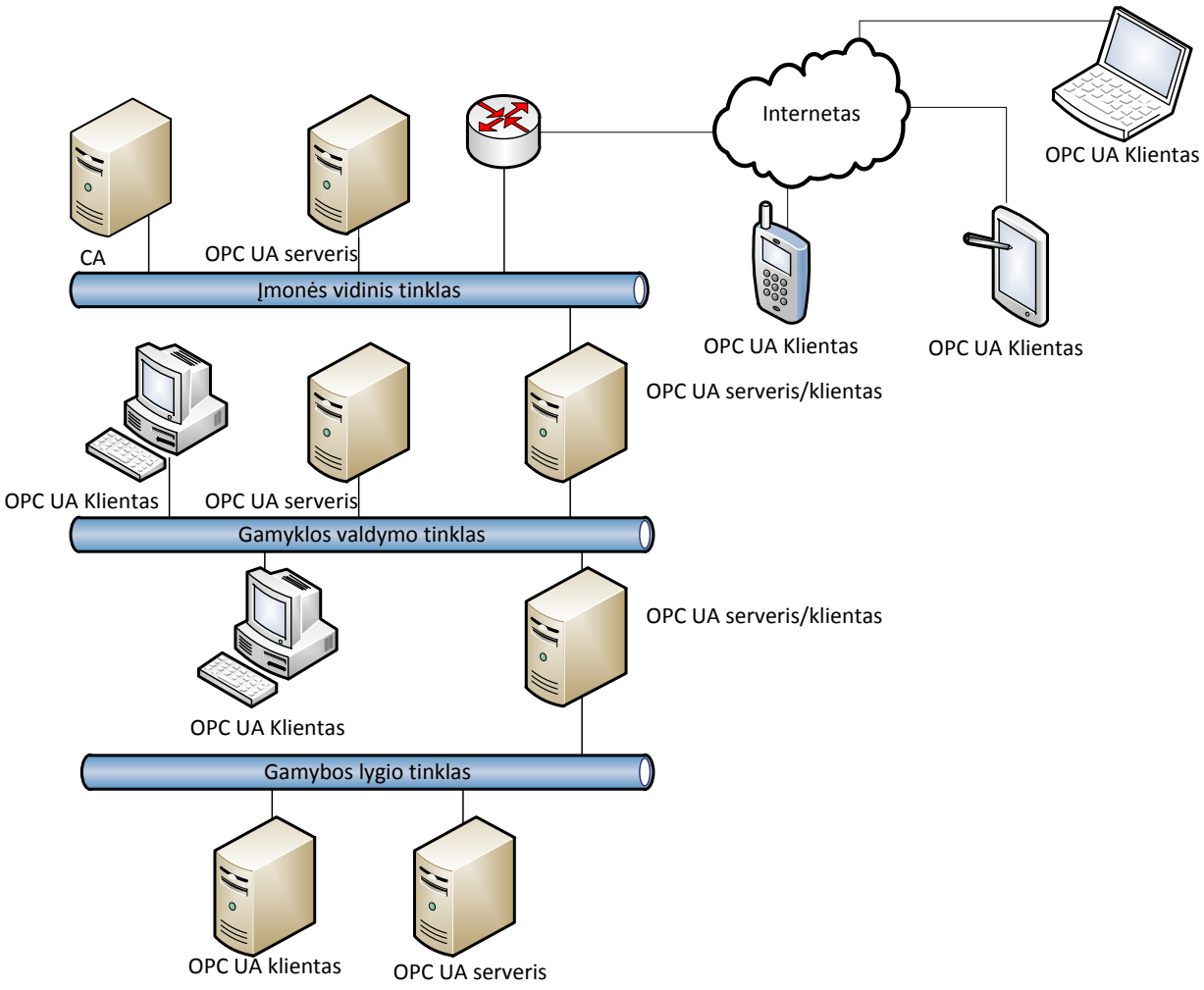
Dvejetainis protokolas siūlo geriausia našumą, naudoja mažai duomenų, siūlo greitą ir lengvą suderinamumą, tarp kliento ir serverio. Perduodant duomenis SOAP/HTTP naudojamos lanksčios ir lengvai pritaikomos XML schemos. Kadangi OPC UA naudoja SOAP/HTTP protokolą, kuris yra plačiai pripažintas standartas, tai suteikia galimybę OPC UA programoms bendrauti tarpusavyje per internetą naudojant standartines žiniatinklio paslaugas (Web Services).



PAV. 1 OPC UA protokolų pavyzdys

OPC UA tinklo pavyzdys yra pateiktas *pav 2.* Pavyzdyje matome, kad OPC UA sudaro skirtingos tinklo dalys, padalintos į sluoksnius. Pirmasis sluoksnis esantis arčiausiai gamybos procesų

yra skirtas perduoti informacijai tarp valdiklių, valdiklių ir jutiklių. Šiame sluoksnyje informacijos perdavimo greitis privalo būti greičiausias, todėl naudojant OPC UA yra naudojamas tik OPC TCP protokolas. Antrasis sluoksnis skirtas visos gamyklos valdymui t.y. duomenims iš valdymo panelių perduoti į valdiklius, kurie atlieką jiems paskirtą užduotį. Iš pirmojo sluoksnio perduodant duomenis į antrąjį ir atvirkščiai yra naudojamas OPC TCP protokolas. Taip pat antrajame sluoksnyje yra saugoma visa informacija apie gamybos esamą būsena. Trečiasis sluoksnis skirtas įmonės vidiniui tinklui. Ji skirtas sutartims su klientais ir tiekėjams vykdyti, perduoti jiems tiesioginę informaciją apie gamybą. Duomenys iš trečiojo sluoksnio į antrąjį ir atvirkščiai gali būti perduodami pasirinktinai tiek OPC TCP protokolu, tiek SOAP/HTTP naudojant XML schemas. Norint perduoti duomenims iš vieno tinklo lygmens į kitą yra naudojamas kompiuteris, kuris vienu metu atlieką ir kliento ir serverio funkcijas. Ši sistema gali būti patraukliu taikiniu tiems, kurie nori pakenkti įmonei ar visuomenei. Būtent todėl OPC UA yra sudaryta iš keleto tinklo lygių ir padidina, apsaugą nuo piktavalių. OPC UA klientai gali veikti bet kur internete ir gali turėti pilną įmonės valdymą. [2]



PAV. 2 OPC UA Tinklo modelio pavyzdys.

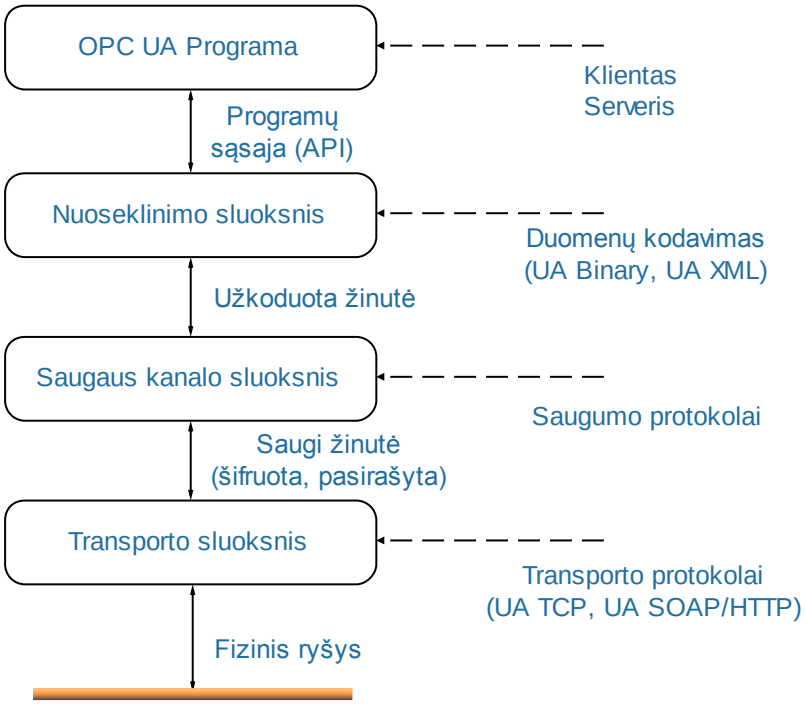
OPC UA OSI modelis:

OPC UA protokolų architektūra yra sudaryta iš pagrindinių trijų sluoksnių, kuris naudoja panašius principus kaip standartinis 7 sluoksnių OSI modelis [3]. Konsultacinė „Digital Bond“ komanda aprašo, kad šis OPC UA modelis, negali būti tiesiogiai lyginamas, o pati specifikacijos terminologija yra paini.

Trys OPC UA protokolų sluoksniai:

- Programos sluoksnis, aprašomas kaip sesijos sluoksnis.
- Komunikacijų sluoksnis, aprašomas kaip saugaus kanalo sluoksnis.
- Transporto sluoksnis, kuris turėtų būti nepainiojamas su ketvirtuoju standartiniu OSI modeliu.

„Digital Bond“ komanda detaliau atvaizduoja OPC UA sluoksnius ir suteikia daugiau informacijos apie jų galimybes, funkcijas. 3 paveiksle pavaizduotas modelis atvaizduoja kaip informacija perduodama iš OPC UA programos, į tinklą. Iš programos duomenys perduodami į nuoseklinimo (Serialization) sluoksnį, kuriame duomenys yra koduojami priklausomai nuo perdavimo būdo pasirinkto programoje, naudojant arba dvejetaini perdavimą arba XML schemas. Koduoti duomenys perduoti į saugaus kanalo sluoksnį (Secure Channel), kur jie yra pasirašomi ir šifruojami, ir perduodami į transporto sluoksnį. Iš jo perduodami duomenys į tinklą [4].



PAV. 3 OPC UA tinklo modelio apžvalga.

2.2.1. Saugumo architektūra

OPC UA saugumo architektūra yra bendrinis sprendimas, kuris padeda įgyvendinti reikalingas apsaugos funkcijas visoje OPC UA sistemoje. Pagrindiniai saugumo OPC UA saugumo tikslai yra šie:

- Konfidencialumas – tai informacijos saugumas, kuris užtikrina, kad perduodant duomenis sistemoje ar internetinėje erdvėje, jie nebus nuskaityti ir/ar pasisavinta trečių šalių.
- Integralumas – tai duomenų saugumas užtikrinantis, kad perduodami duomenys sistemoje internetinėje erdvėje, jie nebus pakeisti kitų asmenų, kad persiunčiamų duomenų turinys išliktų toks pats, koks buvo prieš išsiunčiant.
- Pasiekiamumas – tai duomenų saugumas užtikrinantis, kad duomenys, butu pasiekiami, tada kai jų reikia. Klientui užklausus kokios nors informacijos iš sistemos, ji ją atiduotų.
- Autorizacija – tai duomenų saugumas užtikrinantis, kad tik autorizuoti asmenys, gali prieiti prie jautrių duomenų, ir atlikti leistinus veiksmus, įrašyti, naujus duomenis ar pakeisti senuosius.

Visa informacija: gamyklos bendroji informacija, įvairūs nustatymai, komandos yra perduodamos tarp kliento ir serverio sesijos ir taikymo sluoksniuose. Taikymo (programos) sluoksnis taip pat valdo saugumo uždavinius autentifikuojant ir autorizuojant vartotoją. Taikomojo sluoksnio saugumo uždavinių sprendimas yra atliekamas sesijos servisuose. Taikomajame sluoksnyje sesijai sukurti yra naudojamas saugus kanalas kuris yra sukuriamas komunikacijos sluoksnio ir remiasi jo saugiu ryšiu. Visa sesijos informacija yra perduodama komunikacijos sluoksniui tolimesniam apdorojimui. Norint sesijai komunikuoti saugiu kanalu, jis turi prieš tai būti aktyvuojamas.

Komunikacijos lygmuo suteikia saugumo mechanizmus tam, kad būtų patenkintos konfidencialumo, integralumo ir autentiškumo sąlygos. Tam, kad būtų įvykdytos šios sąlygos yra naudojamas saugus kanalas, tarp kliento ir serverio. Saugus kanalas naudoja šifravimą, tam, kad būtų išlaikomas konfidencialumas, pranešimų pasirašymas, kad būtų patenkinamas integralumas ir skaitmeniniai sertifikatai suteikiantys programoms autentiškumą perduodant duomenis.

OPC UA specifikuoja du alternatyvius steko perdengimus, kurie gali būti naudojami transporto sluoksnyje. Vienas yra vadinamas gimtasis perdengimas, kurio saugumo funkcionalumas konfidencialumui, integralumui, autentiškumui ir saugiam kanalui yra panašūs į SSL/TLS specifikaciją. Antrasis tinklo serviso perdengimas. Jame yra naudojama interneto paslaugų saugumas (WS Security), interneto paslaugų saugus dalinimasis (WS SecureConversion) ir XML failų šifravimas ir pasirašymas, tam, kad tenkintų būtinas sąlygas konfidencialumui, integralumui, taikomųjų programų autentiškumui ir saugiam kanalui sudaryti.

Transporto sluoksnis yra atsakingas už prarastų susijungimų atstatymą, ir bet kokių nedidelių neatitikimų pašalinimą, tam kad būtų išlaikytas nenutrūkstamas saugus loginis kanalas.

Saugumo politikos:

Saugumo politikos specifikuoja, kurie saugumo mechanizmai yra gaunami ir naudojami iš saugumo profilių. Serveriai naudoja saugumo politikas tam, kad paskelbtų kokius saugumo mechanizmus jis palaiko ir kad klientas galėtų pasirinkti vieną iš šių politikų, tam, kad būtų sukurtas saugus kanalas duomenims perduoti. Šios politikos apima:

- Algoritmus pasirašymui ir šifravimui
- Algoritmus raktų perdavimui ir išgavimui

Politikas, kurios bus naudojamos serveryje ar klientuose, yra programuojamos OPC UA aplikacijų programavimo metu. Jei serveris aprūpina keletą klientų, jis palaiko keletą atskirų politikų skirtingiems klientams. Tai leidžia klientams pasirinkti kokią nori saugumo politiką nepriklausomai nuo kitų klientų, kurie naudoja tą patį serverį saugiems kanalams sukurti.

Metams vis bėgant kompiuterių galimybėms didėja, jei saugumo algoritmai yra skaitomi saugūs šiandien, ateityje jie gali tapti visiškai nesaugūs, todėl tai turi prasmę, kad politikos būtų atskiros, jei viena iš jų tampa nesaugi, jos galima atsisakyti. OPC UA suteikia galimybę įdiegti naujas politikas su naujais algoritmais, kurie padidina saugumo lygį sistemoje. [5]

OPC UA naudoja keturias saugumo politikas:

1. Pirmoji iš jų neturi jokio saugumo vadinama “None“.
2. Antroji yra „Basic128RSA15“. Simetriniui parašui yra naudojamas HmacSha1 algoritmas, i RSASha1 asimetriniui parašui. AES-128-CBC simetrinio šifravimo algoritmas, ir RSA15 asimetrinio šifravimo algoritmas. Ji naudoja asimetrinį raktų įvyniojimo algoritmą: KwRsa15 , ir Psha1 raktų išvedimo algoritmą.
3. Trečioji yra „Basic256, naudoja HmacSha1 simetriniam parašui ir RsaSha1 asimetriniui parašui. Palyginti su antrąja tai ši naudoja saugesnį AES-256-CBC simetrinio šifravimo algoritmą. RsaOaep algoritmas naudojamas kaip Asimetrinis šifravimas ir KwRsaOaep raktų įvyniojimo algoritmą. “Basic128Rsa15“ ir „Basic256“ naudoja tą patį sertifikatų pasirašymo algoritmą Sha1. Perduodamo parašo ilgis yra 128 bitų. Atitinkamai minimalus ir maksimalus asimetrinio parašo ilgis yra 1024 ir 2048 bitų.
4. Ketvirtoji politika yra “Basic256Sha256” . Pati saugiausia politika, kuri naudoja Hmac_Sha256 algoritmą simetriniam parašui ir AES-256-CBC algoritmą simetriniui šifravimui. Rsa_Sha256 algoritmas naudojamas asimetriniui parašui. Naudojami tie patys raktų įvyniojimo ir šifravimo algoritmai kaip ir „Basic256“ bet naudoja du kartus ilgesnius asimetrinius raktus. Perduodamo parašo rakto ilgis yra 256-bitai. Asimetrinio rakto ilgis yra 2048 ir 4096. Yra naudojamas Sha256 sertifikato parašo algoritmas.

5. Penktoji politika „BasicECD“ naudoja elipsines kreives. Curve P-256 algoritmas naudojamas simetriniam ir asimetriniam šifravimui asimetriniui parašui raktų įvyniojimui ir išgavimui. HmacSha1 algoritmas naudojamas simetriniam parašams.

Visos politikos gali įgyvendintos generuojant raktus ir parašus su „Microsoft“ produktu „Windows Crypto Library“ arba naudojant OpenSSL.

Sertifikatų kūrimas

OPC UA yra labai didelė sistema, todėl yra naudojama sertifikatų saugykla, kur yra saugomi sertifikatai ir privatūs raktai. Nors OPC UA palaiko dvi saugyklas, tačiau praktikoje yra paplitusi tik OpenSSL. Dažniausiai yra vienas serveris, kuris prižiūri visus sertifikatus ir privačius raktus. Pagal užklausas, kuria naujas privačių ir viešų raktų poras naudojant OpenSSL biblioteką. Ši biblioteka yra naudojama įvairiuose operacinėse sistemose ir suteikia galimybę, naudoti daug įvairių kriptografinių funkcijų, o naudojant komandinę eilutę, galima generuoti ir tvarkyti sertifikatus. Ši biblioteka reikalauja bent minimalių žinių apie kriptografinius algoritmus, ir žinių apie OPC UA saugumą, bet kaip OPC UA naudoja OpenSSL [6].

Norit sugeneruoti privatų raktą su savo pasirytą, galima naudoti funkcijas *oppenssl genrsa 1024 > myapplication.key.* Viešojo rakto išgavimui naudojama funkcija *openssl req –new –key myapplication.key –out myapplication.crt.* Tam, kad šis viešasis raktas būtų pasirašytas galima naudoti funkcija *openssl req –x509 – key myapplication.key – in myapplication.csr – out myapplication.crt –days 365.* Paskutinis simbolis nurodo, kiek laiko galios sertifikatas. Šiuo atvėju jis nurodo, kad šis sertifikatas galios vienerius metus.

Norint sugeneruoti private raktą ir sertifikatą pasirašytą su SC galima naudoti funkciją *oppenssl genrsa 1024 > myapplication.key privataus rakto generavimui* ir *openssl req –new –key myapplication.key –out myapplication.crt viešojo* rakto išgavimui. Tam, kad pasirašyti sertifikatą SC naudojama *openssl x509 -req -days 365 -in myapplication.csr -CA cacert.crt -CAkey private/cakey.key –CA createserial -out myapplication.crt.* Naudojant sertifikavimo centrą sertifikatas gali būti atšauktas naudojant komandą *openssl ca -config X509CA/openssl.cnf -revoke X509CA/certs/ myapplication.pem.*

OPC UA klientų ir serverių produktai yra sertifikuojami pagal jų profilius. Kai kurie profiliai specifikuoja saugumo funkcijas, o kiti profiliai gali aprašyti kitas funkcijas, kurios nėra susijusios su saugumu. Profiliai nustato reikalavimus sertifikavimo produktui, bet nenusako, kam šie produktai yra naudojami. Minimalus saugumo lygis yra reikalaujamas, bet kokiam saugumo profiliui. Skirtingiems saugumo profiliams gali būti taikomi skirtingi saugumo detalės, pavyzdžiui, kokie šifravimo algoritmai yra naudojami, ar yra būtini norint atlikti kokią nors OPC UA funkciją. Jei kokia nors saugumo problema buvo rasta viename algoritme OPC UA gali nesunkiai pašalinti šį saugumo

profilį ir naudoti alternatyvius algoritmus, kuriuose dar nėra žinoma ar nėra užfiksuota saugumo spraga.

Kiekvienas OPC UA saugumo mechanizmas yra numatytas kliento ir serverio produktuose. Pagal saugumo profilius, kiekvienas klientas ir serveris jų turi laikytis. Šie saugumo mechanizmai ir profiliai yra naudojamas pagal poreikį, ir nėra būtini.

OPC UA saugumo sąlygos susideda iš keturių pagrindinių dalių: taikomosios programos egzemplioriaus, taikomosios programos administratoriaus, taikomosios programos operatoriaus ir sertifikavimo centro. Visi OPC UA programos įrašytos į įvairias įrenginius (serverius, valdiklius, kompiuterius, išmaniuosius įrenginius) yra vadinami taikomosios programos egzemplioriais. Kiekvienas programos egzempliorius privalo turi vieną administratorių, kuris tvarko programos egzemplioriaus saugumą. Kiekvienas programos egzempliorius gali būti naudojamas vieno ar daugiau operatorių, iš kurių kiekvienas gali tūrėti vartotojo kredencialus, pagal kuriuos gali būti apibrėžiama priėjimo taisyklės ir/ar sekti vartotojo aktyvumą. Vartotojo kredencialai yra bendrasis terminas apibrėžiant elektroninį tapatybės atpažinimą. Jie gali būti perduodami serveriui, po to kai yra nusiunčiamas sertifikatas, tam, kad būtų galima sudaryti saugų kanalą. Sertifikavimo centras yra administratorius arba organizacija, kuri yra atsakinga už sertifikatų kūrimą. Sertifikavimo centras privalo patikrinti visą sertifikate esančią informaciją ir užtikrinti, kad ši informacija yra teisinga. Taip pat pridėti skaitmeninį parašą, kuris padeda identifikuoti, kad informacija nebuvo pakeista. Kiekvienas sertifikavimo centras turi turėti savo sertifikatą, kuris yra naudojamas kuriant skaitmeninius parašus. Sertifikatas yra elektroninė tapatybės kortelė, kuri gali būti įtraukta į taikomąją programą. Jame yra saugoma informacija apie sertifikato savininką, sertifikato išdavėją , taip pat unikalų slaptą raktą skirtą kurti ir tikrinti skaitmeninius parašus. Šių sertifikatų sintaksė yra atitinka x.509 specifikaciją. Tokie sertifikatai vadinami „x.509 Sertifikatais“. Su šiais sertifikatais yra susietas privatus raktas, kuris yra žinomas tik sertifikato savininkui. Šis privatus raktas leidžia kurti skaitmeninius parašus ir šifruoti informacijai. Jei šis raktas yra atskleidžiamas neautorizuotiems asmenims tada sertifikatas susietas su raktu negali būti daugiau patikimas. Sertifikatų saugykla skirta saugoti sertifikatams ir privatiems raktams saugoti failų sistemoje. Visos „Windows“ operacinės sistemos leidžia saugoti sertifikatus naudojant registrais paremtu “Windows“ sertifikatų saugykla. Visos sistemos palaiko aplankais suskirstytą sertifikatų saugyklą, kuri vadinama OpenSSL sertifikatų saugykla. Pasitikėjimo sąrašas yra sąrašas skirtas saugoti sertifikatams, kurie yra patikimi programos egzemplioriaus. Kai OPC UA saugumas yra naudojamas visos OPC UA programos privalo atmesti visus prisijungimus, jei jie neturi sertifikato iš patikimo sąrašo, ar jų sertifikatas nėra išduotas patikimo sertifikavimo centro, kuris turi jį savo pasitikėjimo sąraše. Atšaukimo sąraše yra saugoma kurie sertifikatai buvo atšaukti sertifikatų centro ir privalo būti nepriimti programos egzemplioriaus.

Kiekvienas programos egzempliorius turi savo sertifikatą sukurta sertifikato centro ir patikimų sertifikatų sąrašą. Administratorius gali atnaujinti sertifikatą, privatų raktą , ar kitą būtiną informaciją. Kiekvienas programos egzempliorius kiekvieną kartą patikriną ar jo sertifikatas yra galiojantis, naudodamasis atšauktų sertifikatų sąrašu. Visi operatoriai turi savo vartotojo kredencialus, ir naudoja juos kad prisijungti prie tam tikros programos ir padaryti reikalingus nustatymus procese.

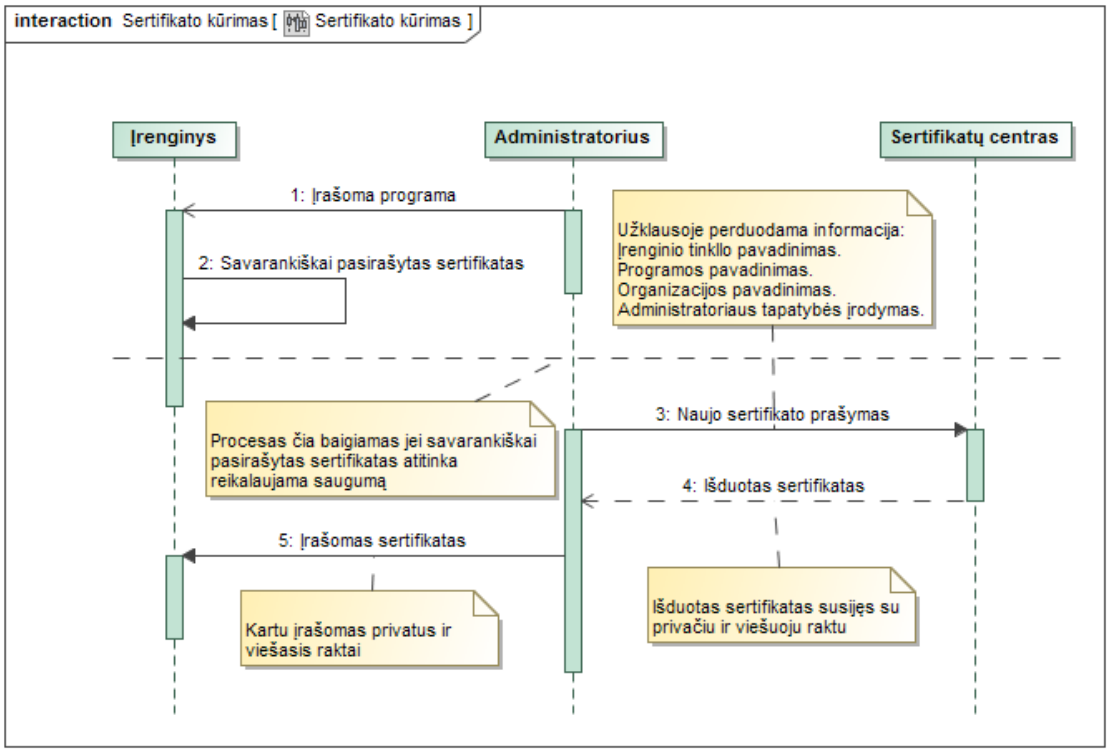
Visi sertifikatai yra kartu su privačiais raktais, kurie leidžia programų egzemplioriams sukurti saugų komunikacijų kanalą, kuris negali būti peržiūrėtas, nuskenuotas ar pakeistas, jokių trečiųjų šalių, kol buvo perduodami duomenys. OPC UA programos gali būti suskirstytos į keturias dalis. Pirmoji pavadinta „Be autentifikavimo“. Ji leidžia klientui ir serveriui tarpusavyje komunikuoti, o visi galiojantys sertifikatai yra patikimi. Programos sertifikatai yra naudojami, kad suteiktų informaciją apie sertifikato savininką, tačiau ši informacija nėra patikima, todėl neįmanoma sužinoti ar sertifikatas yra to savininko. Šioje konfigūracijoje tiek kliente, tiek serveryje įdiegtos programos automatiškai priima visus galiojančius sertifikatus, ir prideda juos į savo patikimų sertifikatų sąrašą, jei jie dar nebuvo pridėti. Šis metodas negali užtikrinti jokios privačios informacijos saugumo perduodant duomenis, įskaitant vartotojo kredencialus. Šis metodas gali būti tinkamas sistemoje tik, tada jei yra naudojamas koks nors kitas sauga užtikrinantis mechanizmas, toks kaip fiziškai saugi ir izoliuota sistema, kuri neturi jokio prisijungimo prie išorinio tinklo, arba naudojant kitokį saugumo mechanizmą: VPN, transporto sluoksnio saugumą ir t.t. Šis metodas tinkamas naudoti jei informacija, serveryje yra vieša ir visiems prieinama, o priėjimas prie jos yra atviras. Programuotojui reikia tik sukonfigūruoti programos egzemplioriaus įrašymo procedūrą įrenginyje. Antrasis vadinamas „Serverio autentifikavimas“. Šitame metode serveris leidžia prisijungti visiems klientams, o jei prie yra reikalaujama vartotojo kredencialai, tokie kaip prisijungimo vardas ir slaptažodis tai jie yra siunčiami tik tada kai yra sukurtas saugus kanalas. Tačiau klientai turi būti sukonfigūruoti administratoriaus. Klientas pasitiki serveriu, jei konfigūravimo metu administratorius įdeda serverio sertifikatą į kliento patikimų sertifikatų sąrašą, arba serverio sertifikatas yra išduotas sertifikavimo centro ir jis yra sertifikavimo centro patikimų sertifikatų sąraše. Jei serverio sertifikatas dar nėra jokiam patikimam sąraše klientas privalo palyginti DNS pavadinimą serverio sertifikate su DNS pavadinimu, prie kurio serverio jungiasi. Jei jie sutampa tai klientas žino, kad jis jungiasi būtent prie to serverio prie kurio ir turi. Jei jie nesutampa tai nėra jokių garantijų, kad klientas prisijungė prie norimo serverio. Šis metodas yra naudojamas daugumoje interneto bankų programų kur serveris turi sertifikatą išduotą sertifikavimo centro tokius kaip „VeriSign“ kurie yra automatiškai patalpinami naršyklėse. Tai suteikia pakankamai gerą saugumą, tačiau serveris negali apriboti kliento programos. Trečioji dali yra skirta kliento autentifikavimui. Šiame metode klientų programos gali prisijungti prie bet kokio serverio, tačiau serveris leidžia prisijungti tik tiem, kurių sertifikatai yra serverio patikimų

sertifikatų sąraše. Klientinė programa niekada neatskleidžia slaptos informacijos, kol nėra žinoma ar serveris yra patikimas. Jokios pradinės konfigūracijos klientinėje pusėje daryti nereikia, neskaitant tik serverio adreso prie kurio jungiamasi. Serverio programoje, administratorius privalo sudėti klientų sertifikatus į patikimų sertifikatų sąrašą jei sertifikatai nėra išduoti sertifikavimo centro. Šis metodas naudojamas paieškų tarnybos, kuri patikrina, ar atitinkama kliento programa turi autorizuotą priėjimą prie atitinkamo serverio. Šiame metode naudojamas vietinis paieškos serveris, kuris leidžia prisijungti ir priregistruoti tik autorizuotoms programoms. Ketvirtoji dalis vadinama „Abipusio autentifikavimo“. Kiekvienas ir serveris, ir klientas priima tik patikimas programas prisijungti. Šis metodas turi patį didžiausią saugumo lygį, bet reikalauja, kad kiekvienas ir klientas, ir serveris būtų sukonfigūruoti administratoriaus. OPC UA rekomenduoja naudoti tik šį metodą, jei sistema yra naudojama viešoje erdvėje, kur saugumas yra pagrindinis rūpestis. Kaip ir antroje dalyje klientai turi patikrinti DNS pavadinimus, tam kad būtų sertifikatas patalpinamas į patikimų sertifikatų sąrašą. Šis metodas naudojamas, aplinkose, kur administratorius nori pilnos kontrolės, tarp serverio ir kliento.

Įrašant naujas programas į OPC UA sistemą saugumas turi būti parinktas nedelsiant ir jei administratorius atsakingas už programą nusprendžia, kad savarankiškai pasirašytas sertifikatas neatitinka reikalaujamo organizacijos saugumo, tai tada administratorius taip pat turi įrašyti sertifikatą išduotą sertifikavimo centro.

Kiekviena OPC UA programa turi leisti administratoriui pakeisti programos sertifikatą į tokį sertifikatą, kurio reikalauja esamas saugumas. Kai administratorius užklausia naujo sertifikato iš sertifikavimo centro, sertifikavimo centras gali reikalauti iš administratoriaus pateikti įrodymą, kad ji gali prašyti naujų sertifikatų organizacijai, kurioje bus jis įdiegtas. Tikslus įrodymų surinkimo mechanizmas priklauso nuo sertifikavimo centro.

Įmonė gali automatizuoti sertifikatų gavimo procesą iš sertifikavimo centro. Nors administratoriui reikės atlikti tokius pačius veiksmus, tačiau jam nereikės sertifikato perkėlinėti į programas, tai bus atliekama automatiškai, ir administratoriui nereikia keliauti po įrenginius ir juos įrašinėti. [6]



PAV. 4 Programos egzemplioriaus įrašymo sekos diagrama

2.3. Saugumo grėsmės

2.3.1. Užliejimas žinutėmis

Puolėjas gali siųsti didelį kiekį žinučių, arba vieną žinutę, kuri turi didelį kiekį užklausų su tikslu užlieti serverį, kad serveris ar jo komponentai: procesorius, TCP/IP stekas, operacinė sistema, failų sistema, sutriktų ir negalėtų atlikti reikalingų operacijų. Pranešimų užliejimo atakos gali būti vykdomos įvairiuose sluoksniuose, naudojant įvairius protokolus: SOAP, HTTP, TCP.

Žinučių užliejimo atakos gali būti dviejų tipų, kai žinutės yra gerai suformuotos ir blogai suformuotos. Pirmuoju atveju puolėjas gali būti piktas asmuo naudojantis teisėtu klienu. Du galimi atvejai, kai klientas turi sesiją su serveriu ir kai neturi. Žinučių užliejimas gali sutrukdyti naujoms sesijoms užsimegzti, arba nutraukti jau esamas. Kitu atveju puolėjas gali pasinaudoti kenkėjišką klientu, kuris užpila serverį su netinkamai suformuotomis žinutėmis tam, kad būtų išnaudoti visi serverio resursai. [7]

Žinučių užliejimas gali pabloginti susisiekimą su OPC UA subjektais ar visai atsisakyti juos aptarnauti. Užliejimas žinutėmis paveikia pasiekiamumą. OPC UA sumažina užliejimą žinutėmis poveikį prieinamumui, sumažinant kiekį dirbant su žinutėmis prieš tai kol yra autentifikuojama. Tai užkerta kelia puolėjui nuo OPC UA taikomoji programa prarastu daug laiko, kol gaus atsakymą, tačiau tai atima dalį resursų iš teisėtos veiklos [8].

GetEndpoints ir OpenSecureChannel vieninteliai servisai, kuriuos serveris apdoroja prieš kliento atpažinimą. Atsakymas į GetEndpoints yra tik dalis statinės informacijos, todėl serveriui nereikia atlikinėti daug veiksmų. Tačiau atsakymas į OpenSecureChannel sunaudoja ženklų serverio

resursų kiekį todėl, kad yra atliekami parašų ir šifravimo veiksmai. Nors OPC UA ir sumažino veiksmų kiekį atliekant šiuos veiksmus, tačiau jie negali būti pašalinti. Serveris gali apsaugoti save nuo žinučių užtvindymo naudojant OpenSecureChannel servisą dviem būdais:

1. Serveris gali tyčia atidėti OpenSecureChannel užklausas, kai gaunama bloga užklausa ir pranešdamas pavoju, kad yra galimas užpuolikas aptarnaujančiam personalui.
2. Kai OpenSecureChannel užklausos bando viršyti serverio nustatytą maksimalų skaičių kanalų serveris atsako su klaida, ir neatlieka jokių veiksmų su pasirašymų ir šifravimu. Serveriuose turi būti nustatomas galimas maksimalus kanalų skaičius.

2.3.2. Slapto pasiklausymo

Slaptas pasiklausymas - elektroninė ataka, kurios metu perduodami duomenys yra perimti asmenų kuriems šie duomenys nebuvo skirti. Slaptas pasiklausymas atliekamas perimant arba šnipinėjant komunikacijas tarp dviejų taškų.

Skaitmeniniame pasaulyje slaptas pasiklausymas yra atliekamas šnipinėjant perduodamus paketus ir yra vadinamas „Tinklo slaptu pasiklausymas". Yra naudojamos specializuotos programos tam, kad šnipinėtų ir įrašytų tinklu perduodamų duomenų paketus. Surinkus pakankamą paketų kiekį, yra naudojami kriptografiniai įrankiai paketų analizei ir iššifravimui. Pavyzdžiui naudojant IP telefonija (Voice over IP(VoiP)) pokalbiams yra naudojamas interneto protokolas, kadangi visi paketai yra perduodami internetu jie gali būti paimami įrašomi naudojant protokolo analizatorius, o vėliau naudojant specialią įrangą konvertuoti į garso įrašus, kurių galima pasiklausyti. [9]

Duomenų rinkimas yra labai paprastas vietiniuose tinkluose, kuriuose yra naudojami tinklo šakotuvai, nes visi pranešimai yra siunčiami į visus prievadus, todėl asmeniui šnipinėjant paketus tereikia tik priimti visus perduodamus duomenis. Toks pat principas yra naudojamas bevieliuose tinkluose, kur paketai yra perduodami laisvai ir šiuos paketus gali priimti ne tik tikrasis gavėjęs, bet ir kiti asmenys, jei tik jie turi tinkamas priemones.

Slapto pasiklausymo metu yra informacija, kurią panaudojus su tam tikromis priemonėmis galima atskleisti slaptą informacija. Tai gali privesti prie tiesioginio saugumo pažeidžiamumo. Surinkta informacija, gali būti parduota, norint pasipelnyti, ar panaudojama valesnėms atakoms. Slaptas pasiklausimas tiesiogiai veikia konfidencialumą, ir netiesiogiai gali paveikti kitus saugumo mechanizmus.

OPC UA saugumo architektūra naudoja duomenų šifravimą, prieš slaptą pasiklausymą. [8]

2.3.3. Žinučių klastojimas

Dauguma protokolų naudojančių TCP/IP paketus, neturi jokių autentifikacijos mechanizmų ir nėra žinomas, kas yra žinutės gavėjas, ar žinutės siuntėjas. Dėl to visos žinutės perduodamos tinkle gali būti pažeidžiamos jas klastojant. Kad to išvengti, gali būti naudojamas papildomos atsargumo priemonės aplikacijos lygmenyje, kurios tikrina tapatumą tiek siuntėjo, tiek gavėjo. Pavyzdžiui interneto protokolo adreso (IP) klastojimas ir adresų perkodavimo protokolas (ARP) gali būti naudojanti, kad būtų įvykdoma „Žmogus – viduryje“ atakoms, prieš pasirinkta kompiuterinį tinklą. Žinučių klastojimas, naudojantis TCP/IP protokolus, gali būti sušvelnintas naudojant ugniasienes, kurios tikrina perduodamus paketus ar naudojant matavimo priemones, tam kad patikrinti žinutės siuntėjo tapatybę. Puolėjas gali klastoti žinutes tiek iš kliento pusės, tiek iš serverio, ir atlikti neautorizuotas operacijas.

Žinučių klastojimas veikia vientisumą, nes jų turinys gali būti pakeistas, taip pat autorizacija, jei jos buvo siunčiamos, ne tikrojo siuntėjo. OPC UA saugumo politika siūlo naudoti kiekvienos žinutės pasirašymą. Pasirašant žinutes yra žinoma kas jas išsiuntė. [8]

2.3.4. Žinučių pakeitimas

Žinučių klastojimas atliekamas tada, kai yra neautorizuotai modifikuojamas kodas ar duomenys, pažeidžiant vientisumą. Tinkle, duomenys srauto žinutės gali būti sugautos, modifikuotos ir išsiųstos į klientus arba serverius. Žinučių pakeitimas gali leisti neteisėtą prieigą prie sistemos, pakeisti duomenis, taip, kad jie sutrigdytų visą gamybą. Žinučių pakeitimas gali paveikti vientisumą ir autorizaciją.

OPC UA atremia žinučių pakeitimą, pasirašant žinutes naudojant SHA-2 šeimos algoritmus. Gavus žinute yra tikrinamas parašas ir perduoda žinutė. Jei patikrinus hash reikšmės neatitinka - žinutė buvo pakeista, ir leis gavėjui žinutę ignoruoti. [10]

2.3.5. Žinučių kartojimas

Žinučių kartojimas yra tinklo ataka, kai galiojantys duomenys yra perduodami pavėluotai arba pakartotinai. Piktavaliai gali pagauti galiojančių programų žinutės ir išsiųsti jas vėliau be jokių pakeitimų, taip puolėjas gali dezinformuoti vartotoją/ar sistemą, kuris gali netinkamu laiku atlikti netinkamą komandą, pavyzdžiui atidaryti vožtuvą, netinkamu metu. Žinučių klastojimas paveikia autorizaciją.

OPC UA naudoja sesijos numerį, saugos kanalo numerį, laiko žymas, sekos numerius, užklausos numerį kiekvienai užklausos ir atsako žinutėms. Žinutės yra pasirašytos ir negali būti pakeistos be pastebėjimo, todėl bus labai sunku pakartoti žinutę kuri turėtų galiojantį sesijos numerį, saugaus kanalo numerį, tinkamą laiko žymę, tinkamą sekos numerį ir užklausos numerį. [8]

2.3.6. Netinkamas žinučių formavimas

Puolėjas gali sukurti daug įvairių žinučių su negaliojančia žinutės struktūra (neteisingai suformuotas XML, SOAP, UA dvejetainis) ar duomenų reikšmės ir nusiųstos klientams ar serveriams.

OPC UA klientai ar serveriai gali neteisingai elgtis pagal neteisingai suformuotas žinutes. Gali atlikti neautorizuotą operaciją ar apdirbti nereikalingą informaciją. Tai gali privesti prie servisų pablogėjimo, programos užstrigimo, ar „nulūžimo“ arba įterptųjų prietaisų visišką programos ar prietaiso išjungimą. Blogai suformuotos žinutės paveikia integralumą ir pasiekiamumą.

OPC UA klientai ir serveriai tikrina žinutės formą ir joje esančius parametrus ar jie yra tinkamame diapazone. Netinkančios žinutės su netinkamais duomenimis nėra nustatomos ar vykdomos. [11]

2.3.7. Serverio profiliavimas

Puolėjas gali bandyti išgauti informacija apie tapatybę, tipą, programos versiją ir serverio ar kliento kūrėjus tam, kad sužinotų apie sistemos specifinius pažeidimus. Puolėjas gali bandyti sukurti reikiamą žinutės formą siųsdamas galiojančias ar negaliojančias žinutes į taikinį.

OPC UA apriboja informacijos kiekį, kurį serveris suteikia klientams, kurie nėra identifikuoti. Visa ši informacija yra atsakas į GetEndpoints užklausas. [12]

2.3.8. Sesijos užgrobimo

Puolėjas gali naudoti informacija (gautą šnipinėjant komunikaciją arba spėjant) apie einamąją sesiją, esančia tarp dviejų programų tam, kad įterptų suklastotą žinutę (su galiojančia sesijos informacija). Tai leidžia jam perimti sesiją iš autorizuoto vartotojo. Puolėjas gali gauti neautorizuota priėmimą prie duomenų arba atlikti neautorizuotas operacijas. Sesijos perėmimas daro poveikį visiems saugumo objektams.

OPC UA naudoja saugumo kontekstą (saugų kanalą) kuriant kiekvieną sesiją, kad apsisaugotų nuo sesijos užgrobimo. Norint įvykdyti sesijos užgrobimą pirmiausia tektų sukompromituoti saugumo kontekstą.

2.3.9. Kenkėjiškas serveris

Puolėjui sukūrus kenkėjišką serverį ar įdiegiant neautorizuotą programą į autentišką serverį, OPC UA klientai gali atskleisti reikalingą informaciją.

OPC UA klientų programinės įrangos, naudoja serverio programinės įrangos egzempliorių sertifikatus, kad jis būtų patvirtintas. Nors ir lieka galimybė, kad kenkėjiškas serveris gali gauti sertifikatą iš autentiško serverio ir ji naudoti, tačiau jis neturi tinkamo privataus rakto žinučių iššifravimui ir patvirtinimui, kurios buvo apsaugotos su viešuoju raktu, kenkėjiškas serveris niekada negalės perskaityti paslėptos informacijos kurią atsiuntė kliento programinė įranga.

2.3.10. Sukompromituoti vartotojo tapatybės duomenys

Puolėjas gauna kokius nors tapatybės duomenis, tokius kaip vartotojo vardas, slaptažodis, sertifikatai ar raktai, ieškant jų popieriuose, ekranuose ar kituose elektroninėse komunikacijose, nulaužiant juos spėjant, ar naudojant automatinius įrankius slaptažodžiams nulaužti.

Neautorizuotas vartotojas gali prieiti prie sistemos tam, kad gautų visą informaciją. Gaudamas duomenis jis, galėtų tuo pasinaudoti ir padaryti žalą gamyklos operacijoms ar informacijai. Naudojant sukompromituotus tapatybės duomenis visi veiksmai yra atliekami teisėtai.

OPC UA apsaugo vartotojų tapatybes duomenis siunčiant juos tinklu tik šifruojant. OPC UA pasitiki svetainių kibernetine saugumo valdymo sistema (Cyber Security Management System), kuri apsaugo nuo kitų atakų galinčių išgauti vartotojo tapatybės duomenis, bandant atspėti slaptažodžius ar naudojant socialinę inžineriją.

2.4. Fizinis saugumo užtikrinimas

Įrangos dubliavimas yra pagrindinis dalykas tam, kad būtų padidintas patikimumas ir pasiekiamumas. Jei vienas iš komponentų pranešė apie klaidą, tai visą jo darbą perima kitas. OPC UA dubliavimas yra pagristas serverių ir klientų programų dubliavimų ir gali būti lengvai įgyvendintas laikantis tam tikros duomenų ir servisų struktūros.

Klientų dubliavimas yra reikalingas aplinkose kuriose, pavyzdžiui jei yra reikalingas pastovus gamybinio proceso stebėjimas. Kitas pavyzdys yra kai serveris suveda duomenis iš pagrindinių serverių atlikti specialiems skaičiavimams. OPC UA palaiko dubliavimą naudojant TransferSubscriptions servisą, kombinuojant kartu su kliento informacija esančia serverio Adress Space lauke, kur serveris turi aktyvią kliento duomenų prenumeratą ir kliento atsarginę kopiją. Ši atsarginė kliento kopija stebi aktyvaus kliento sesijos informacija adressspace lauke. Kai tik aktyvus klientas tampa nepasiekiamas ir sesijos būsena pasikeičia adressspace laukelyje, atsarginis klientas perima visus TransferSubscriptions servisus tam, kad gautų visas reikiamas prenumeratas iš aktyvaus kliento. Prenumeratos gali išlikti pereinant nuo vienos sesijos į kitą dėl to, kad jos gyvavimo laikas yra kur kas ilgesnis nei sesijos. Serveris privalo turėti buferį, kuris saugo duomenis, kurie buvo išsiųsti klientui kliento nulūžimo metu, kad nebūtų prarasti duomenys. Šis mechanizmas reikalauja kliento atsarginės kopijos, kurioje yra informacija apie sesijos numerį ir prenumeratos numerį, tam kad būtų galima perimti prenumeratą iš aktyvaus kliento. OPC UA neturi standartinės procedūros kaip klientai turi keistis šia informacija. [13]

Serverių dubliavimas gali būti skaidomas į matomo serverio dubliavimą ir nematomo serverio dubliavimą. Pirmuoju atveju serverio dubliavimas tvarkomas klientui nežinant. O tai reiškia, kad klaidos atveju klientas net nepastebės, kad buvo įvykusi klaida ir jam nereikia nieko daryti, kad būtų atliktos užduotys. Serveris yra atsakingas , kad klientas gautų visą reikiamą informaciją. Norint tai įgyvendinti reikia, kad dubliuotas serveris stebėtų aktyvų serverį. Jie abu turi turėti visiškai

vienodus duomenis ir informaciją apie sesijas, ir jei aktyvus serveris nebėra pasiekiamas visos kitos užklausos yra perduodamos į dubliuotą atsarginį serverį.

2.5. OPC UA pažeidžiamumai

Dale G Peterson iš „Digital Bond“ komandos straipsnyje „OPC UA: Part 3 – Specification Vulnerabilities“ [4] aprašė, kad daugybėje OPC UA specifikacijos dalių saugumo funkcijos suteikia aukšto lygio saugumą, tačiau didžiausia problema yra OPC UA vartotojas, kuris gali nesunkiai pajausti klaidinantį saugumo jausmą, net su lengvai valdomu produktu, kuriame yra įjungti/įdiegti saugumo nustatymai.

OPC UA specifikacijos aprašo, kad tiek serverių tiek klientų programos, jas pirmą kartą paleidus, jos sukuria sertifikatus ir juos pasirašo su savo privačiuoju raktu. Kadangi bet koks piktavalis gali susikurti tokį sertifikatą ir jį pateikti, tai nereiškia, kad jis yra tas asmuo kuo jis teigia esąs. OPC UA specifikacija leidžia priimti tokius sertifikatus be jokio automatinio ar rankinio patikrinimo ar sertifikatas gali būti patikimas. Yra tikėtina, kad daugelis naudojančių šią sistemą tiesiog nesidiegs viešojo rakto infrastruktūros.

Dėl šios priežasties bet kas, kuris jungiasi į serverį gali gauti prisijungimą. O atakos bus perduodamos per saugų kanalą, naudojant pasirašymą ir šifravimą.

2.6. VPN

VPN (angl. k. Virtual Private Network) tai – virtualus privatus tinklas suteikiantis galimybę išplėsti vietinį tinklą, per viešąjį tinklą/internetą, suteikiantis galimybę kompiuteriams ar tinklo įrenginiams siųsti ir gauti informaciją iš viešojo tinklo, taip lyg būtum prisijungęs tiesiogiai prie vidinio tinklo. Virtualus privatus tinklas kuria taško į tašką (point – to – point) susijungimus naudojantis esamu prisijungimu taip sukurdamas virtualų tunelį, tarp taškų.

VPN OSI modelis

VPN tinklas gali būti kuriamas skirtinguose OSI modelio sluoksniuose [14]:

1. Taikymo sluoksnyje – VPN tinklas veiks tik su tam tikrom programom, pavyzdžiui naudojant interneto naršyklę ir atliekant kokias nors pinigines operacijas. VPN tinklas yra kuriamas tarp interneto naršyklės ir serverio.
2. Tinklo sluoksnyje – VPN tinklas šifruos bet kokius perduodamus duomenis, nepriklausomai nuo programos.
3. Ryšio sluoksnyje – naudojamas sujungti 2 VPN tinklus.

VPN protokolai:

- L2TP (Layer 2 Tunneling Protocol) Antrame OSI sluoksnyje veikiantis protokolas, dažniausiai naudojamas interneto tiekėjų. Naudojamas kaip atviras tinklas, nes neužtikrina konfidencialumo.

- L2TP/IPsec (Internet Protocol Security) L2TP protokolas papildomas su IPsec protokolo savybėmis: konfidencialumu, autentiškumu, vientisumu.
- PPP (Point-to-point protocol) naudojamas tik tiesioginiams susijungimams. Palaiko autentifikavimą, šifravimą ir suspaudimą. Veikia antrame OSI sluoksnyje.
- PPTP (Point-to-point tunelling protocol) dirbantis antrame OSI modelio sluoksnyje. Šis protokolas apjungia PPP ir IP protokolus naudojant naudojant GRE (Generic Routing Encapsulation) sutraukimo protokolą. Saugumas palaikomas tik PPP protokolu.
- MPPE (Microsoft Point-to-Point Encryption) naudojamas duomenų šifravimui naudojant PPP ar PPTP protokolus. Šifravimas galimas 128-bit, 56-bitų, 40-bit raktais.
- SSTP (Secure Socket Tunneling Protocol) dirbantis 4 OSI sluoksnyje, skirtas VPN tinklams naudojančius PPP ar L2TP šifruoti ir tikrinti integralumą naudojant SSL 3.0.
- SSLv3/TLSv1 kriptografiniai protokolai skirti komunikacijos šifravimui, autentifikavimui. Žinučių integralumui palaikyti. Sukuria sesija 5 OSI modelio sluoksnyje, dirba 6 OSI sluoksnyje.
- HTTPS (Hypertext Transfer Protocol Secure) veikia 7 OSI modelyje. Skirtas tekstinei, grafinei informacijai perduoti. Naudoja SSL/TLS protokolą. [15]

Pasirinkta programinė įranga OpenVPN, kuri naudoja SSL/TLS VPN konfidencialumui, autentifikavimui ir integralumui užtikrinti. OpenVpn tunelis gali veikti viename iš dviejų OSI sluoksnių: antrajame arba trečiajame. Trečiajame sluoksnyje kuriamas tunelis (nustatymuose: TUN) yra paremtas IP protokolu, ir gali veikti su bet kokia tinklo plokšte kaip sąsaja. Antrajame sluoksnyje kuriamas tunelis (TAP), perduoda bet kokį duomenų srautą per „Ethernet“ sąsaja.

2.7. Išvados

1. Naudojant duomenų perdavimui *UA TCP* protokolą perduodamų duomenų kiekis yra mažesnis, nei *SOAP/HTTP,* todėl yra tikslinga jį naudoti mobiliuosiuose įrenginiuose, kur perduodamų duomenų kiekis yra labai svarbus.
2. OpenVPN puikiai tinka naudojant kartu su OPC UA. Jį sunku atskirti nuo standartinių HTTPS ir dėl to yra labai sunku jį užblokuoti. Taip pat yra pritaikytas įvairioms platformoms ir yra lengvai konfigūruojamas ir pritaikomas.
3. Integravus VPN į OPC UA yra supaprastinamas OPC UA saugumo realizavimas.
4. Simetriniui šifravimui OPC UA vartotojams siūlo AES-128-CBC ir AES-256-CBC algoritmus, tuo tarpu OPEN VPN vartotojams siūlo: RC2-40-CBC, CAST5-CBC, RC2-64-CBC, AES-128-CBC ir kt.
5. VPN negali realizuoti sąlyginių prieigų, todėl norint jas įdiegti yra naudojama OPC UA.

3. OPC UA SISTEMOS PROJEKTAS

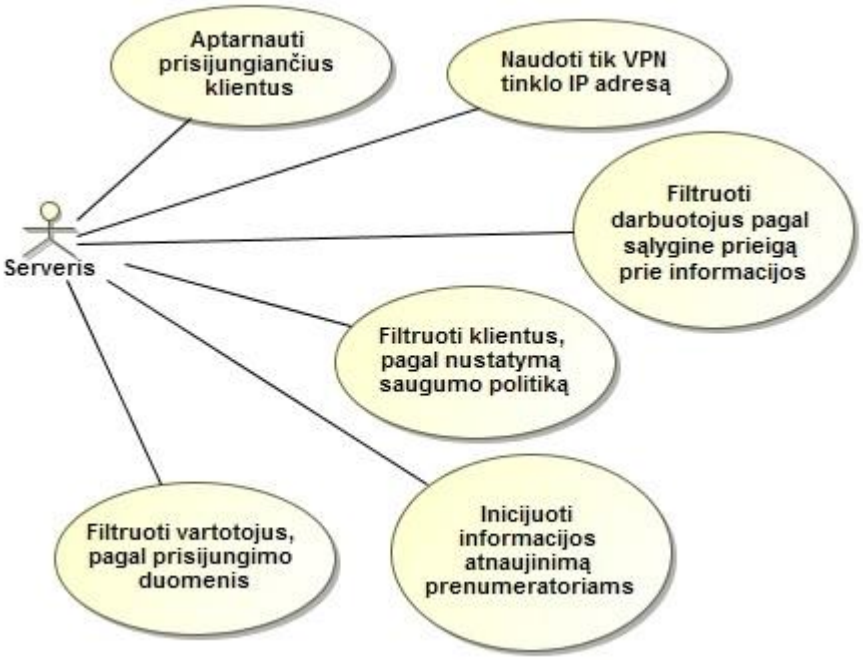
3.1. Projekto tikslas

Projekto tikslas yra sukurti OPC UA klientą ir serverį. Perduoti saugumo užtikrinimą iš OPC UA į VPN tinklą.

3.2. Reikalavimai sistemai ir sistemos funkcijos

OPC UA sistemos projektui surinkti reikalavimai:

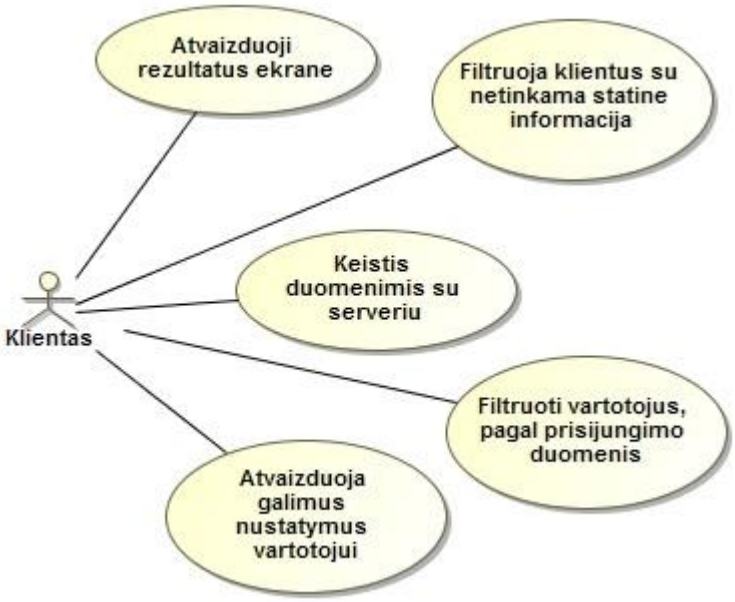
1. Turi veikti naudodama kliento - serverio architektūrą.
2. VPN tinklui sukurti naudojamas OpenVpn programinė įranga.
3. Sistema duomenų apsikeitimui tarp kliento ir serverio turi naudoti UA TCP protokolą.
4. Sistema turi naudoti atviro kodo ir nemokamą programinę įrangą.
5. Sistema turi taupiai naudoti mobilaus įrenginio baterijos energiją.
6. Klientinė programa turi veikti su Android OS operacine sistema.
7. Klientinė programa rodo pranešimus ekrane ar pavyko prisijungti prie serverio.
8. Klientinė programa leidžia įkelti savo privataus ir viešojo raktų porą, sertifikatą.
9. Klientinė programa leidžia pasirinkti šifravimo algoritmą.
10. Klientinė programa leidžia pasirinkti kokius kredencialus naudoti jungiantis prie serverio.
11. Klientinė programa leidžia vartotojui pasirinkti ar norima duomenų prenumeratos.
12. Klientinė programa matuoja siunčiamų duomenų greičius.
13. Vartotojas pasirinkęs norimą saugumo politiką jungiasi prie serverio.
14. Klientinė programa gali jungtis prie bet, kurio OPC UA serverio, įvedant jo adresą.



PAV. 5 Serverio panaudos atvejų diagrama

Serverio pagrindinės funkcijos yra šios:

1. Aptarnauti prisijungiančius klientus.
2. Naudoti tik VPN tinklo IP adresą.
3. Filtruoti klientus, pagal nustatymą saugumo politiką.
4. Filtruoti vartotojus, pagal prisijungimo duomenis.
5. Inicijuoti informacijos atnaujinimą prenumeratoriams (klientams).



PAV. 6 Kliento panaudos atvejų diagrama

Kliento pagrindines funkcijos yra šios:

1. Komunikuoti su serveriu.
2. Atvaizduoti iš serverio gautus duomenis.

3. Priimti saugumo nustatymus.

3.3. Nefunkciniai reikalavimai

1. Paprasta, greitai perprantama vartotojo sąsaja.
2. Vartotojams turi būti lengvai suprantama reikalaujama konfigūracija.
3. Paaiškinimai, pavadinimai ir terminai anglų kalba.
4. Efektyvus aparatinės įrangos panaudojimas.
5. Kiek įmanoma aukštesnis patikimumas.
6. Kūrimo metu turi būti laikomasi IEC 61131 standartų.

3.4. Panašūs projektai

Yra keletas panašių projektų, kaip OPC ar OPC UA skirtų pramonei, duomenims perduoti tarp įvairių platformų.

3.4.1. MQTT

MQTT (Message Queue Telemetry Transport) – žinučių protokolas. Protokolą sukūrė dr. Andy Stanfor-Clark iš „IBM“ įmonės ir Arlen Nipper iš „Eurotech“ įmonės. Protokolas taip pat skirtas pramonei. [16]

Protokolo savybės:

- Yra realizuotas įvairiuose platformose.
- Paprastas ir neužimantis daug vietos žinučių protokolas
- Dirba virš TCP/IP protokolo
- Skirtas įterptinėms sistemoms, maksimaliai sumažindamas įrenginio išteklius.
- Suprojektuotas tinklams, kuriuose perdavimo greitis yra lėtas, yra didelis vėlavimas, kur duomenų perdavimas nėra patikimas.
- Protokolas taip pat naudojamas „Daiktų internete“ ir programose mobiliems įrenginiams, pavyzdžiui: „Facebook Messenger“

Šis protokolas yra tik kaip transporto protokolas. Nėra realizuoti jokie saugumo mechanizmai. Taip pat neturi aukštesnio lygio funkcijų, tokių kaip informacijos modeliavimas.

3.4.2. MTConnect

MTConnect yra gamybos pramonės standartas skirtas lengviau perteikti valdiklių apdorotą informaciją. Kūrėjai David Edstrom iš „Sun Microsystems“ ir Dr. David Patterson iš „Berkeley“ universiteto, Kalifornija. [17]

Protokolo savybės:

- Paprastas ir neužimantis daug vietos protokolas
- Lengvai plečiamas
- Skirtas duomenims tarp valdiklių perduoti
- Tik skaitymo galimybė
- Visiškai atviras ir nemokamas
- Duomenys perduodami XML duomenų schemomis
- Informacija į aplikacijos sluoksnį perduodama per HTTP protokolą, naudojant „Agentus“
- Nėra jokių prisijungimų
- Nėra sesijų

Protokolas skirtas duomenų atvaizdavimui, rinkimui ir analizei, t.y. negalima daryti jokių nustatymų gamybos lygyje. Nėra jokių saugumo mechanizmų.

3.4.3. REST-PCA

Protokolas taip pat sukurtas „OPC Foundation“. Sukurtas kaip alternatyvus modelis skirtas situacijoms, kur OPC UA sistema nesuderinama ar netinkama. [18]

Protokolo savybės:

- Naudoja REST(REpresentational State Transfer) protokolą
- Atviras ir nemokamas
- Saugumas gali buti atliekamas numatytu HTTP protokolu:
    1. HTTP autentifikacija naudojant vartotojo vardą ir slaptažodį
    2. HTTP autentifikacija naudojant klientams suteiktu API raktu
    3. HTTP autentifikacija naudojant klientams suteiktu ratu, taip pat kiekviena užklausa pasirašoma su serveriui žinomu klientu raktu.
- Lengvai realizuojamas su trečių šalių programomis.
- Veikia tiek su XML, tiek su Json.

Protokolas neskirtas duomenims perduodi iš valdiklio į valdiklį. Veikia pakankamai lėtai lyginant su UA TCP.

3.5. Sistemos architektūros aprašymas

Sistema veikia naudojant kliento – serverio architektūrą. Jos veikimo principas:

- Duomenų perdavimui yra naudojamas UA TCP protokolas.
- Vartotojas naudojant OpenVPN protokolą prisijungia prie VPN.
- OPC UA serveris naudoja tik VPN tinklo priskirtą IP adresą.
- Vartotojas prisijungia prie OPC UA serverio naudojant OPC UA klientą.
- Serveris pagal nutylėjimą gražiną pradinį mazgą (root node).

- Vartotojas gali naršyti po serverį, ieškoti reikalingos informacijos, prenumeruoti duomenų pasikeitimus.
- Atliekamas prisijungimo veiksmas naudojant pasirinktą saugumo politiką.
- Serveris pagal nutylėjimą gražina adresų erdvės (Adress Space) .

3.5.1. Serverio architektūros aprašymas

Serveryje naudojamos šios technologijos:

- Prosys OPC UA Java SDK Client Server Evaluation - 2.0.0 – 194 biblioteka.
- OpenVpn klientas.
- Java virtuali mašina.

Pagrindinės serverio atliekamos operacijos yra dvi: prisijungiančių klientų aptarnavimas ir informacijos atnaujinimas, pridėjimas, šalinimas. Prisijungiančių klientų aptarnavimas yra atliekamas klientui jungiantis prie serverio 52520 – ąją jungtį. Jei klientas sėkmingai prisijungia prie serverio vartotojas gali naršyti po serverį ir atlikti autorizuotas komandas, šalinti, pridėti, atnaujinti norimas reikšmes.

3.5.2. Kliento architektūros aprašymas

Kliente naudojamos šios technologijos:

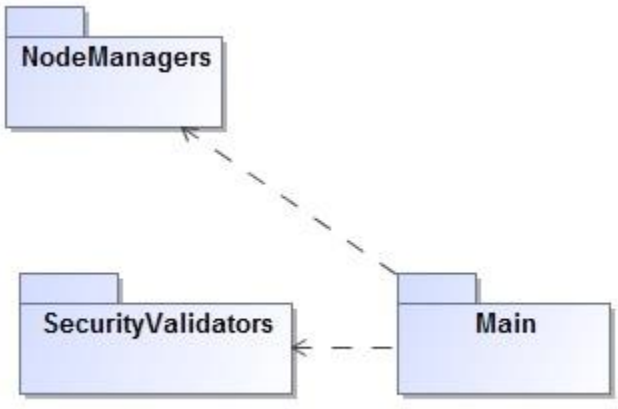
- Prosys OPC UA Java SDK Client Server Evaluation - 2.0.0 – 194 biblioteka.
- Android SDK („Android“ operacinei sistemai nuo 4.0.0 (Ice Cream Sandwich) OS)
- OpenVpn klientas.

Pagrindinės kliento atliekamos operacijos yra duomenų atvaizdavimas iš serverio.

Vartotojas naudodamasis klientine programa gali pasirinkti norimą saugumo politiką, gali įvesti vartotojo autorizacijai reikalingus vartotojo vardą ir slaptažodį.

3.6. OPC UA serverio architektūra

Serverio komponentų diagrama:



PAV. 7 Serverio pagrindinė komponentų diagrama

Serveris sudarytas iš kelių pagrindinių komponentų:

- SecurityValidators. Sistemos komponentas atsakingas už prisijungiančių vartotojų ir prisijungiančių aplikacijų autentiškumo ir autorizacijos tikrinimus.
- NoteManagers. Sistemos komponentas skirtas duomenų pasiekimo iš klientinės programos valdymui. Naudojantis šiuo komponentu, objektai yra pridedami, šalinami, ieškomi.
- Main. Pagrindinis sistemos komponentas. Inicijuojamas OPC UA serveris, ir sujungiami visi komponentai.

3.6.1. SecurityValidators komponentas

Šio paketo tikslas yra apdoroti prisijungiančių vartotojų ir aplikacijų sertifikatus, prisijungimo duomenis. Yra tikrinama kurie sertifikatai yra priimami, kurie yra atmetami. Taip pat tikrinama, kokiu būdu vartotojas prisijungia ir tikrinama vartotojo prisijungimo duomenys, priklausomai ar jungiamasi naudojant sertifikatą ar prisijungimo vardą ir slaptažodį.

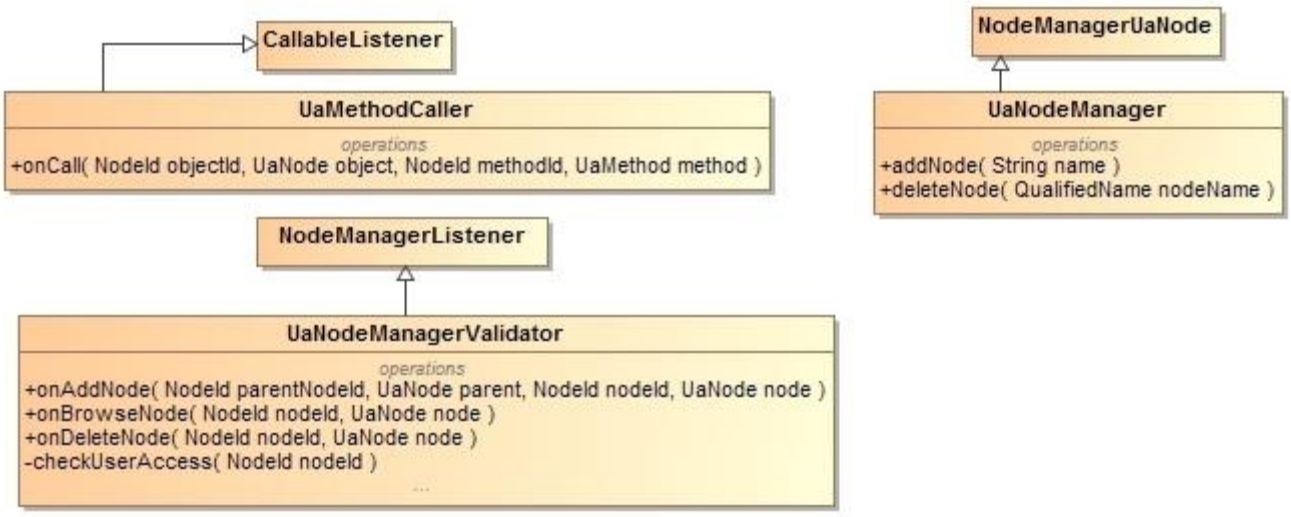
PAV. 8 SecurityValidaros komponento diagrama

1. lentelė. SecurityValidaros komponento metodų aprašymas

Klasė	Metodas	Aprašymas
ApplicationCertificateValidator	onValidate (Cert certificate) : ValidationResult	Jei klientas jungiasi saugiu kanalu yra tikrinamas jo sertifikatas.
UaUserValidator	onValidate (Session session, ServerUserIdentity userIdentity) : boolean	Jei klientas jungiasi naudodamas kokius nors kredencialus, yra tikrinama ar vartotojas gali prisijungti
	onValidationError(Session session, UserIdentityToken userToken) : void	Jei vartotojo validacija nebuvo sėkminga, kviečiamas šitas metodas, kuris praneša kuris vartotojas negavo prisijungimo ir kodėl.

3.6.2. NodeManagers komponentas

Šio komponento paskirtis yra suteikti OPC UA serveriui duomenis, kuriuos jis turi gražinti klientui.



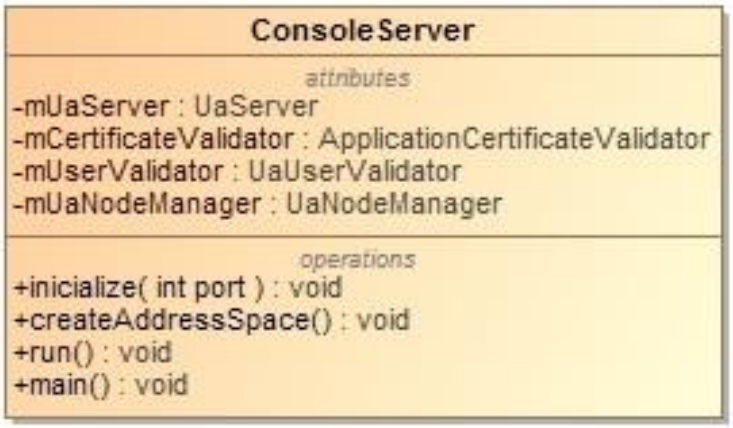
PAV. 9 NodeManagers komponento diagrama

2. lentelė. NodeManagers komponento metodų aprašymas

Klasė	Metodas	Aprašymas
UaMethodCaller	onCall (NodeId objectId, UaNode object, NodeId methodId, UaMethod method)	Klientas inicijuoja dinaminio metodo pakvietimą. Serveris atlieka iškviesta metodą.
UaNodeManager	addNode(QualifiedName name)	Pridedamas „QualifiedName“ objektas, matomas kitiems klientams.
	deleteNode(QualifiedName nodeName)	Ištrinamas „QualifiedName“ objektas
UaNodeManagerValidator	onAddNode(NodeId parentNodeId, UaNode parent, NodeId nodeId, UaNode node)	Prieš vartotojui įterpiant objektą į serverį, yra tikrinamas vartotojas ar jis turi reikalingus leidimus.
	onBrowseNode(NodeId nodeId, UaNode node)	Prieš naršant vartotojui per mazgus “Node“, yra tikrinamas vartotojas ar jis turi reikalingus leidimus.
	onDeleteNode(NodeId nodeId, UaNode node)	Prieš ištrinant vartotojui objektą, yra tikrinama ar klientas
	checkUserAccess(NodeId nodeId)	Tikrinama vartotojo leidimai atlikti tam tikrą veiksmą.

3.6.3. Main komponentas

Šio komponento paskirtis yra inicijuoti OPC UA serverį ir sujungti visus komponentus į vieną visumą.



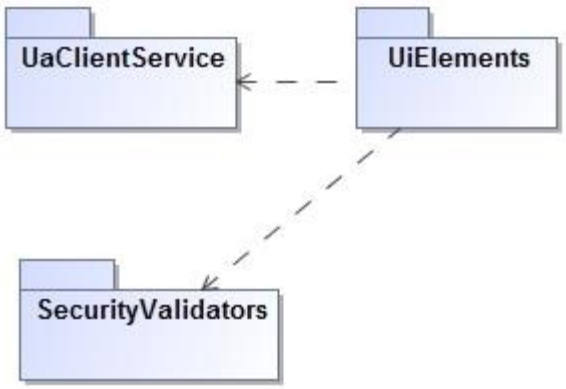
PAV. 10 Main komponento diagrama

3. lentelė. Main komponento metodų aprašymas

Klasė	Metodas	Aprašymas
ConsoleServer	main()	Metodas inicijuojantis visą sistemą
	inicialize(int port)	Sukuriami pagrindiniai objektai reikalingi prieš paleidžiant serverį
	createAddressSpace()	Sukuriamas adresų erdvė, kurioje saugomi objektai.
	run()	Sukuriamas UaServerio objektas, jam priskiriami būtini nustatymai, paleidžiamas.

3.7. OPC UA Kliento architektūra

3.7.1. OPC UA Kliento komponentų diagrama



PAV. 11 Kliento pagrindinė komponentų diagrama

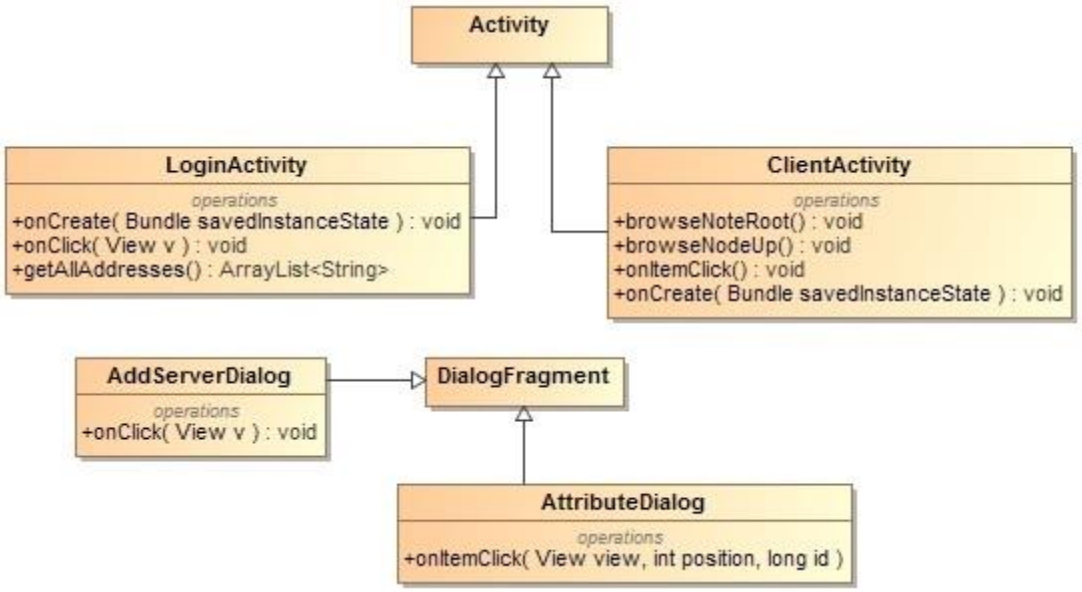
Klientas sudarytas iš kelių pagrindinių komponentų:

- Activities. Sistemos komponentas, atsakingas už Android klientinės programos sąveika su vartotoju.
- SecurityValidators. Sistemos komponentas, atsakingas už OPC UA serverio sertifikato tikrinimą.

- UaClientService. Sistemos komponentas, atsakingas už duomenų perdavimą tarp vartotojo ir serverio.

3.7.2. UiElements komponento diagrama

Šiame pakete esančios klasės yra skirtos realizuoti vartotojo sąsajai, gautai informacijai atvaizduoti. Komponentuose bus atvaizduojami įvairūs laukai, leidžiantis sukonfigūruoti OPC UA kliento prisijungimą prie serverio.



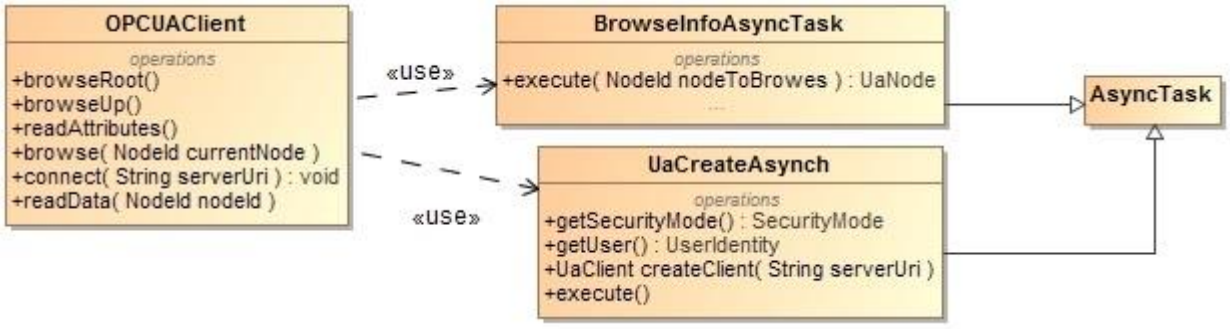
PAV. 12 UiElements komponento diagrama

4. lentelė UiElements komponento metodų aprašymas

Klasė	Metodas	Aprašymas
LoginActivity	onCreate(Bundle savedInstanceState) : void	Skirtas vartotojo sąsajai sukurti, vartotojo sąsajos XML elementams surasti.
	onClick(View v) : void	Skirtas mygtukų paspaudimų realizavimui
	getAllAddresses() : ArrayList<String>	Skirtas išsaugotiems adresams paimti iš talpyklos.
ClientActivity	browseNoteRoot() : void	Skirtas mygtukui, kuris iš serverio gražina pradinius duomenis
	browseNodeUp() : void	Naršant serveryje, šis metodas grįžta vienu žingsniu atgal
	onItemClick() : void	Pasirenkamas norimas iš serverio atvaizduotas elementas.
	onCreate(Bundle savedInstanceState) : void	Skirtas vartotojo sąsajai sukurti, vartotojo sąsajos XML elementams surasti.
AddServerDialog	onClick(View v) : void	Skirtas atidaryti langui, kuriame galima įvesti serverio adresą
AttributeDialog	onItemClick(View view, int position, long id)	Mazgo (node) norimos informacijos atvaizdavimui

3.7.3. UaClientService komponento diagrama

Šiame pakete esančios klasės yra skirtos OPC UA kliento objekto inicijavimui. Atlieka duomenų perdavimus tarp kliento ir serverio. Apdoroja perduotus duomenis.



PAV. 13 UaClientService komponento diagrama

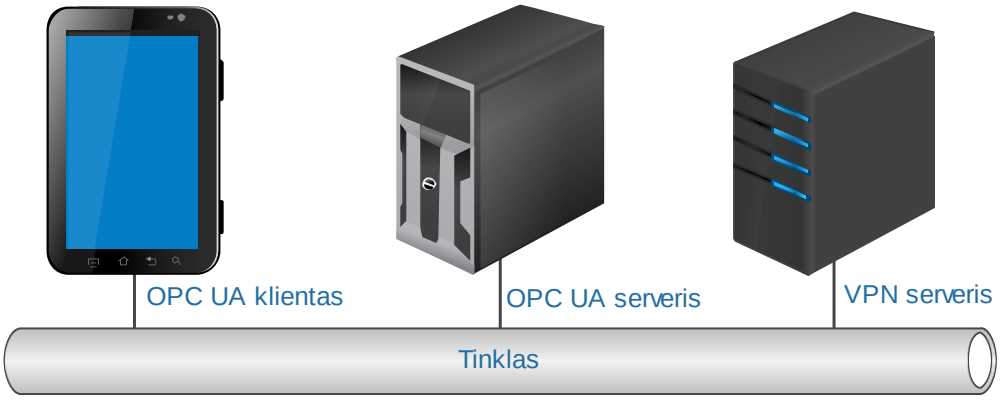
5 lentelė. UaClientService komponento metodų aprašymas

Klasė	Metodas	Aprašymas
OPCUAClient	browseRoot() : void	Vykdo pradinės informacijos gavimą iš serverio
	browseUp() : void	Vykdo grįžimą prie prieš tai buvusio mazgo (node).
	readAttributes() : void	Iškviečia AttributeDialog
	browse(NodeId currentNode) : void	Atliekamas pasirinkto mazgo duomenų naršymas serveryje
	connect(String serverUri) : void	Iškviečiamas jungimąsis prie serverio.
	readData(NodeId nodeId) : void	Nuskaitomos mazgo reikšmės
BrowseInfoAsyncTask	execute(NodeId nodeId) : UaNode	Skirtas asinkroniškai naršyti po serverį.
UaCreateAsychTask	getSecurityMode() : SecurityMode	Gaunama pasirinkta saugumo politika ir nustatomą į serverį.
	getUser() : UserIdentity	Gaunami įvesti vartotojo duomenys.
	createClient(String serverUri) : UaClient	Kuriamas OPC UA klientas asichroniškai su nurodytu serverio adresu, nes gali būti generuojama privataus ir viešo raktų pora
	execute() : void	
CertificateValidator	onValidate(Cert certificate) : ValidationResult	

3.8. VPN tinklas

VPN tinklo sukūrimui yra naudojama OpenVpn programinė įranga. VPN tinklo sukūrimui naudojamas atskiras VPN serveris *14 pav.*. Kiekvienas esantis elementas VPN tinkle turi būti sukonfigūruotas. OpenVPN programinės įrangos konfigūravimui yra naudojamas konfigūracinis

failas su .ovpn plėtiniu. Failas gali būti kuriamas su bet kuriu pasirinktu teksto redaktoriumi, išsaugant failą su plėtiniu .ovpn.



PAV. 14 VPN tinklo modelio diagrama

3.8.1. VPN serverio konfigūracija

VPN serverio konfigūracinio failo duomenys pateikti *pav. 15.*

server.ovpn server.ovpn

```
1  local 158.129.25.87
2  port 1194
3  proto udp
4  dev tun
5  dev-node "Local Area Connection 3"
6  ca ca.crt
7  cert vpn-server.crt
8  key vpn-server.key
9  dh dh1024.pem
10 server 10.8.0.0 255.255.255.0
11 ifconfig-pool-persist ipp.txt
12 keepalive 10 120
13 cipher AES-128-CBC   # AES
14 persist-key
15 persist-tun
16 client-to-client
17 comp-lzo
18 max-clients 10
19 status openvpn-status.log
20 verb 6
21 client-config-dir ccd
```

PAV. 15 VPN serverio konfigūracinis failas

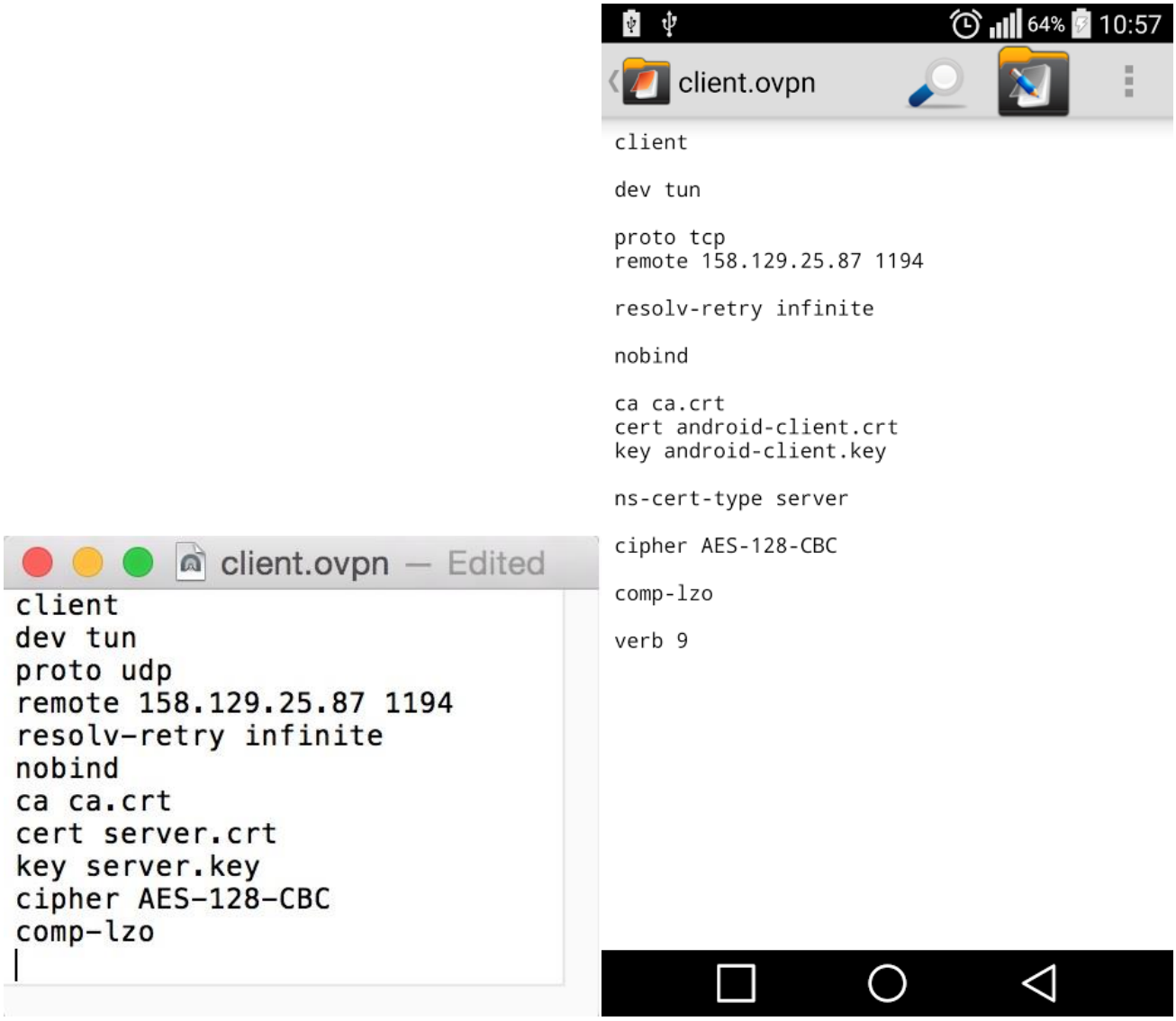
6 lentelė. VPN serverio konfigūracijos aprašymas

Parametras	**Reikšmė**	**Paskirtis**
local	158.129.25.87	Serverio išorinis IP adresas, per kurį bus perduodami duomenys
port	1194	Prievadas per kurį bus perduodami duomenys
proto	udp	Naudojamas UDP protokolas
dev	tun	Naudojamas nukreipimas į IP tunelį

dev-node	Local Area Connection 3	Windows sistemoje turi būti nurodomas tinklo adapteris, kuris naudoja pasirinktą IP adresą.
ca	ca.crt	sertifikato centro sertifikatas
cert	vpn-server.crt	VPN serverio sertifikato failas
key	vpn-server.key	VPN serverio privatus raktas
dh	dh1024.pem	Failas saugantis sugeneruotus Diffie ir Hellman raktų apsikeitimo parametrus
server	10.8.0.0 255.255.255.0	Tinklo sietuvo ir potinklio adresai. Serveriui suteikiamas 10.8.0.1 adresas.
ifconfig-pool-persist	ipp.txt	Saugomi VPN klientų IP adresai, kad prisijungus kelis kartus būtų priskiriamas tas pats.
Keep-alive	10 120	Nustatoma, kad ryšio tikrinimas yra atliekamas kas 10 sekundžių, o neesant atsakui po 120 sekundžių yra manoma, kad klientas yra išjungtas
cipher	AES-128-CBC	Nurodoma kokiu šifravimo algoritmu, bus šifruojamas kanalas.
client-to-client	-	Leidžia sudaryti susijungimus tarp klientų
comp-lzo	-	Yra naudojamas perduodamų duomenų suspaudimas
client-config-dir	ccd	Direktorijos pavadinimas, kurioje saugoma, kokie statiniai adresai yra priskiriami pasirinktiems klientams.

„ccd“ direktorijoje yra sukuriamas failas su teksto redaktoriumi, kuriame yra nurodomas koks IP adresas yra suteikiamas. Faile yra nurodomas parametras „ifconfig-push“, su reikšme 10.8.0.100. Išsaugoto failo vardas yra kliento sertifikate nurodytas vardas ir be jokio išplėtimo. OpenVpn taikomoji programa priskiria faile nurodytą statinį IP adresą norimam VPN klientui.

3.8.1. VPN klientų konfigūracija



PAV. 16 VPN klientų konfigūraciniai failai

OPC UA kliento ir OPC UA serverio VPN konfigūraciniai failai yra panašūs. Skirtingi failai yra nurodomi tik cert ir key parametruose, nurodomas kiekvieno atskirai sugeneruotas sertifikatas su privatus raktu.

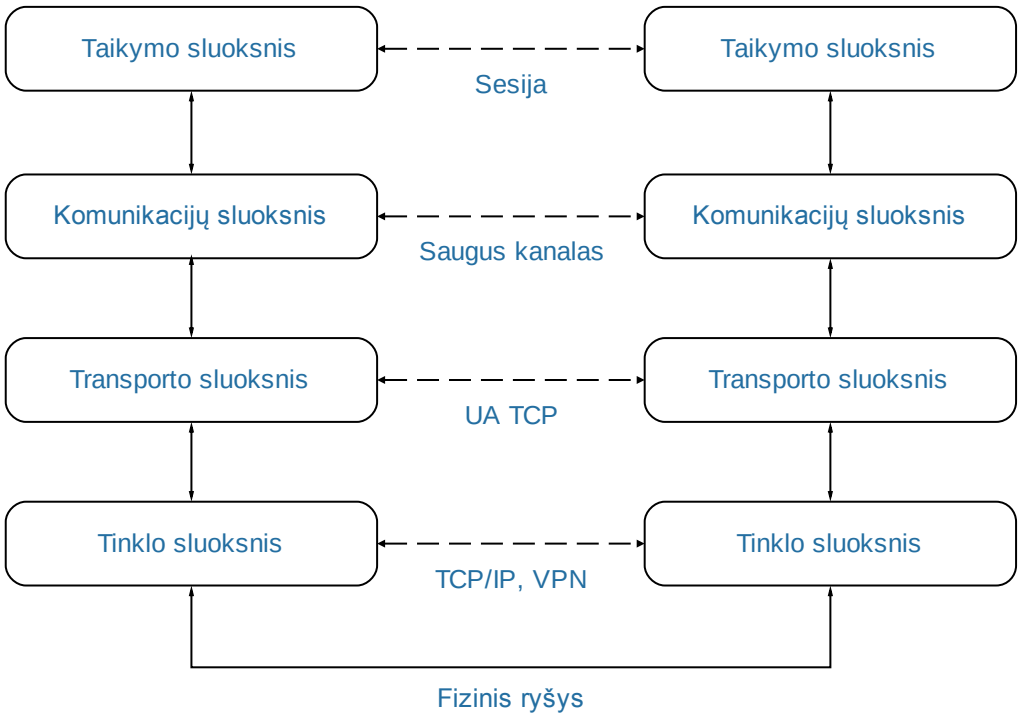
7 lentelė. VPN klientų konfigūracijos aprašymas

Parametras	**Reikšmė**	**Paskirtis**
client	-	Nurodoma, kad OpenVpn bus naudojamas kaip klientas
dev	tun	Naudojamas nukreipimas į IP tunelį
proto	udp	Naudojamas UDP protokolas
remote	158.129.25.87 1194	Nurodomas VPN serverio IP adresas ir prievadas
resolv-retry	infinite	Parametras skirtas nurodyti kiek, kartu bandyti prisijungti prie serverio
nobind	-	Nurodo, kad galima naudoti, bet kurį kliento prievadą
ca	ca.crt	sertifikato centro sertifikatas

cert	server.crt android-client.crt	Kliento sertifikatas
key	server.key android-client. key	Kliento privatus raktas
cipher	AES-128-CBC	Nurodoma kokiu šifravimo algoritmu, bus šifruojamas kanalas.
comp-lzo	-	Yra naudojamas perduodamų duomenų suspaudimas

3.8.2. OPC UA ir VPN tinklo modelis

OPC UA ir VPN modelį galima atvaizduoti, kaip keturių sluoksnių modelį 17 pav.



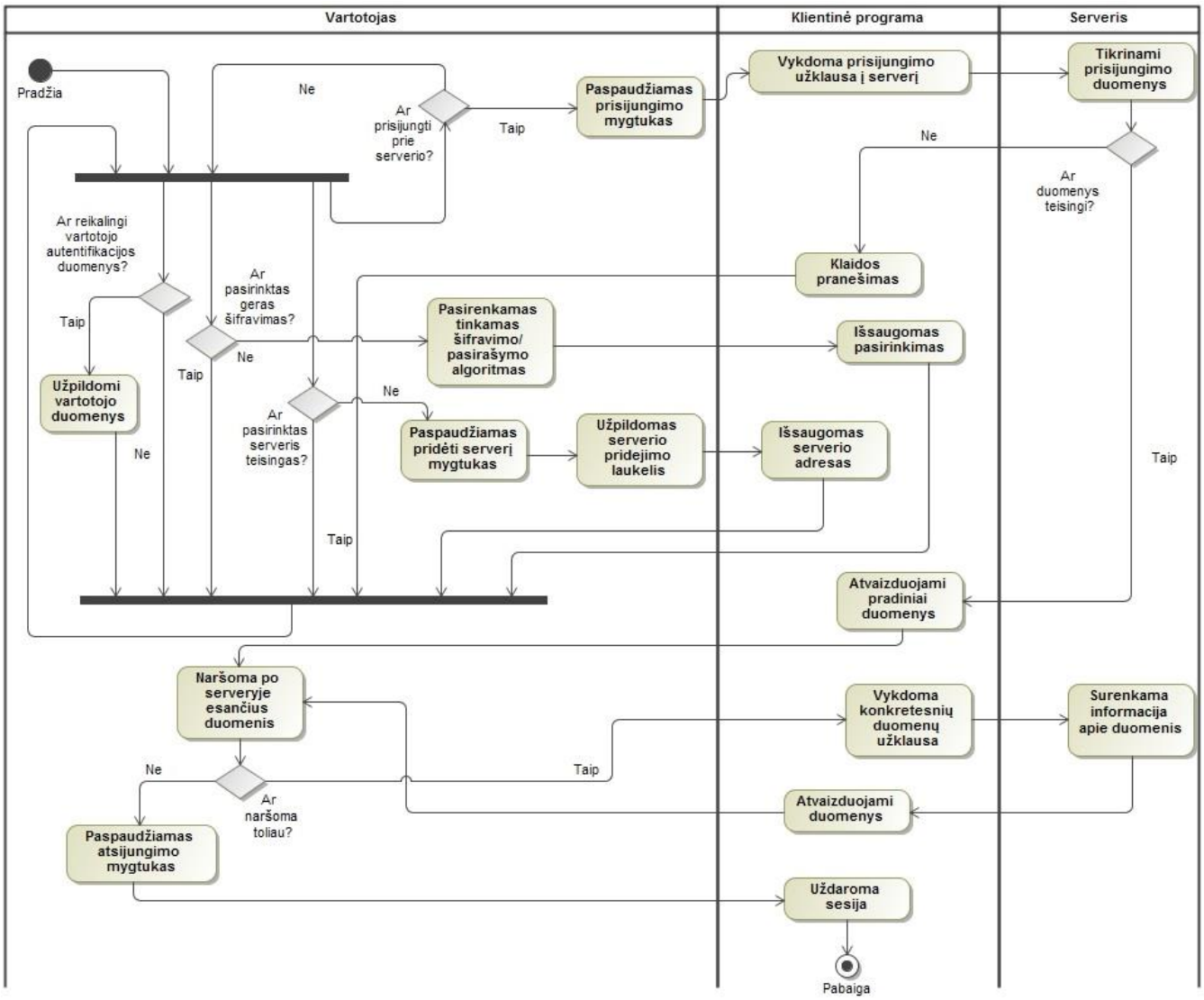
PAV. 17 OPC UA ir VPN tinklo modelis

Taikymo sluoksnyje yra OPC UA kliento ir serverio programos. Naudodamos sesiją perduoda duomenis tarpusavyje. Komunikacijų sluoksnis, OPC UA pritaiko pasirinktą saugumo politiką, ir vykdo politikos reikalavimus (pasirašo žinutes, šifruoja duomenis). Transporto sluoksnis – perduoda duomenis dvejetainiu kodu UA TCP protokolu. Tinklo sluoksnis – duomenų perdavimas per VPN, kuris šifruoja visus perduodamus duomenis, nurodytu šifravimo algoritmu.

3.9. Sistemos dinaminis vaizdas

Šiame skyriuje pateikiamos OPC UA sistemos veikimo principas sudarytas iš trijų pagrindinių dalių, iš kurių kiekvienas turi jam priskirtas užduotis:

1. Vartotojo
2. Klientinė programos
3. Serverio



PAV. 18 OPC UA serverio ir kliento veiklos diagrama

3.10. Išvados

1. Integravus VPN, OPC UA saugumas nebereikalauja realizacijos, todėl tai sumažina projekto kūrimo laiką.
2. Palaikant pastovų ryšį tarp OPC UA kliento mobiliajame įrenginyje ir OPC UA serveryje, mobiliojo įrenginio baterija yra eikvojama kur kas sparčiau.
3. UA TCP protokolas leidžia sumažinti perduodamų duomenų kiekį per internetą.
4. Kiekvienam VPN klientui reikią generuoti privačių/viešų raktų poras.
5. VPN tinklas leidžia OPC UA serveriams turėti dinaminius IP adresus, todėl net ir serveriui keičiant vietą ar tinklą jis gali būti pasiekimas.
6. Naudojant vienu metu OPC UA saugumą ir VPN saugumą galima atlikti dvigubą duomenų šifravimą.

4. EKSPERIMENTINĖ DALIS

4.1. Techninė ir programinė įranga

Sistemai testuoti buvo naudojama:

- LG G3 mobilusis telefonas. Mobiliojo telefono parametrai pateikti prieduose 8 lentelė.
- Testavime naudotas kompiuteris su AMD Phenom II x4 965 (4 branduolių po 3,4 Ghz.) procesoriumi, 12GB RAM, Windows 7 SP1 Ultimate 64-ių bitų operacinė sistema.

4.2. Realizacija

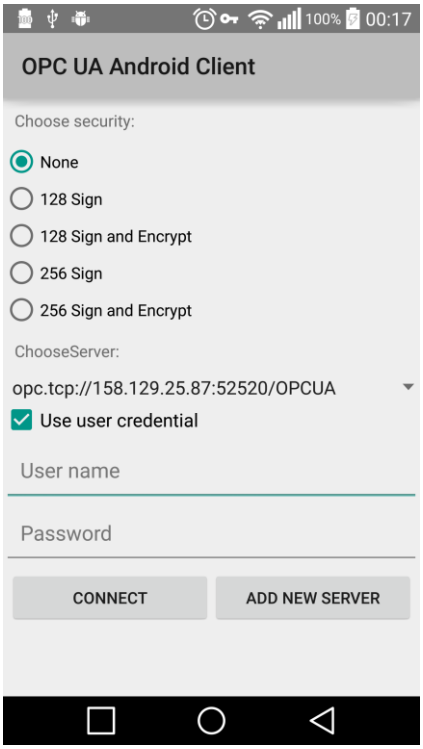
Realizuota sistema veikia naudojant klientas – serveris architektūra.

- OPC UA serveris yra realizuotas naudojant JAVA programavimo kalbą ir Prosys-OPC-UA-Java-SDK-Client-Server-Evaluation-2.1.0-436.jar biblioteką. Veikia kaip komandinės eilutės programa. VPN tinklas realizuotas naudojant OpenVpn programinę įrangą.
- Mobiliam telefonui skirtas OPC UA klientas parašytas naudojant JAVA programavimo kalbą, Android SDK ir Prosys-OPC-UA-Java-SDK-Client-Server-Evaluation-2.1.0-436.jar biblioteką. VPN tinklas realizuotas naudojant „OpenVPN for Android“ Android programėlę.

Android OPC UA kliento vartotojo sąsaja pateikta 19 pav. Paveikslėlyje pavaizduota OPC UA saugumo politikos kurią atitinkamai pasirinkus yra jungiamasi prie serverio.

- None – nenaudojama jokia saugumo politika.
- 128 Sign – naudojamas „Basic128RSA15“ žinučių pasirašymui.
- 128 Sign and Encrypt – naudojamas „Basic128RSA15“ tik žinučių pasirašymui ir šifravimui.
- 256 Sign – naudojamas „Basic256“ žinučių pasirašymui.
- 256 Sign and Encrypt - naudojamas „Basic256“ žinučių pasirašymui ir šifravimui.

Kitame laukelyje nurodomas OPC UA serverio adresas prie kurio yra jungiamasi, jei sąraše jo nėra, jis gali būti įvedamas paspaudžiant „ADD NEW SERVER“ mygtuką. Vėliau yra nurodoma ar bus naudojami vartotojo prisijungimo duomenys, ar bus jungiamasi anonimiškai.



PAV. 19 OPC UA „Android“ kliento vartotojo sąsaja

OPC UA komandinės eilutės pavyzdys pateiktas atliekant prisijungimus su OPC UA klientu 20 pav. Klientas prisijungęs prie serverio per VPN tinklą su IP adresu 10.8.0.100. Klientas nenaudodamas VPN per standartinį, šiuo atveju 158.129.20.229.

```
05/13/2015 22:32:14.882 INFO  Session activated: SampleConsoleClient Session1 (ID=ns=1;
 g=1b52b530-a77d-4363-9eb1-4b54f4e70c28 Token=i=1319839322 Channel=(SecureChannelId=12 State=Open URL=opc
 .tcp://Vytautokas-PC:52520/OPCUA/SampleConsoleServer RemoteAddress=/158.129.20.229:36288))
05/13/2015 22:32:19.357 INFO  Session closed: SampleConsoleClient Session1 (ID=ns=1;
 g=1b52b530-a77d-4363-9eb1-4b54f4e70c28 Token=i=1319839322 Channel=(SecureChannelId=12 State=Open URL=opc
 .tcp://Vytautokas-PC:52520/OPCUA/SampleConsoleServer RemoteAddress=/158.129.20.229:36288))
05/13/2015 22:32:37.390 INFO  Session created: SampleConsoleClient Session2 (ID=ns=1;
 g=2d641eb1-4b07-462e-9ee9-6b5a4e5ccfed Token=i=1319839323 Channel=(SecureChannelId=13 State=Open URL=opc
 .tcp://Vytautokas-PC:52520/OPCUA/SampleConsoleServer RemoteAddress=/158.129.20.229:36291))
onValidate: userIdentity=Type=Anonymous
05/13/2015 22:32:37.423 INFO  Session activated: SampleConsoleClient Session2 (ID=ns=1;
 g=2d641eb1-4b07-462e-9ee9-6b5a4e5ccfed Token=i=1319839323 Channel=(SecureChannelId=13 State=Open URL=opc
 .tcp://Vytautokas-PC:52520/OPCUA/SampleConsoleServer RemoteAddress=/158.129.20.229:36291))
05/13/2015 22:39:39.010 INFO  Session created: SampleConsoleClient Session1 (ID=ns=1;
 g=3fd6389c-aee7-43d0-b74a-ce4a5aed81db Token=i=1319839324 Channel=(SecureChannelId=15 State=Open URL=opc
 .tcp://Vytautokas-PC:52520/OPCUA/SampleConsoleServer RemoteAddress=/10.8.0.100:64992))
onValidate: userIdentity=Type=Anonymous
05/13/2015 22:39:39.040 INFO  Session activated: SampleConsoleClient Session1 (ID=ns=1;
 g=3fd6389c-aee7-43d0-b74a-ce4a5aed81db Token=i=1319839324 Channel=(SecureChannelId=15 State=Open URL=opc
 .tcp://Vytautokas-PC:52520/OPCUA/SampleConsoleServer RemoteAddress=/10.8.0.100:64992))
05/13/2015 22:46:07.868 INFO  Session created: SampleConsoleClient Session1 (ID=ns=1;
 g=141af29f-11cd-4d1f-a894-9822e2556b1f Token=i=1319839325 Channel=(SecureChannelId=17 State=Open URL=opc
 .tcp://Vytautokas-PC:52520/OPCUA/SampleConsoleServer RemoteAddress=/10.8.0.100:65013))
onValidate: userIdentity=Type=Anonymous
```

PAV. 20 OPC UA serverio išvesties pavyzdys prisijungimų metu

4.3. Eksperimento uždaviniai

Realizuotoje sistemoje:

- Išmatuoti duomenų perdavimo greičius tarp OPC UA serverio ir kliento naudojant OPC UA saugumo politikas “Base128Rsa15“ ir „Basic256“
- Išmatuoti duomenų perdavimo greičius tarp OPC UA serverio ir kliento naudojant VPN tinklą su šifravimo algoritmais: RSA-128-CBC, RSA-192-CBC ir RSA-256-CBC
- Palyginti perdavimo greičius naudojant skirtingus duomenų perdavimo būdus.

4.4. Eksperimento eiga

Eksperimento metu klientas parsisiųsdamas iš serverio didelį kiekį duomenų matuoja, per kiek laiko, kokį kiekį duomenų jis parsisiuntė ir surašo gautus duomenis į failą. Eksperimentas yra atliekamas šešis kartus skirtingomis sąlygomis t.y.:

1. Nenaudojant jokio šifravimo
2. OPC UA saugumo politiką „Base128Rsa15 Sign and Encrypt“ (BASIC128RSA15_SIGN_ENCRYPT).
3. OPC UA saugumo politiką „Basic256 Sign and Encrypt“ (BASIC256_SIGN_ENCRYPT).
4. Naudojant VPN tinklo šifravimą AES-128-CBC (cipher AES-128-CBC)
5. Naudojant VPN tinklo šifravimą AES-192-CBC (cipher AES-192-CBC)
6. Naudojant VPN tinklo šifravimą AES-256-CBC (cipher AES-256-CBC)

Eksperimento metu bus naudojamas skirtingas naudojamo buferio dydis, didinat jį 20%. Vienu su kiekvienu iš buferio dydžių bus atliekamas 50 kartų matavimas ir vedamas vidurkis.

4.5. Rezultatai ir jų analizė

Eksperimento surinktų duomenų atitinkantis grafikas yra pateiktas 6.2.1 Punkte *21 pav.*

1. Iš 21 pav. matyti, kad naudojant VPN tinklą tarp OPC UA serverio ir kliento duomenų perdavimo greitis yra 20 % mažesnis.
2. Iš 21 pav. matoma, kad naudojant „Base128Rsa15 Sign and Encrypt“, „Basic256 Sign and Encrypt“ ar „None“ saugumo politikas duomenų perdavimo greičiai yra apytikriai panašūs.
3. Iš grafiko matyti, kad perduodant duomenys yra tam tikrų trikdžių. Netoliese esančios Wi-Fi stotelės galėjo darytį įtaka perdavimo greičiui.

4. Eksperimento metu pastebėta, kad naudojant VPN tinklą mobiliajame įrenginyje, yra labai greitai iškraunama baterija, nes mobilusis įrenginys siunčia į serverį pastovias „keepalive“ žinutes, kurios neleidžia įrenginiui pereiti į budėjimo rėžimą ir taip taupyti bateriją

5. IŠVADOS

1. Kuriant OPC UA sistemą integruotą į VPN tinklą yra supaprastinama OPC UA saugumo realizacija, nes VPN tinklas perima autentiškumo, integralumo ir konfidencialumo užtikrinimui reikalingas atlikti užduotis.
2. VPN tinklas leidžia serveriui turėti dinaminį IP adresą.
3. VPN tinklas neturi autorizacijos ir sąlyginės prieigos, todėl šie uždaviniai paliekami OPC UA sistemai.
4. Naudojant VPN tinklą duomenų perdavimo greitis yra ~20 % mažesnis, nei naudojant OPC UA saugumo politikas.

Literatūra

[1] N. Matic, „Mikroe,“ 2003. [Tinkle]. Available: http://www.mikroe.com/old/books/plcbook/plcbook.htm. [Kreiptasi 12 Balandis 2015].

[2] OPC Foundation, OPC Unified Architecture Specification-part 1: Overview and Concepts, IEC 62541-1, 2008.

[3] „The OSI Model's Seven Layers Defined and Functions Explained,“ Microsoft, 13 Birželis 2014. [Tinkle]. Available: https://support.microsoft.com/en-us/kb/103884. [Kreiptasi 15 Balandis 2015].

[4] D. G. Peterson, „OPC UA Owerview,“ Digital Bond, 2008. [Tinkle]. Available: https://www.digitalbond.com/blog/2008/08/14/opc-ua-assessment-series-part-1/. [Kreiptasi 12 Sausis 2015].

[5] „OPC UA Profiles,“ OPC Foundation, [Tinkle]. Available: http://opcfoundation-onlineapplications.org/profilereporting/index.htm?ModifyProfile.aspx%3fProfileID=53605a50-a6ac-44ed-9baa-36c4873ff504. [Kreiptasi 15 Sausis 2015].

[6] R. Armstrong, P. Hunkar, OPC Foundation ir Yokogawa, „The OPC UA Security Model for Administrators v1.00,“ 7 Liepa 2010. [Tinkle]. Available: https://opcfoundation.org/wp-content/uploads/2014/05/OPC-UA_Security_Model_for_Administrators_V1.00.pdf. [Kreiptasi 20 Sausis 2015].

[7] „Spam and Flooding,“ SANS Technology Institute, Security Laboratory, 2007. [Tinkle]. Available: http://www.sans.edu/research/security-laboratory/article/spam-flooding. [Kreiptasi 10 Kovas 2015].

[8] OPC Foundation, OPC Unified Architecture Specification-part 2: Security Model, IEC 62541-2, 2009.

[9] „Explains Eavesdropping,“ Techopedia , [Tinkle]. Available: http://www.techopedia.com/definition/13612/eavesdropping. [Kreiptasi 23 04 2015].

[10] „Alteration Attacks,“ SANS Technology Institute, Security Laboratory, 2007. [Tinkle]. Available: http://www.sans.edu/research/security-laboratory/article/alter-code. [Kreiptasi 10 Kovas 2015].

[11] OPC Foundation, OPC Unified Architecture Specification-part 5: Information Model, IEC 62541-5, 2007.

[12] OPC Foundation, OPC Unified Architecture Specification-part 7: Profiles, IEC 62541-7, 2008.

[13] OPC Foundation, OPC Unified Architecture Specification-part 4: Services, IEC 62541-4, 2008.

[14] „VPN Types and the OSI Model,“ [Tinkle]. Available: https://vpnreviewz.com/vpn-types-and-the-osi-model/. [Kreiptasi 20 Balandis 2015].

[15] „Alternativeto,“ [Tinkle]. Available: http://alternativeto.net/software/openvpn/. [Kreiptasi 18 04 2015].

[16] „MQTT,“ 2014. [Tinkle]. Available: http://mqtt.org/. [Kreiptasi 22 Balandis 2015].

[17] „MTConnect - What is it?,“ The Association for Manufacturing Technology, 12 Birželis 2014. [Tinkle]. Available: http://www.amtonline.org/TechnologyandStandards/TechnologyInitiatives/mtconnct-whatisit.htm. [Kreiptasi 15 Balandis 2015].

[18] T. Tuddenham, „XPCA,“ [Tinkle]. Available: http://www.xpca.org/rest-pca/. [Kreiptasi 15 Balandis 2015].

[19] „OPC the interoperability standard,“ OPC Foundation, 2015. [Tinkle]. Available: https://opcfoundation.org/about/what-is-opc/. [Kreiptasi 10 Sausis 2015].

[20] S.-H. Leitner ir W. Mahnke , „OPC UA – Service-oriented Architecture for Industrial Applications,“ *ABB Corporate Research Center*, 2006.

[21] S.-H. Leitner, W. Mahnke ir M. Damm, OPC Unified Architecture, Berlin, Heidelberg: Springer, ISBN 978-3-540-68898-3, 2009, p. 339.

[22] OPC Foundation, OPC Unified Architecture Specification-part 3: Address Space Model, IEC 62541-3, 2008.

[23] OPC Foundation, OPC Unified Architecture Specification-part 6: Mapping, IEC 62541-6, 2008.

6. PRIEDAI

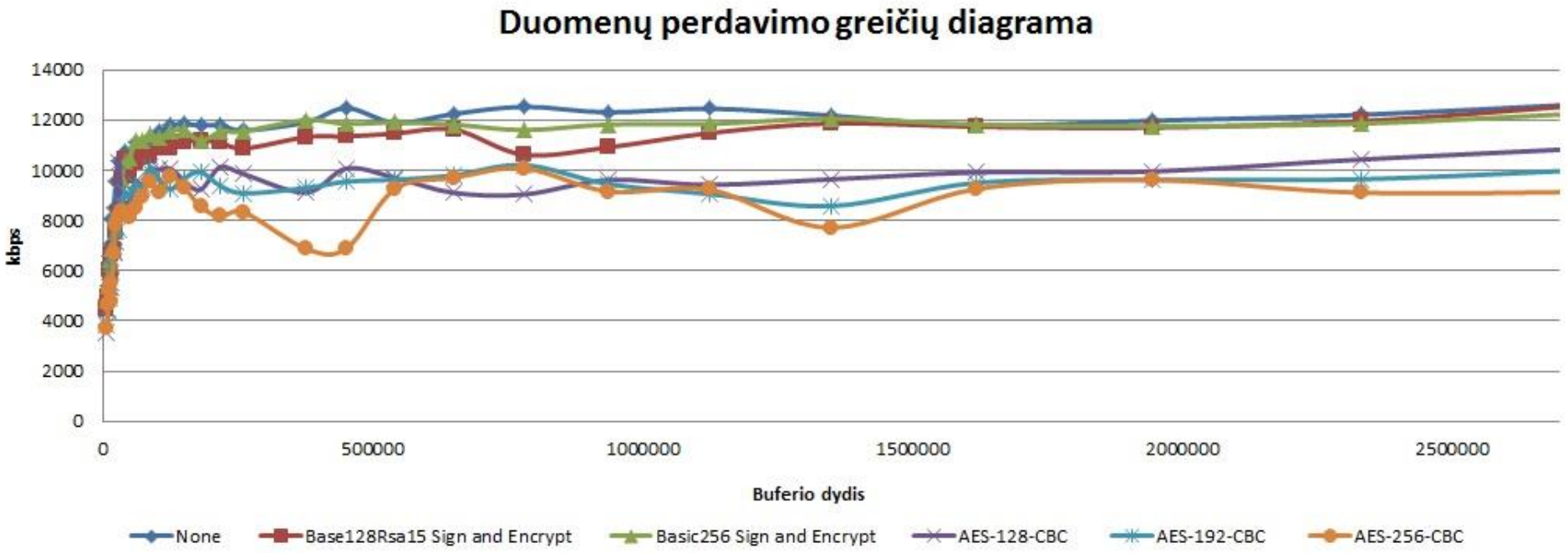
6.1. Mobiliojo telefono duomenys:

8 lentelė. Mobiliojo telefono parametrai

Parametras	**Reikšmė**
2G tinklas	GSM 850 / 900 / 1800 / 1900
3G tinklas	HSDPA 850 / 900 / 1900 / 2100
4G tinklas	LTE 800 / 1800 / 2600
Ekranas	1440 x 2560 pikselių, 5.5 colių (534 ppi pikselių raiška)
Atmintis	Vidinė: 16GB
GPRS	32 kbps
EDGE	48 kbps
HSUPA	21 Mbps
HSDPA	42 Mbps
Wi-Fi	802.11 a/b/g/n/ac, Wifi Direct, DLNA, Wifi-hotspot
OS	Android OS v5.0 (Lollipop)
Lustų rinkinys	Qualcomm MSM8975AC Snapdragon 801
CPU	Quad-core 2.5 GHz Krait 400
GPU	Adreno 330
Baterija	Li-Ion 3000 mAh

6.2. Rezultatai

6.2.1. Duomenų perdavimo greičių diagrama



PAV. 21 Duomenų perdavimo greičių diagrama