Kauno technologijos universitetas

Informatikos fakultetas

Daiktų interneto protokolo saugumo didinimo naudojant vienkartinius slaptažodžius metodas

Baigiamasis magistro projektas

Dominykas Gimžūnas

Projekto autorius

Prof. Algimantas Venčkauskas

Vadovas

Kaunas, 2025

Kauno technologijos universitetas

Informatikos fakultetas

Daiktų interneto protokolo saugumo didinimo naudojant vienkartinius slaptažodžius metodas

Baigiamasis magistro projektas

Informacijos ir Informacinių Technologijų Sauga (6211BX008)

Dominykas Gimžūnas

Projekto autorius

Prof. Algimantas Venčkauskas

Vadovas

Doc. Gedeiminas Činčikas

Recenzentas

Kaunas, 2025

Kauno technologijos universitetas

Informatikos fakultetas

Dominykas Gimžūnas

Daiktų interneto protokolo saugumo didinimo naudojant vienkartinius slaptažodžius metodas

Akademinio sąžiningumo deklaracija

Patvirtinu, kad:

1. baigiamąjį projektą parengiau savarankiškai ir sąžiningai, nepažeisdama(s) kitų asmenų autoriaus ar kitų teisių, laikydamasi(s) Lietuvos Respublikos autorių teisių ir gretutinių teisių įstatymo nuostatų, Kauno technologijos universiteto (toliau – Universitetas) intelektinės nuosavybės valdymo ir perdavimo nuostatų bei Universiteto akademinės etikos kodekse nustatytų etikos reikalavimų;

2. baigiamajame projekte visi pateikti duomenys ir tyrimų rezultatai yra teisingi ir gauti teisėtai, nei viena šio projekto dalis nėra plagijuota nuo jokių spausdintinių ar elektroninių šaltinių, visos baigiamojo projekto tekste pateiktos citatos ir nuorodos yra nurodytos literatūros sąraše;

3. įstatymų nenumatytų piniginių sumų už baigiamąjį projektą ar jo dalis niekam nesu mokėjęs (-usi);

4. suprantu, kad išaiškėjus nesąžiningumo ar kitų asmenų teisių pažeidimo faktui, man bus taikomos akademinės nuobaudos pagal Universitete galiojančią tvarką ir būsiu pašalinta(s) iš Universiteto, o baigiamasis projektas gali būti pateiktas Akademinės etikos ir procedūrų kontrolieriaus tarnybai nagrinėjant galimą akademinės etikos pažeidimą.

Dominykas Gimžūnas

Patvirtinta elektroniniu būdu

Gimžūnas, Dominykas. Daiktų interneto protokolo saugumo didinimo naudojant vienkartinius slaptažodžius metodas. Magistro baigiamasis projektas / vadovas prof. Algimantas Venčkauskas; Kauno technologijos universitetas, Informatikos fakultetas.

Studijų kryptis ir sritis (studijų krypčių grupė): Informacijos ir informacinių technologijų sauga.

Reikšminiai žodžiai: MQTT 5.0 protokolo sauga, vienkartiniai slaptažodžiai, fiziniu būdu neklonuojamos funkcijos, ribotų resursų įrenginiai, daiktų internetas.

Kaunas, 2025. 51 p.

Santrauka

Daiktų interneto sistemose dažnai naudojami labai įvairūs įrenginiai, turintys skirtingus – dažnai itin ribotus kompiuterinius resursus. Dėl šios priežasties daugelyje DI įrenginių atsisakoma naudoti standartines saugumo priemones, tokias kaip TLS protokolas, nes jos gali būti per daug resursų reikalaujančios. Toks saugumo trūkumas paverčia DI aplinkas pažeidžiamomis – kyla grėsmės duomenų nutekėjimui, neteisėtai prieigai prie įrenginių ar sistemų perėmimui.

Šio darbo tikslas – pasiūlyti MQTT protokolo saugos patobulinimą, pritaikytą ribotų resursų įrenginiams. Darbe apžvelgiami dažniausiai naudojami DI komunikacijos protokolai ir jų saugos ypatumai, įvardijamos pagrindinės saugos spragos bei analizuojami esami ir eksperimentiniai sprendimai, siūlomi mokslinėje literatūroje.

Išanalizavus MQTT protokolą, pasirinktas jo naujausias 5.0 standartas, leidžiantis naudoti „AUTH“ kontrolinius paketus. Pasiūlytas sprendimas remiasi kelių žingsnių autentifikacija, naudojant vienkartinius slaptažodžius bei fizinės neklonuojamos funkcijos technologiją, leidžiančią saugoti jautrią informaciją be nuolatinio jautrios informacijos, skirtos generuoti vienkartinius slaptažodžius, saugojimo. Taip pat integruotas lengvas „AES-CBC“ šifravimo algoritmas, pritaikytas naudoti įrenginiuose, kurių resursai riboti.

Prototipo tyrimo metu naudotos trys virtualios mašinos, imituojančios atskirus sistemos komponentus. Atlikta siūlomo metodo analizė – matuotas veikimo laikas, atminties ir tinklo resursų suvartojimas, vertintas skirtingų šifravimo algoritmų poveikis, žinučių dydžių įtaka sistemai. Gauti rezultatai lyginti su įprastu MQTT klientu tiek be šifravimo, tiek su TLS, siekiant įvertinti, kaip efektyviai veikia siūlomas sprendimas ribotų resursų kontekste.

Gimžūnas, Dominykas. A method for increasing the security of the Internet of Things protocol using one-time passwords. Master's Final Degree Project / supervisor prof. Algimantas Venčkauskas; Faculty of Informatics, Kaunas University of Technology.

Study field and area (study field group): Information and Information Technology Security.

Keywords: MQTT 5.0 protocol security measures, one-time passwords, physically unclonable functions, hardware constrained devices, internet of things.

Kaunas, 2025. 51 pages.

Summary

Internet of Things systems are composed of a wide range of devices, many of which operate with extremely limited computing resources. Due to these constraints, such devices often avoid using standard security protocols like TLS, as they are too resource intensive. This lack of security makes IoT environments attractive targets for malicious actors, increasing risks such as data leakage and unauthorized access to devices.

The goal of this work is to improve the security of the MQTT protocol while considering the limitations of resource-constrained devices. The paper begins with an overview of commonly used communication protocols in IoT environments, examining their security features and identifying typical vulnerabilities. Various security mechanisms are compared, including experimental solutions proposed in recent academic research.

A more in-depth analysis is conducted on the MQTT protocol – one of the most widely used in IoT and its latest version, MQTT 5.0, is chosen due to its support for AUTH control packets. The proposed solution introduces a multi-step authentication mechanism using one-time passwords generated with physical unclonable functions, which enhance secure information storage without the need for permanent secret storage on the device. Additionally, a lightweight AES-CBC encryption algorithm is integrated to ensure data confidentiality without overloading system resources.

To evaluate the proposed solution, a prototype was developed using three virtual machines, each simulating a component of the system. The evaluation includes analysis of latency performance, memory usage, and network traffic consumption. The impact of parameters such as encryption algorithm and message size is assessed. Finally, the results are compared against a standard MQTT client, both with and without TLS, to evaluate how effectively the prototype meets the needs of low-resource IoT environments.

Turinys

Paveikslų sąrašas

Lentelių sąrašas

Santrumpų ir terminų sąrašas

Santrumpos:

1. M2M – įrenginys įrenginiui (angl. *Machine to machine*);
2. DI – daiktų internetas;
3. OSI – Atviras sistemų sujungimo modelis (angl. *Open Systems Interconnection*);
4. MQTT – Pranešimų eilės telemetrijos transportavimas (angl. *Message Queue Telemetry Transport*);
5. PUF – Fiziškai neklonuojamos funkcijos (angl. *Physically unclonable function*);
6. TLS – Transporto lygmens sauga (angl. *Transport Layer Security*);
7. ACL – Prieigos teisių sąrašas (angl. *Access control list*);
8. QOS – Paslaugos kokybė (angl. *Quality of Service*);
9. OTP – Vienkartinis slaptažodis (angl. *One time password*);
10. TOTP – Laiko žyme pagrįstas vienkartinis slaptažodis (angl. *Time based one time password*);
11. HMAC – maišos kodu pagrįstas žinučių autentifikacijos kodas (angl. *Hash Based Message Authentication Code*);
12. HOTP – HMAC pagrįstas vienkartinis slaptažodis (angl. *HMAC one time password*);
13. SRAM – statinė laisvos prieigos atmintis (angl. *Static random access memory*);
14. SDK – programinės įrangos kūrimo rinkinys (angl. *Software development kit*);
15. DoS – paslaugų atsisakymas (angl. *Denial of Service*);
16. CRP – užklausų ir atsakymų pora (angl. *Challenge-response pair*);
17. PLC – programuojamas loginis valdiklis (angl. *Programmable logic controller*);
18. MIB – valdymo informacijos dokumentas (angl. *Management Information Base*)

Terminai:

1. **MQTT brokeris –** MQTT protokolo subjektas, užtikrinantis MQTT žinučių ir MQTT klientų valdymą.
2. **MQTT klientas –** prie MQTT brokerio prisijungęs subjektas.
3. **MQTT siuntėjas –** MQTT klientas, siunčiantis duomenis į MQTT temas.
4. **MQTT prenumeratorius –** MQTT klientas kuris gauna žinutes iš MQTT temų.
5. **MQTT tema –** reikšmė, naudojama MQTT žinučių identifikacijai. Visos MQTT žinutės privalo turėti „temą“ pagal kurią brokeris perduos informaciją kitiems klientams

Įvadas

Šiuolaikinėje ir sparčiai besivystančioje daiktų interneto aplinkoje M2M komunikacijos protokolai tapo svarbiomis technologijomis, kurios leidžia įvairiems įrenginiams bendrauti tarpusavyje ir siųsti duomenis internetu. Šios ryšio priemonės sudaro didelę DI ekosistemą, tačiau nepaisant to, kad šios technologijos siūlo didelę naudą, jos taip pat kelia rimtų saugumo rūpesčių. Saugumo problemos, kaip duomenų privatumas, autentiškumas ir prieinamumas tarp sujungtų įrenginių, yra nuolat aktualios DI aplinkoje. DI ekosistema susideda iš įvairaus tipo įrenginių – nuo galingų serverių iki paprastų jutiklių. Viena iš esamų saugumo problemų – komunikacijos sauga įrenginiuose su ribotais kompiuteriniais resursais. Tiriamojo darbo tikslas – pasiūlyti saugos mechanizmą MQTT protokole, naudojant vienkartinių slaptažodžių metodą, pritaikytą negalingiems gaminiams.

Darbo uždaviniai:

1. Išanalizuoti dažniausiai naudojamus DI protokolus;
2. Susipažinti su MQTT protokolo saugos metodais;
3. Nustatyti ribotų resursų įrenginių komunikacijos saugos problemas;
4. Išanalizuoti vienkartinių slaptažodžių apsikeitimo metodus bei galimybę naudoti PUF funkcijas;
5. Pasiūlyti vienkartinių slaptažodžių apsikeitimo metodą MQTT protokolu;
6. Realizuoti prototipą, įvertinti greitaveiką ir resursų sunaudojimą.

1. DI ir M2M sistemų protokolai, saugos problemos ir jų sprendimo būdai

Šiuolaikinėje besivystančioje DI ekosistemoje vyrauja begalė skirtingų protokolų kurių pagalba realizuoti sprendimai leidžiantys rinkti ir valdyti duomenis iš skirtingų įrenginių. Šie protokolai skirti tiek bendram sistemų naudojimui, tiek specializuotoms reikmėms. Šiame skyriuje apžvelgiama DI svarba, dažniausiai naudojami protokolai bei jų saugos problemos.

1.1. DI ir M2M svarba šiandieniniame pasaulyje

M2M yra DI dalis, skirta tiesioginei komunikacijai ir duomenų apsikeitimui tarp įrenginių be žmogaus įsiterpimo. Šie įrenginiai naudoja skirtingus protokolus. Pavyzdžiui, M2M komunikacijoje gali veikti industriniai vandens ar elektros jutikliai, siunčiantys informaciją apie temperatūrą, signalo stiprumą ar jutiklio būseną sistemai, kuri saugo šią informaciją žurnaluose ir stebi informacijos kaitą. Esant nurodytoms sąlygoms sistema gali automatiškai atlikti veiksmus, kaip pakeisti temperatūros lygį ar pranešti sistemos administratoriui apie dienos suvestinę ir kitus įvykius. Šie įrenginiai sudaro didelę tarpusavyje sujungtų įrenginių ekosistemą, veikiančių savarankiškai. DI technologija yra plačiai naudojama išmaniuosiuose miestuose. Ši technologija leidžia realiu laiku stebėti įvairias priemones, pavyzdžiui, gatvių šviestuvus, transportą ir aplinką. Tai padeda gerinti miesto veiklą, efektyvumą ir saugumą. DI sistemos plačiai naudojamos prekybos sektoriuje. Tai leidžia prekybininkams gauti išsamesnę informaciją apie savo klientus ir produkto naudojimą, pagerinti prekių sekimą ir stebėseną, taip padedant optimizuoti prekių tiekimą bei sandėliavimą [1]. Esant saugos sistemų trūkumams piktavaliai gali suklastoti skirtingų jutiklių siunčiamus duomenis. Išmanios agrokultūros priemonės atlieka esminį vaidmenį žemės ūkio sektoriuje. Ši technologija leidžia stebėti ir valdyti žemės ūkio procesus nuotoliniu būdu, panaudojant įvairius jutiklius, duomenų rinkimo sistemas ir automatizavimą. DI suteikia galimybę stebėti dirvožemio sąlygas, augalų sveikatą, oro sąlygas ir kitus svarbius parametrus, padedant efektyviau naudoti resursus, tokius kaip vanduo ar trąšos [1]. Išmaniajame energetikos tinkle DI sistemos leidžia efektyviai valdyti elektros energijos naudojimą, stebėti tinklo būklę, greitai aptikti ir ištaisyti klaidas. Tai padeda optimizuoti energijos tiekimą, sumažinti energijos nuostolius ir pagerinti elektros tinklo patikimumą, prisidedant prie efektyvesnės, tvarios ir patikimos energijos tiekimo sistemos [1].

DI yra glaudžiai susijęs su skaitmeniniais dvyniais. Tai yra vis plačiau naudojama modeliavimo metodika, skirta sutapatinti realius fizinius objektus, produktus, sistemas ar procesus su skaitmenine jų kopija. Tai leidžia atlikti centralizuotą objektų testavimą, stebėseną ir simuliavimą. Pavyzdžiui, DI įrenginys, turintis konfigūruojamus parametrus kaip nuskaitomų duomenų siuntimo spartą arba relės būseną gali šią informaciją sutapatinti su skaitmeniniu šio įrenginio dvyniu. Pakeitus dvynio stadiją DI įrenginys turėtų atitinkamai atnaujinti savo informaciją bei prisitaikyti prie naujų sąlygų. Ši technologija plačiai taikoma anksčiau minėtose srityse kaip gamyba, išmanieji miestai ir kitose. Nors iš ši metodika turi didelį potencialą, tačiau tai sukelia iššūkių kaip skirtingų įrenginių sąveika, protokolų standartizavimas, modelių sudėtingumas priklausomai nuo reikalavimų [2].

Daugumoje šiuolaikinių DI sričių taikomos technologijos yra glaudžiai susijusios su įrenginiais, kurie pasižymi itin ribotais skaičiavimo resursais. Tokie įrenginiai kaip išmanieji jutikliai, nedidelės stebėjimo kameros ar kiti išmanūs gaminiai dažniausiai naudoja mažos galios mikrovaldiklius ar silpnus procesorius, kurie buvo sukurti siekiant užtikrinti minimalų energijos suvartojimą, ilgaamžišką veikimą net esant ribotam energijos tiekimui ir patrauklią kainą. Daugelis šių prietaisų veikia nuo nestabilių ar nepriklausomų energijos šaltinių, tokių kaip saulės baterijos ar

akumuliatoriai. Dėl šios priežasties jiems būtina naudoti programinę įrangą ir tinklo protokolus, kurie ne tik taupo elektrą, bet ir reikalauja kuo mažiau operatyviosios atminties, procesoriaus apkrovos bei laikmenos vietos. Tokiose sąlygose labai svarbus tampa efektyvumo ir funkcionalumo balansas. Pavyzdžiui, plačiai žinoma atvirojo kodo operacinė sistema „OpenWRT“, sukurta maršrutizatoriams ir mažiems tinklo įrenginiams, yra puikus pasirinkimas šiems tikslams. Ji paremta „Linux“ operacine sistema, tačiau išlaiko labai mažą sisteminį pėdsaką, todėl gali būti diegiama į negalingus įrenginius. Standartiškai rekomenduojama naudoti „OpenWRT“ įrenginiuose su bent 16 MB RAM ir 128 MB vidinės atminties (16/128), tačiau su tam tikrais kompromisais ji gali būti įdiegta ir į 4/32 MB įrenginius – nors tai apriboja funkcionalumą, pavyzdžiui, neleidžia naudoti grafinių sąsajų ar reikalauja minimalios paketų konfigūracijos. Tokie sprendimai ypač aktualūs atokiose, nuo infrastruktūros atitrūkusiose vietovėse, kur DI įrenginiai turi veikti autonomiškai ir patikimai, o priežiūros galimybės – ribotos.

1.2. DI ir M2M saugos problemos

Dauguma atakų būdų, naudojamų prieš informacines technologijas, taip pat yra efektyvios arba kartais efektyvesnės nei DI įrenginiai. Kadangi nemaža dalis išmanių įrenginių turi didelius komponentų suvaržymus, kaip negalingi procesoriai ar gaminiai neturintys pastovaus elektros šaltinio, pasirenkama naudoti mažesnio saugumo lygio technologijas. Dėl šios priežasties šie įrenginiai yra ypač pažeidžiami. Remiantis [3], toliau aprašytos dažniausiai pasitaikančios atakos prieš DI sistemas:

- Sena programinės įrangos versija – tai dažniausiai pasitaiko dėl to, jog naudotojas neatnaujina programinės įrangos, tačiau to priežastis neretai, ypač industrijos srityje, pasitaiko dėl sunkaus gaminių fizinio pasiekiamumo. Jei DI įrenginiai įrengti sunkiai pasiekiamose vietose ir nėra būdų nuotoliniu būdu atlikti programinės įrangos atnaujinimo, dažnai jie lieka neatnaujinti ir lengvai pažeidžiami. Įmonėms atnaujinti visus nutolusius gaminius gali neproporcingai daug kainuoti, todėl neretai nusprendžiama dėl to nieko nedaryti;
- Nesaugūs numatyti nustatymai – gaminiai gali turėti nesaugius numatytus nustatymus, pavyzdžiui galimybė prisijungti prie gaminio nuotoliniu būdu, atviri nereikalingi prievadai ar numatytai paliktos įjungtos pažeidžiamos programos. Ši problema lieka neišspręsta, jei įrangą naudojantis asmuo neturi pakankamai techninių žinių;
- Įrenginių valdymo sistemos stoka – sesijos ir autentifikacijos valdymas gali būti neteisingai sukonfigūruoti, tai leidžia piktavaliui apsimesti kitais vartotojais [3].
- „Miego trūkumo“ atakos – šio tipo atakos taikosi į mažos galios įrenginius. Atakos metu įrenginiai apkraunami, pavyzdžiui, nesibaigiančiu darbo ciklu, kurio metu įrenginio baterija išeikvojama greičiau nei nustatyta [1].
- Atakos gaminių įsijungimo proceso metu – šioje atakoje limituotų resursų gaminiai atakuojami įsijungimo proceso metu. Tokie gaminiai neturi įjungtų saugumo įrankių, dėl to jie tampa lengvai pažeidžiami.

Prieš tokias atakas piktavaliai dažnai atlieka šių gaminių aparatinio lygio analizę, kurios metu nustatomi pažeidžiamumai. Piktavaliai gali nusipirkti gaminius, prieš kuriuos bus daromos atakos. Pavyzdžiui, gaminių vidiniai komponentai, kaip atmintis, būna išanalizuoti [4]. Tokiu būdu piktavaliai gali išskleisti programinę ir aparatinę įrangą, bei nustatyti sudedamųjų programų ir bibliotekų versijas, kurios gali būti nustatytos, kaip turinčios pažeidžiamumų.

1.3. DI protokolų stekas

DI sistemos susideda iš įvairiausių tipų įrenginių, skirtų atlikti skirtingas užduotis, todėl egzistuoja aibė protokolų, pritaikytų tam tikram naudojimo atvejui. Šie protokolai egzistuoja ir skirtinguose OSI lygiuose, todėl turi įvairių saugumo priemonių. Šiame skyrelyje apžvelgiami dažniausiai naudojami DI protokolai, bei jų saugumo sprendimai.

MQTT protokolas yra aplikacijos OSI modelio lygio protokolas, kuris pasižymi tuo, jog jis neapkrauna daug kompiuterio resursų lyginant su kitais protokolais. Dėl šios priežasties, dauguma negalingų gaminių naudoja šį protokolą apsikeičiant bendro tipo informacijai. Šis protokolas paremtas skelbimo ir prenumeravimo principais, kada klientas, norintis siųsti informaciją, skelbia žinutes į specialų MQTT brokerį, kurio vaidmuo yra kaip tarpininkas tarp siuntėjo ir prenumeratoriaus. Žinutės skelbiamos tam tikrose apibrėžtose MQTT temose, kurias informacijos gavėjas užsiprenumeruoja. Skirtingai nei HTTP, MQTT protokolas neveikia užklausų ir atsakymų principu, todėl protokolas tinka aplinkose, kur komunikacija nėra pastovi ir gali dažnai nutrūkti. MQTT protokolas buvo sukurtas 1999 metais ir buvo tobulinamas keliais etapais [5]. Šiuo metu dažniausiai naudojamos MQTT versijos yra dvi: versija 3.1, išleista 2010 metais yra plačiausiai naudojama. Versija 5.0, kurios naujausia revizija paskelbta 2019 metais [6], suteikia tiek daugiau funkcijų, tiek, svarbiausia, atgalinį suderinamumą su versija 3.1.1. Versija 5.0 suteikia greitaveikos, patikimumo patobulinimus, kontrolės patobulinimus komunikacijai tarp kliento ir brokerio. Viena iš naujovių yra galimybė išplėsti autentifikavimo procesą naudojant kelių žinučių apsikeitimo principu tarp kliento ir serverio. MQTT 5.0 versija aprašo „AUTH“ paketą, todėl nauji autentifikacijos metodai galimi realizuoti ryšio sudarymo metu. MQTT protokolas plačiai naudojamas telekomunikacijos, gamybos ir energetikos sektoriuose.

MQTT-SN (Sensor Network) yra alternatyvi MQTT protokolo versija turinti daug panašumų. Šis protokolas yra specialiai skirtas bevieliams jutikliams ir kitiems resursų ribotiems įrenginiams. Protokolas skirtas panaudojimo sritims kur visos MQTT protokolo funkcijos nėra reikalingos arba yra per daug apkraunančios įrenginiams. Pagrindiniai skirtumai apima trumpesni temų pavadinimai, trumpesni kontroliniai paketai bei pašalintos „QoS“ ir išliekančių žinučių funkcijos. Šis protokolas yra rečiau pritaikomas lyginant su MQTT, kadangi turi mažiau funkcijų bei yra skirtas tik daliai DI sistemų panaudojimo atvejų. Kadangi protokolas skirtas negalingiems įrenginiams, šis protokolas taip pat nepalaiko jokių pažangių saugumo priemonių duomenų srautui užtikrinti, apart naudotojų autorizacijos metodo, aptarto MQTT protokolo analizės metu. Šiam protokolui galima taikyti TLS, kaip duomenų šifravimo metodą tačiau tai naudojant prieštarauja tikslui taupyti įrenginių kompiuterinius resursus.

CoAP - panašiai kaip MQTT, yra resursų neapkraunantis protokolas, skirtas komunikacijai tarp mažos galios įrenginių. Skirtingai nuo MQTT, šis protokolas veikia panašiu principu kaip HTTP, kur žinutės siunčiamos kaip užklausos, kurios turi turėti atsakymus. Tai reiškia, jog šiuo protokolu abi komunikuojančios pusės bendrauja tiesiogiai, priešingai, nei naudojant MQTT, kur komunikacija vyksta per tarpininką – MQTT brokerį. Kitas skirtumas tarp MQTT yra tai, jog CoAP neturi ryšio palaikymo mechanizmo. Abu protokolai sunaudoja nedidelį kiekį resursų ir duomenų, tačiau MQTT protokolas suteikia daugiau galimybių plėsti DI infrastruktūrą [7]. CoAP gali būti sukonfigūruotas naudoti DTLS protokolą apsaugoti duomenis, taip pat, „PSK“ arba „X.509“ sertifikatus norint autentifikuoti klientus.

SNMP skirtas rinkti ir rūšiuoti informaciją apie valdomus gaminiu IP tinkle. Šiuo protokolu galima ir keisti informaciją jei norima keisti gaminių veikimą. Dažniausiai SNMP protokolą palaiko modemai, maršrutizatoriai, spausdintuvai ir kiti gaminiai. Sistemos klientai norėdami skaityti arba rašyti informaciją į SNMP serverį daro tai nurodydami objektų identifikatoriaus numerį „OID“ kuriuose suskirstyti sistemos parametrai kaip sistemos laikas, tinklo informacija ir kita. SNMP protokolas turi keletą versijų: „v1“, „v2c“ ir „v3“. „v1“ yra seniausia versija, turinti labai ribotas saugumo galimybes – autentiškumas grindžiamas paprastu bendru slaptažodžiu, vadinamu bendruomenės raktu, kuris perduodamas atviru tekstu. „v2c“ pasižymi tam tikrais techniniais patobulinimais, bet saugumo modelis išlieka toks pat nesaugus kaip ir „v1“. „v3“ pristato pažangesnį saugumo modelį, įskaitant vartotojų autentifikavimą, prieigos kontrolę bei galimybę užšifruoti duomenų srautą naudojant TLS arba kitus metodus. Dėl to SNMPv3 laikoma vienintele tinkama versija naudoti saugiuose tinkluose. Nepaisant SNMP galimybių, protokolas turi keletą saugumo problemų:

- Perduodami duomenys atviru tekstu („v1“ ir „v2c“), leidžia potencialiems užpuolikams perimti ar pakeisti informaciją, pasinaudojant MITM atakoms.
- Silpna autentifikacija („v1“, „v2c“) leidžia lengvai atspėti ar brutalios atakos metodu atskleisti bendruomenės raktų reikšmes.
- Nepakankama prieigos kontrolė, jei ACL taisyklės nėra tinkamai sukonfigūruotos, gali leisti nepagrįstą prieigą prie jautrios įrenginio informacijos arba leisti keisti veikimo parametrus.
- Perdėtas pasitikėjimas UDP protokolu, kuris neturi pakartotinio siuntimo mechanizmo ir neužtikrina patikimumo.

Kitas svarbus aspektas yra gamintojų specifinės SNMP įgyvendinimo problemos. Nors SNMP yra standartizuotas protokolas, skirtingi gamintojai dažnai naudoja savo unikalius MIB plėtinius arba netgi pažeidžia SNMP specifikacijas. Tai gali sukelti suderinamumo problemų, kai bandoma valdyti įrenginius iš skirtingų tiekėjų vienu centriniu valdymo įrankiu, pavyzdžiui, kai kurie įrenginiai neteisingai interpretuoja OID struktūras, kiti gali turėti nestandartinius atsakymų formatus, kai kurie gamintojai neįgyvendina pilnos SNMPv3 specifikacijos, arba neleidžia naudoti šifravimo be papildomų licencijų. Dėl šių priežasčių, tinklo administratoriams dažnai tenka atlikti kruopštų suderinamumo testavimą, dokumentacijų peržiūrą ir naudoti gamintojų pateiktus MIB failus, kad užtikrintų sklandų SNMP veikimą mišriose tinklo infrastruktūrose.

„Modbus“ yra kliento ir serverio principu paremtas aplikacijos lygio protokolas, skirtas programuojamuose loginiuose valdikliuose „PLC“. Šis protokolas vienas iš populiariausių industrijos sektoriuje, kadangi jis yra nemokamai prieinamas ir pakankamai lengvai pritaikomas sistemose. Protokolas veikia valdančiojo – pavaldžiojo principu, kur valdantysis įrenginys kaupia informaciją apie esamus klientus arba pavaldinius, bei siunčia komandas atlikti tam tikrus veiksmus. Informacijos perdavimo metu siunčiami „Modbus“ registrai bei jų esanti informacija, serveris nustatomas skaityti tam tikrus registrus, jų kiekį, esantį duomenų tipą, baitų seką ir kitus parametrus, pavyzdžiui 32 bitų skaitmuo. „Modbus“ gali naudoti serijinę komunikaciją, „Ethernet“ arba IP kaip transporto sluoksnį. Pats protokolas neturi jokių saugumo mechanizmų. Jei naudojama serijinė komunikacija, tai yra, klientas ir serveris fiziškai sujungtas komunikacijos laidu, reikalinga ir fiziškai pasiekti šiuos gaminius norint sužinoti kokie duomenys apsikeičiami. Kadangi „Modbus“ naudoja kitus protokolus kaip IP, piktavaliai gali naudoti šių protokolų saugumo spragas [4].

M-Bus yra fizinio ir link lygmens protokolas, skirtas komunikacijai su industriniais skaitikliais, kaip vandens, dujų ir elektros. Šis protokolas, panašiai kaip „Modbus“, paremtas valdančiojo – pavaldžiojo komunikacijos principu, tinkantis negalingiems įrenginiams. M-Bus protokolas neapibrėžia jokių saugumo priemonių, todėl komunikacijos saugumo priemonės dažnai realizuojamos aukštesniuose OSI lygmens protokoluose, pavyzdžiui, TLS.

1.4. MQTT protokolo saugos problemos ir sprendimai

Šiame skyrelyje apžvelgiamos, kokios saugos problemos iškyla naudojant MQTT protokolą, bei kokios priemonės dažniausiai naudojamos sprendžiant šias problemas. Pradžioje apžvelgiama kokias saugos priemones suteikia pats MQTT protokolas ir kokie MQTT nustatymai įtakoja saugų protokolo naudojimą. Vėliau apžvelgiama kokios papildomos priemonės, nesusijusios su pačiu MQTT protokolus taikomos dažniausiai, bei kokią įtaką tai daro DI sistemose. Galiausiai atliekama analizė saugos metodai analizuojami kituose moksliniuose straipsniuose.

1.4.1. MQTT protokolo suteikiami įrankiai saugumui pagerinti

MQTT protokolas veikia kliento ir brokerio principu. Protokolas aprašo specialias temas su grotelių „#“ arba pliuso „+” simboliais. Naudojant šiuos simbolius klientai gali gauti visas žinutes, kurių temų pavadinimai sutampa su specialiaisiais simboliais. Pavyzdžiui klausytojas, klausantis temos „jutiklis_1/#“ gaus žinutes, skirtas temoms „jutiklis_1/temperatūra“ „jutiklis_1/tinklas/tx” bei kitas. Jei klientas klausysis žinučių „#” temoje, jis gaus visas žinutes einančias į MQTT brokerį. Tai gali sukelti saugumo problemą, jei prie MQTT brokerio prijungta didelis kiekis įrenginių kurie siunčia savo žinutes – visi klientai gali gauti vienas kito žinutes. Tai gali būti traktuojama kaip nesaugi operacija jei norima riboti kokia informacija prieinama tam tikriems klientams. Šiai situacijai suvaldyti MQTT brokeris turi būti sukonfigūruotas neleisti klientams naudotis šių specialių simbolių gaunant žinutes.

MQTT protokolas aprašo papildomą autentifikacijos metodą, kurį gali naudoti klientai. Šis autentifikacijos metodas yra paremtas naudotojo vardo ir slaptažodžio naudojimu. MQTT brokeriui nurodomas slaptažodžių failas kuris susideda iš leistinų naudotojų ir jų slaptažodžiais, kuriems pritaikyta maišos funkcija. MQTT klientai, sudarydami ryšio kanalą su brokeriu, gali siųsti naudotojo vardą ir neprivalomą slaptažodį, o brokeris patikrina ar naudotojas yra autorizuotas ir slaptažodis sutampa. Papildomai, klientai gali nurodyti savo identifikacijos numerį, pagal kurį brokeris atskiria skirtingus klientus. Tai reiškia, jog daug klientų gali sudaryti ryšį su tokiu pačiu naudotojo vardu. Jei klientas nenurodo savo identifikacijos numerio, jis yra automatiškai sugeneruotas, tačiau jei nenurodomas naudotojo vardas, MQTT klientas skaitomas kaip esantis anoniminis naudotojas. Visa informacija siunčiama ryšio kanalais nėra šifruojama, todėl visi tinkle esantys naudotojai gali lengvai perskaityti kokia informacija siunčiama autentifikacijos metu.

Remiantis [6], MQTT „CONNECT“ kontrolinis paketas palaiko paprastą ryšio sudarymo autentifikaciją, naudojant naudotojo vardą ir slaptažodį. Patobulintas autentifikacijos metodas naudoja šiuos laukus realizuojant iššūkio ir atsako tipo autentifikaciją. Šis metodas naudoja papildomą „AUTH“ paketo tipą. Klientas ir brokeris naudoja šį paketą autentifikacijos metu. Klientas išsiunčia „CONNECT“ paketą su nurodytu autentifikacijos metodu. Jei MQTT brokeris palaiko šį autentifikacijos metodą, jis gali reikalauti papildomos informacijos iš kliento arba jei autentifikacijos metodas reikalauja brokeriui nusiųsti autentifikacijos duomenis klientui, „AUTH“ paketai yra

siunčiami tarp MQTT brokerio ir kliento. Klientas ir MQTT brokeris gali apsikeisti „AUTH“ paketais tiek kartų kiek jiems reikia. Tai reiškia jog, kadangi MQTT protokolas neprivalo žinoti jokios informacijos apie autentifikacijos metodo detales, autentifikacijos metodai kaip OAuth2 ir kiti gali būti realizuoti ryšio sudarymo metu tarp kliento ir MQTT brokerio. Tai suteikia papildomą priemonę pagerinti saugumo lygį autentifikacijos metu.

MQTT brokeriai gali būti atakuojami ir kitais būdais. Viena iš lengvai įgyvendinamų – DoS tipo atakos. Jei piktavalis gali siųsti žinutes brokeriui, jie lengvai gali apkrauti brokerį siųsdami didelio kiekio žinutes. Maksimalus vienos žinutės dydis, neįskaitant temos, yra 256 megabaitai, remiantis MQTT protokolo standartu [6]. Temos dydis yra limituojamas 65536 baitais. DoS tipo atakoms brokeriai paprastai sukonfigūruojami priimti mažesnio dydžio žinutes, bei sukonfigūruojami nebepriiminėti naujų žinučių, jei užpildytas laukiamų žinučių kiekis. Šios priemonės realizuojamos MQTT brokerių programose, kurie stebi paketų informaciją. Norint apsisaugoti nuo brutalių atakų, pavyzdžiui, jei piktavalis nori atspėti MQTT kliento naudotoją slaptažodį siunčiant daug paketų, paprastai tai įgyvendinama ugniasienės pagalba. Ugniasienė gali būti sukonfigūruota nepraleisti paketų kurie nukreipti į MQTT brokerio prievadus po nurodyto kiekio neatspėtų slaptažodžių.

1.4.2. MQTT protokolo duomenų srauto apsaugojimas

Pagal numatymą, MQTT protokolas nenaudoja jokių šifravimo priemonių duomenų saugumui užtikrinti – visi duomenys siunčiami atviru tekstu. Norint išlaikyti duomenų autentiškumą, ir konfidencialumą, egzistuoja keli metodai. Vienas iš dažniausiai naudojamų – TLS kriptografinis protokolas. Jis paremtas kriptografinių protokolų naudojant sertifikatus, veikiantis atvaizdavimo OSI lygmeniu. Klientui ir serveriui norint sudaryti ryšio kanalą, abi pusės susitaria naudoti nurodytą TLS protokolą ir atlieka rankos paspaudimo procedūrą. Šio žingsnio metu susitariama, kokie kriptografiniai algoritmai bus naudojami, apsikeičiami sertifikatai, kuriuose pateikiama informacija apie abi puses ir sukuriami sesijos raktai, kurie bus naudojami duomenims šifruoti. Dauguma populiarių MQTT brokerių, kaip „HiveMQ“ ir „Mosquitto“, gali naudoti TLS protokolą vietoje nešifruojamo TCP/IP protokolo, kaip duomenų kanalą. Pagal numatymą, naudojant TCP yra sudaromas ryšio kanalas prievadu 1883, o TLS naudojamas prievadas 8883. TLS protokolas kartu su MQTT yra plačiai pritaikytas debesijos sistemose. Praktiškai visos debesijos platformos, pritaikytos M2M komunikacijai naudoja MQTT su TLS protokolu. Platformos kaip „Azure IoT Hub“, „AWS IoT Core“, „Teltonika Remote Management System“ paremtos šiais protokolais.

Viena didžiausių problemų naudojant TLS – stipriai padidinti greitaveikos reikalavimai, norint naudoti TLS šifravimą. Kadangi abi komunikuojančios pusės turi atlikti sudėtingus kriptografinius skaičiavimus, gaminiams, turintiems ribotus kompiuterinius išteklius, kyla problema: stipriai iškyla energijos reikalavimai, nukenčia įrenginio greitaveika bei padidėja sunaudotų duomenų kiekis [8]. Remiantis [9] ir [10], nustatyta, kad TLS protokolo naudojimas vietoj jokio duomenų šifravimo padidina energijos sunaudojimo lygį iki dviejų kartų priklausant, koks naudojamas QoS lygis. Ši problema ypač aktuali, jei įrenginio energijos šaltinis nėra pastovus, o teikiamas iš įkraunamų baterijų. Kai kurie MQTT brokeriai kaip „Mosquitto“ gali komunikuoti TLS protokolu naudodami „PSK“ šifravimo metodą. Šiuo metodu abi komunikuojančios pusės atlieka rankos paspaudimo procedūrą, naudodami iš anksto tarpusavyje pasidalintus raktus. Šis metodas, lyginant su sertifikatų naudojimu, gali būti paprasčiau įgyvendinamas jei reikalinga administruoti daug įrenginių, tačiau saugumo lygis gali nukentėti.

Dauguma MQTT programų ir bibliotekų palaiko papildomą funkcionalumą sudaryti MQTT komunikaciją naudojant WebSocket protokolą. Šis metodas dažnai naudojamas jei reikalinga gauti MQTT duomenis tiesiai į interneto naršyklę. Šiuo metodu duomenys siunčiami panašiai kaip HTTP/HTTPS protokolais, todėl galimas ir duomenų šifravimas. Duomenų šifravimui naudojamas TLS protokolas, todėl tokios pačios problemos išlieka negalingiems įrenginiams. Kadangi šis metodas skirtas siųsti duomenis į naršykles, jis nėra tinkamas M2M komunikacijai.

Kitas duomenų srauto šifravimo metodas yra sudarant VPT ryšį tarp kliento ir brokerio. Šis metodas kartais taikomas jei dėl tam tikrų priežasčių jei pačios MQTT brokerio arba kliento programos negali palaikyti TLS protokolo. Šiuo būdu duomenys, keliaujantys tinkle, bus apsaugoti, tačiau šis sprendimas yra netinkamas M2M komunikacijai. Tai yra, nes abi komunikuojančios pusės, kaip ir naudojant TLS protokolu, privalo atlikti sudėtingus kriptografinius skaičiavimus bei palaikyti VPN ryšiui sudaryti reikalingas programas.

Kitas būdas duomenų šifravimui yra MQTT žinučių šifravimas vietoj viso MQTT paketo. Naudojant šį metodą nebereikia naudoti TLS protokolo, todėl MQTT komunikacijos paketai nekinta. Šiuo būdu galima sukurti naujus šifravimo metodus, pavyzdžiui, įrenginiai su ypač ribotais kompiuteriniais ištekliais gali naudoti žinučių šifravimą tik tada jei tai reikalinga. Taikant tik žinučių šifravimą, kompiuterinių resursų sunaudojimas taip pat gali sumažėti, lyginant su TLS protokolu, kadangi nebereikalinga šifruoti viso paketo. Šifravimo algoritmas taip pat gali būti specialiai parinktas paprastesnis, norint taupyti kompiuterinius resursus. Viena pagrindinių problemų taikant šį metodą yra paplikymo problema. Visi MQTT protokolo dalyviai turi žinoti kokiu metodu vyks ši komunikacija, todėl reikalinga turėti šiems metodams palaikomus MQTT klientus ir brokerius. [11] pateikiamas metodas apsaugoti MQTT protokolą šifruojant tik pačias žinutes. Šaltinyje taikomas šifravimas remiasi kortelių skaitytuvais, kuriais atliekama autentifikacija ir autorizacija „CONNECT“ ir „CONNACK“ kontroliniais paketais. Šis metodas nėra visiškai tinkamas naudoti M2M komunikacijai kadangi reikalinga papildoma aparatinė įranga bei yra galimybė pagerinti protokolo veikimą pilnai išnaudojant naujas MQTT 5.0 standarto funkcijas. [12] taip pat naudoja tik MQTT žinučių šifravimą bei taiko metodiką, pagal kurią bandoma nukreipti kompiuteriui sudėtingus veiksmus atlikti brokeryje, kuris turės didesnius kompiuterinius išteklius. Šiuo būdu stengiamasi pagerinti klientų greitaveiką ir sumažinti jų resursų sunaudojimą naudojant duomenų šifravimą.

Kai kurie sprendimai saugojant duomenų srautą bei mažinant resursų sunaudojimą negalingiems gaminiams padalina ir atskiria vietas, kur duomenų šifravimas bus taikomas. Pavyzdžiui, paprastas sprendimas yra įtraukti naują tarpinį MQTT brokerį tarp abiejų komunikuojančiųjų. Tokiu sprendimu visi negalingi įrenginiai siunčia duomenis be šifravimo tarpiniam brokeriui, kuris perduoda visas einančias žinutes gavėjui, naudojant TLS. Šiuo pagrindu galima dar labiau suefektyvinti negalingų įrenginių resursų sunaudojimą, pritaikant MQTT-SN protokolą tik tarp šių įrenginių ir tarpinio brokerio, kuris pavers šias žinutes į MQTT tinkamą formatą. Šis metodas, nors ir nesudėtingai įgyvendinamas, tačiau nesuteikia nei šifravimo, nei saugaus autentifikavimo metodo negalingiems gaminiams. [13] remiasi minėtais principais ir siūlo pagerintą saugos architektūrą MQTT-SN tinklui. Šaltinyje pateikiamas sprendimas naudoja MQTT ACL funkciją, valdant, kokie klientai kuriose temose gali skaityti arba rašyti. Temos taip pat yra valdomos taip, jog tik autorizuoti naudotojai gali skaityti, o perduodami duomenys yra užšifruoti, todėl jokie tarpiniai brokeriai negali jų skaityti. Šis sprendimas reikalauja MQTT protokolo modifikavimo, užtikrinti tarpusavio autentifikaciją, prieigos kontrolę, kontrolinių žinučių saugumą ir kitas saugos funkcijas, todėl šis sprendimas gali nevisiškai veikti su įprastu MQTT protokolo standartu.

1.4.3. Kiti MQTT saugos metodai

Specialiųjų temų simbolių užblokavimas suteikia tik dalinę apsaugą, norint užtikrinti kokie klientai galės skaityti ir siųsti į kokias temas. Klientai gali apeiti šį tikrinimą užprenumeruojant didelį kiekį skirtingų temų arba atlikti kryptingus spėliojimus, kokiose temose bus siunčiama informacija, norint skaityti jiems neskirtą informaciją. Šiai situacijai išspręsti egzistuoja dinaminis saugos, kuris suteikia rolėmis pagrįstą autentifikavimo ir prieigos kontrolės funkcijas. Šis įrankis susieja MQTT protokolo kontrolinius paketus, kaip „CONNECT“, su įskiepio klientais, grupėmis ir rolėmis. Pagal šią informaciją dinaminis saugos įskiepis leidžia sistemos administratoriui kurti prieigos kontrolės sąrašus, kurių pagrindu galima valdyti prieigą prie temų. Šie sąrašai leidžia valdyti kokie MQTT klientai gali tiek siųsti, tiek klausytis nurodytų temų. Norint naudotis šiuo įrankiu reikalinga naudoti tai palaikantį MQTT brokerį, vienas iš populiariausių ir šią funkciją palaikančių – „Mosquitto“ MQTT brokeris. „Mosquitto“ MQTT brokeriui taip pat yra sukurti atviro kodo plėtiniai, kurie suteikia kitų autentifikacijos būdų, pavyzdžiui naudojant duomenų bazes kaip „SQLite3” ir „MongoDb“, naudojant „JWT“ ir kitais būdais. Tai išplečia dinaminio saugos įskiepio galimybes. Šie plėtiniai ne visada yra aktyviai prižiūrimi, pavyzdžiui „mosquitto-auth-plug“ projektas yra nebeatnaujintas nuo 2019 metų, todėl gali turėti saugos spragų kaip sena „Mosquitto“ bibliotekos versija.

Atsižvelgiant į kitus eksperimentinius metodus ir mokslinius straipsnius, susijusius su DI įrenginių saugos protokolų patobulinimais, jau yra pasiūlyta metodų kurie taip pat naudoja PUF metodus kartu su MQTT protokolu. [14] šaltinyje siūlomas metodas naudoja PUF funkcijas, norint suteikti tarpusavio autentifikaciją tarp serverio ir kliento. Vienas iš siūlomo metodo reikalavimų yra, jog serveris privalo žinoti tam tikrus kliento, norinčio autentifikuotis, PUF užklausų atsakymus. Šio reikalavimo užtikrinimui bei įgyvendinimui nėra apibrėžiamos realizacijos sąlygos ar pateikiami pavyzdžiai, o pateiktame prototipe nėra laikomasi MQTT komunikacijos standarto atliekant siūlomą metodą – papildoma raktų apsikeitimo informacija siunčiama prieš „CONNECT“ kontrolinį paketą, siunčiant TCP/IP srautą tiesiogiai į serverį. [15] pateikiamas metodas pagerinti MQTT saugą naudojant „AugPAKE“ protokolą kaip papildomą saugos lygmenį, norint apsaugoti nurodytas komunikacijos temas. Nors ir šis metodas prideda egzistuojantį ir išsamiai aprašytą saugos mechanizmą prie MQTT protokolo, tokiu būdu visumoje išplečiant MQTT protokolo galimybes, tačiau tai pažeidžia MQTT protokolo standartą, kadangi atsiranda papildomas komunikacijos kanalas tarp kliento ir serverio, kuris atliekamas nesilaikant nei MQTT 3.1.1 nei 5.0 standarto versijų. Šaltinyje [16] pateikiamas saugos mechanizmas, naudojant vienkartinius slaptažodžius. Šis metodas, lyginant su kitais anksčiau minėtais, yra pranašesnis vien dėl to, jog visa komunikacija tarp kliento ir serverio vyksta tik MQTT protokolo kontroliniais paketais. Visa reikalinga informacija, susijusi su vienkartiniais slaptažodžiais, yra siunčiama serveriui, naudojant „CONNECT“ kontrolinius paketus, papildomai, brokeris gali valdyti klientams pasiekiamas komunikacijos temas, kurios yra prisegtos prie kiekvieno kliento informacijos atskiroje duomenų bazėje. Šis metodas suteikia tik HOTP mechanizmą, bei, kadangi papildoma informacija siunčiama tik „CONNECT“ kontroliniais paketais, tai reiškia jog klientas turi tik vieną progą išsiųsti savo visą reikalingą papildomą informaciją serveriui. Tai reiškia, jog nėra galimybės serveriui ir klientui apsikeisti papildoma informacija daugiau kaip vieną kartą laikantis MQTT protokolo specifikacija, o tai reikšmingai apmažina protokolo įgyvendinimo galimybes. Šaltinyje [17] pateikiamas metodas pagerinti protokolo saugą naudojant „Eclipse Arrowhead“ karkasą kartu su JWT žetonais. Šis metodas, kitaip nei visi anksčiau minėti, naudoja atnaujintą MQTT 5.0 protokolo versiją, bei užtikrina jog visa papildomai siunčiama informacija susijusi su siūlomu metodu atitinka protokolo standartus. Siūlomame metode papildoma

saugos informacija serveriui siunčiama, naudojant standarte aprašytais papildomais naudotojo parametrais laukais, kuriuose prisegama informacija apie JWT žetonus ir kita reikalinga informacija apie klientus. Vienas iš nagrinėtų darbų [18] pagerina protokolo saugą naudojant PUF. Vienas iš autentifikacijos mechanizmų yra susijęs su užklausų atsakymų apsikeitimu tarp kliento ir brokerio, tačiau darbe nebuvo plačiai apibūdinta, kaip siūloma sistema atitinka MQTT protokolo standartus. Darbe taip pat yra pristatomos privačios komunikacijos temos, kas neatitinka standartų, bei naudoja informacijos užslėpimo principus kaip priemonę gerinti bendrą sistemos saugą.

1 lentelė. Skirtingų MQTT protokolo saugos sprendimų palyginimas

Funkcija / **Sprendimas**	**Autentifikacijos informacijos apsikeitimas daugiau nei vienu paketu**	**Naudotojo-slaptažodžio autentifikacija**	**MQTT žinučių turinio šifravimas**	**MQTT kontrolinio paketo šifravimas**	**Temų prieigos teisių valdymas**	**MQTT 5.0 standarto atitikimas**
Įprastas MQTT klientas	-	+	-	-	-	+
Įprastas MQTT klientas su TLS	+	+	-	+	-	+
„Mosquitto“ patobulintos saugos įskiepis	-	+	-	-	+	+
[16]	-	+	-	-	+	-
[17]	-	+	-	-	+	+

1 lentelėje palyginamos skirtingos su sauga susijusios funkcijos, kurios yra įgyvendintos MQTT sprendimuose. Pasirinktos funkcijos atvaizduoja aktualius saugos aspektus DI aplinkoje. Visi sprendimai, išskyrus MQTT klientą su TLS, nešifruoja viso MQTT protokolo paketo. Tai yra dėl to, jog TLS protokolo naudojimas yra galimas visuose sprendimuose, kaip papildoma ir neprivaloma funkcija. Tai reiškia, jog MQTT žinutės šifravimo funkcija dažniausiai nėra reikalinga, kadangi panašų rezultatą galima pasiekti kitais, standartiniais būdais. Kadangi TLS protokolas gali būti laikomas netinkamu DI aplinkoje, dėl to, reikalinga naudoti alternatyvias funkcijas, norint įgyvendinti duomenų šifravimą. Autentifikacijos informacijos apsikeitimas tarp serverio ir kliento dažniausiai vyksta vienu kontroliniu paketu. Pavyzdžiui, naudojant įprastus autentifikavimo metodus pakanka, jog klientas vieną kartą pateiktų naudotojo vardą ir slaptažodį, kad autentifikacija būtų įvykdyta. Kitaip tariant, dauguma esamų sprendimų remiasi paprasta vieno žingsnio autentifikacija, kur kliento tapatybė patvirtinama vos tik gavus jo prisijungimo duomenis. Tačiau MQTT standartas nenumato standartinių autentifikacijos mechanizmų, kurie veiktų keliais etapais. Protokolai kaip TLS gali pasiūlyti sudėtingesnį autentifikacijos procesą su rankos paspaudimo mechanizmais, tačiau jie veikia virš MQTT protokolo sluoksnio ir nėra tiesiogiai susiję su pačiu MQTT protokolu. Naudotojo slaptažodžio autentifikavimo mechanizmas yra papildoma ir neprivaloma saugos funkcija, pavyzdžiui, esant poreikiui, galima pasirinkti naudoti šį autentifikavimo mechanizmą kartu su TLS protokolu – šios funkcijos nepriklauso viena nuo kitos. MQTT standartas neaprašo temų prieigos teisių valdymo reikalavimų, tik gerąsias praktikas [6]. Kadangi egzistuoja specialios MQTT temos, kurios leidžia atlikti filtravimą kaip „#”, atsiranda galimybė piktavaliams lengvai stebėti MQTT brokerio srautą. Taip pat, kiekvienas MQTT brokeris turi specialias vidines temas, skirtas statistikų stebėsenai, kaip prisijungusių klientų kiekis, MQTT brokerio būsena ir kitą svarbią informaciją. Ši

informacija, randama „$SYS/broker/…" paprastai neturėtų būti aktuali MQTT klientams, tačiau jie gali šias temas užsiprenumeruoti ir gauti ateinančias žinutes. Ši laisva temų prieigų savybė gali būti laikoma nesaugia, ypač kai sistema turi atitikti griežtesnius saugos reikalavimus – pavyzdžiui, industrinėse ar privatumo reikalaujančiose DI sistemose. Todėl atsiranda būtinybė riboti ir valdyti klientams prieinamas temas, atsižvelgiant į jų rolę, paskirtį ar autentifikacijos lygį. Tokia funkcija realizuota kai kuriuose modifikuotuose MQTT brokeriuose, pavyzdžiui, naudojant „Mosquitto" patobulintos saugos įskiepį. Šis įskiepis leidžia administratoriams susieti MQTT klientus su naudotojais ir grupėmis, kurioms galima apibrėžti skirtingas prieigos kontrolės taisykles. Tokiu būdu galima tiksliai apibrėžti, kokias temas konkretus klientas gali prenumeruoti ar į kurias temas gali publikuoti žinutes. Pavyzdžiui, viena grupė gali turėti prieigą tik prie jutiklių temų, o kita – tik prie sisteminių įrašų ar valdymo komandų. Dinaminio saugumo mechanizmas taip pat palaiko realaus laiko konfigūravimą – administratoriui nereikia perkrauti MQTT brokerio keičiant naudotojų teises ar kuriant naujas taisykles, todėl šis sprendimas puikiai tinka dinamiškai kintančioms sistemoms.

1.5. Vienkartinių slaptažodžių pritaikymas

Vienkartiniai slaptažodžiai plačiai naudojami, atliekant kelių faktorių autentifikaciją, pavyzdžiui, jungiantis prie naudotojo paskyrų. Vienkartiniai slaptažodžiai suteikia papildomą saugumo lygį, kadangi nesankcionuotas subjektas, gavęs vienkartinį slaptažodį, negali apsimesti kitu subjektu, o pats slaptažodis neatskleidžia jokios informacijos apie autentifikuojamą subjektą. Šiame skyriuje bus apžvelgta, kokiais būdais realizuoti vienkartinių slaptažodžių mechanizmai ir kokie metodai dažniausiai pritaikomi šiuolaikinėse sistemose.

Maišos funkcijos panaudojimas yra pirmasis vienkartinių slaptažodžių panaudojimo metodas. Šiuo metodu autentifikuojantieji atsiunčia sąrašą slaptažodžių, kuriems buvo pritaikyta maišos funkcija. Kiekvienos prisijungimo sesijos metu skirtingi slaptažodžiai siunčiami serveriui. Remiantis [19], pagrindinės šio metodo problemos yra tai, jog visi slaptažodžiai saugomi kliento kompiuteryje ir slaptažodžių kiekis yra limituotas, todėl gali iškilti situacija, kur klientas ir serveris privalo iš naujo susitarti, kokie slaptažodžiai bus naudojami.

HOTP metodas yra paremtas HMAC funkcija. Klientas ir serveris turi bendrą slaptą raktą ir skaitiklį, su kuriais vienkartinis slaptažodis sugeneruojamas naudojant HMAC funkciją. Kiekvieną kartą, kai autentifikacija yra reikalinga, klientas padidina skaitiklio reikšmę. Serveris patvirtina vienkartinio slaptažodžio tinkamumą, sugeneruodamas naują slaptažodį ir palygindamas ar jie sutampa. Naudojantis šiuo metodu, vienkartinių slaptažodžių kiekis nėra ribotas. Remiantis [19], šis metodas nėra tobulas, nes vienkartinių slaptažodžių galiojimo laikas nėra ribojamas, tai gali sukelti saugumo spragų. Kita problema yra sinchronizacija tarp kliento ir serverio – skaitiklio reikšmė tarp kliento ir serverio gali tapti skirtinga ir autentifikuotis nepavyks.

TOTP yra HOTP metodo patobulinimas. Pagrindinis skirtumas yra esamo laiko panaudojimas vienkartinio slaptažodžio generavimo metu vietoj serverio ir kliento skaitliuko reikšmės. Tai išsprendžia sinchronizacijos problemas ir pagerina saugumą – jei piktavalis gaus vienkartinį slaptažodį iš HOTP metodo, jį galimai galės panaudoti, tačiau TOTP metodu slaptažodžio galiojimo laikas gali baigtis iki kol piktavalis jį panaudos.

Tiek TOTP, tiek HOTP metodai yra dažniausiai naudojami šiuolaikinėse sistemose realizuojant vienkartinių slaptažodžių funkcionalumą, tačiau egzistuoja ir kitos alternatyvos. [20] analizuoja blokų

grandinių technologiją, realizuojant vienkartinių slaptažodžių metodą. Šis sprendimas pritaikytas mažo galingumo gaminiams, kurie nori būti MQTT klientais be TLS protokolo naudojimo. Šiuo metodu naudotojai asocijuoja savo identitetą su jų „Ethereum“ adresu, paskelbdami savo „Ethereum“ viešą raktą ir jo signatūrą MQTT brokeriui.

Norint piktavaliui sužinoti, koks vienkartinis slaptažodis bus naudojamas neįsilaužiant į sistemą, pagal [19], statistiškai geriausią šansą turės spėliojant. Remiantis [21], buvo atliktas vienkartinių slaptažodžių saugos tyrimas, kurio metu buvo pastebėta, jog teoriškai piktavalis gali atlikti spėjimus su pakankamai nemažu atspėjimo šansu, jei vienkartinių slaptažodžių generavimo algoritmas nėra pakankamai sudėtingas.

1.6. PUF

PUF yra fizinės funkcijos, kurioms gavus įvestis, gaunamas unikalus fizinis piršto antspaudas, naudojamas kaip unikalus identifikatorius. Pagal [22], funkcijų unikalumas kyla nuo fizinių objektų, pavyzdžiui, puslaidininkių mikroskopinio lygio skirtumai, atsirandantys gamybos proceso metu. Keliama prielaida, jog labai sunku arba beveik neįmanoma klonuoti objekto PUF. Tai reiškia, jog PUF naudojama kaip saugos priemonė autentifikacijos metu ir kriptografiniuose algoritmuose, kur unikalumas yra svarbus aspektas. Šiame skyriuje apžvelgiama į kokias kategorijas skirstomos PUF, kaip jos naudojamos saugos kontekste ir kokios rizikos gali kilti naudojant šią technologiją.

PUF dažnai skirstomi į stiprius ir silpnus tipus. Stiprūs PUF pasižymi aukštu atsitiktinumu, duodami atsaką ir dėl galimybės stipriai skirtis gamybos proceso metu yra sunkiai atkartojami jei piktavaliai bando apsimetinėti šia sistema. Norint turėti stiprų PUF, sudėtingėja gamybinis procesas. Priešingai, silpni PUF neužtikrina aukšto unikalumo lygio ir yra lengviau klonuojami, tačiau gamybinis procesas yra paprastesnis. Vienas iš PUF realizacijos pavyzdžių yra SRAM, kuris naudoja statinės operatyvios atminties ląstelių skirtumus, kuriant unikalius atsakus. Autentifikacijos metodai pasiūlyti [23] ir [22] naudoja SRAM PUF, tinkamus DI įrenginiams.

1.7. Programinės įrangos analizė

Kadangi MQTT protokolas yra atvirasis standartas, egzistuoja aibė skirtingų programų kuriomis galima sukurti tiek brokerius tiek klientus. Taip pat, egzistuoja daug programinių bibliotekų, kurių pagalba integruojamas MQTT palaikymas į naujas programas. Šios bibliotekos yra pritaikytos skirtingoms programavimo kalboms, pavyzdžiui, „.NET“, „Python“, „Java“, „C“ ir kitoms. Programinės kalbos pasirinkimas turi didelę reikšmę realizuojant geresnę MQTT protokolo saugą, ypač kai ši programinė įranga veiks negalinguose įrenginiuose. Papildomai, svarbu nuspręsti kokią MQTT programinę biblioteką pasirinkti kadangi ne visos bibliotekos palaiko visas MQTT protokolo funkcijas kaip pastebėta [24]. Šaltinyje atlikta analizė kurioje lyginamos populiariausios bibliotekos, jų funkcijos, greitaveika ir kiti parametrai. Pastebėta, jog ne visos bibliotekos palaiko MQTT 5.0 standartą. Tai reiškia, jog šiose bibliotekose nebus įmanoma naudoti saugos funkcijas kaip „AUTH“ kontroliniais paketais, todėl norint realizuoti naujus autentifikacijos metodus, reikės modifikuoti patį MQTT protokolą. Pagal esamus darbo reikalavimus, labiausiai tinkama programinė biblioteka yra „Mosquitto“. Ši biblioteka parašyta „C“ programavimo kalba, todėl yra tinkanti naudoti negalinguose įrenginiuose dėl palyginus nedidelio resursų sunaudojimo lyginant su kitoms programavimo kalboms. Taip pat, biblioteka palaiko visas pagrindines MQTT versijas (3.1, 3.1.1 ir 5.0). Galiausiai, „Mosquitto“ plačiai naudojama biblioteka skirtingose sistemose. Viena iš plačiai naudojamų – „OpenWRT“ operacinė sistema, skirta įterptinėms sistemoms, tinklo ir kitiems įrenginiams.

„OpenWRT“ projekte „Mosquitto“ biblioteka ir ją palaikančios programos yra parinktos realizuojant MQTT palaikymą.

1.8. Analizės išvados

1. DI sistemoms nėra vieno universalaus sprendimo – dėl didelės įrenginių įvairovės, naudojamų protokolų ir technologijų, kiekviena architektūra turi būti individualiai pritaikyta pagal turimus resursus ir keliamus funkcinius bei saugumo reikalavimus.
2. MQTT yra plačiai naudojamas DI komunikacijos protokolas, tačiau jo saugumas dažniausiai užtikrinamas naudojant TLS, kuris gali būti per daug apkraunantis silpnesniems įrenginiams. Todėl būtina ieškoti alternatyvių, lengvesnių saugumo sprendimų, tinkančių mažai resursų turintiems gaminiams.
3. MQTT 5.0 standarto „AUTH“ paketo funkcionalumas leidžia įgyvendinti lankstesnius autentifikacijos metodus, kurie gali būti pritaikyti konkrečiai sistemai, įskaitant žingsninę autentifikaciją ar papildomas tapatybės patikros priemones.
4. PUF technologija suteikia galimybę generuoti unikalius įrenginių identitetus, kurie gali būti panaudoti saugesniems autentifikacijos mechanizmams kurti. Visgi jos naudojimui būtina turėti tam tinkamą aparatinę įrangą bei įvertinti realų saugumo lygį, kurį ši technologija gali užtikrinti konkrečiame įrenginyje.

2. DI protokolo saugaus komunikavimo architektūra

Šiame skyriuje aprašoma siūlomo sprendimo vizija, pateikiami architektūros pavyzdžiai, aprašomi sistemos objektai, jų atliekamos rolės. Toliau aprašoma kokie pakeitimai atlikti MQTT protokolo lygmenyje kad būtų pilnai įgyvendinta autentifikacija vienkartiniais slaptažodžiais bei pridėtas papildomas duomenų šifravimas.

2.1. MQTT protokolo su vienkartiniais slaptažodžiais komunikacijos metodo sprendžiama problema

Siūlomas sprendimas skirtas spręsti problemas, kurios vyrauja DI sistemose, su ribotų išteklių įrenginiais. Sprendimui įgyvendinti pasirinktas MQTT protokolas, kuris bus patobulintas su vienkartinių slaptažodžių bei PUF posisteme. Tai sieks išspręsti šias problemas: Sunaudoja mažiau tinklo duomenų kiekio nei MQTT naudojant TLS; Sunaudoja mažiau energijos kiekio nei MQTT naudojant TLS; Patobulinta MQTT protokole aprašyto naudotojo slaptažodžio autentifikacijos metodą pridedant papildomą vienkartinio slaptažodžio reikalavimą; Suteikia naudotojų, kurie jungiasi vienkartiniais slaptažodžiais, privilegijų valdymo sprendimą; Siūlomas komunikavimo metodas apsaugo nuo pakartojimo atakų; Turi modulinį dizainą; Pilnai atitinka MQTT 5.0 standarto reikalavimus. Naują sprendimą galima nesudėtingai integruoti į egzistuojančias sistemas kaip DI srityje plačiausiai naudojamus MQTT brokerius. Siūlomas metodas taip pat suteiks duomenų šifravimo funkciją, nors ir ši dalis paprastai atliekama TLS/SSL protokolo lygyje šifruojant visą MQTT protokolo srautą, siūlomas sprendimas suteiks lengvesnio pavidalo turinio šifravimą, kuris bus taikomas tik pačioms MQTT žinutėms. Tai reiškia, jog visa kita informacija, prisegta MQTT kontroliniame pakete, kaip kliento numeris, tema, bei kita informacija bus siunčiama atviru tekstu.

2.2. MQTT protokolo su vienkartiniais slaptažodžiais komunikacijos metodo vizija

Siūlomas patobulinto MQTT protokolo komunikacijos metodas naudos vienkartinius slaptažodžius kaip autentifikacijos metodą tarp kliento ir MQTT brokerio. Šiam sprendimui įgyvendinti sprendimo topologija susidės iš šių naudotojų. Jutikliai – tai negalingi DI įrenginiai, turintys ribotus kompiuterinius išteklius. Šie įrenginiai sugeba rinkti informaciją ir ją perduoti kitiems įrenginiams. Jutikliai gali būti termometrai, vandens arba elektros skaitliukai, kurie komunikuoja skirtingais komunikacijos kanalais. Šiuo atveju, tai bus įrenginiai, komunikuojantys MQTT protokolu, kurie sugeba tik siųsti informaciją. Tai reiškia jog jie nesugeba gauti jokios ateinančios MQTT informacijos – komunikacija vyksta viena kryptimi. Ši savybė dažnai randama ribotų resursų DI aplinkose, paprastai įrenginiai, galintys klausytis ateinančių MQTT žinučių, gali valdyti kitus įrenginius arba persikonfigūruoti save;

MQTT brokeris – tinkle esantis įrenginys, pavyzdžiui maršrutizatorius, mikrovaldiklis ar serveris su pilna operacine sistema, palaikantis MQTT brokerio funkciją. Šio įrenginio tikslas dabartinėje architektūroje – klausyti ir perduoti visą informaciją iš tinkle esančių jutiklių;

Klientų informacijos duomenų bazė – tai sistema, kurios tikslas autentifikuoti besijungiančius klientus pagal jų pateikiamą informaciją. Kadangi vienkartiniai slaptažodžiai bus generuojami kartu su PUF posistemės pagalba, duomenų bazėje bus reikalinga saugoti tik informaciją apie klientų užklausas ir atsakymus, sugeneruotus jų individualių PUF sistemų pagalba. Dėl šios priežasties ši posistemė bus vadinama CRP posisteme. Pagal šią informaciją bus išgaunami vienkartiniai slaptažodžiai. Šis serveris komunikuos tik MQTT brokeriu, kuris perduos informaciją apie

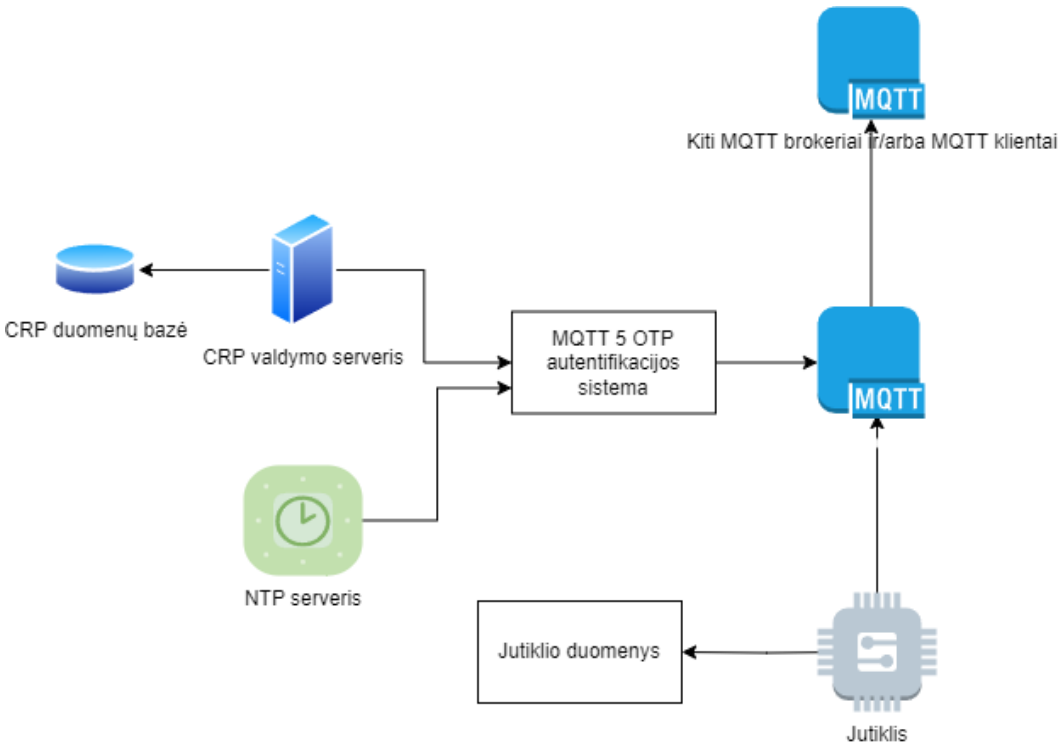
besijungiančius klientus ir, atitinkamai, leis arba uždraus jiems prisijungti. Pagal gautą atsakymą iš šios duomenų bazės, MQTT brokeris žinos kaip toliau elgtis, kai jutikliai bandys prisijungti;

NTP serveris – įrenginys, suteikiantis tikslią laiko informaciją kitiems tinkle esantiems įrenginiams. Tikslus laiko nustatymas yra esminė DI sistemų dalis, kuri leidžia įrenginiams atlikti tikslius ir koordinuotus veiksmus. NTP serveriai dažniausiai nustato tikslų laiką su specialiu laikrodžiu, GPS pagalba ar kitais būdais, tačiau šiame darbe į tai nebus atsižvelgiama. Šis serveris šiame darbe reikalingas norint sinchronizuoti laiką TOTP metodu. Tai yra todėl, nes dauguma negalingų įterptinių įrenginių neturi vidinių laikrodžių, o tai reiškia jog jei jutiklis ir vienkartinių slaptažodžių duomenų bazė neturės sinchronizuoto laiko, jie tarpusavyje negalės teisingai komunikuoti dėl problemų kaip nesuderintas laikas pagal kurį bus generuojami žetonai. Tokios situacijos dažnai gali iškilti kai jutiklis išsijungia ir nebeatnaujina savo vidinio laiko kito įsijungimo metu;

MQTT protokolo su vienkartiniais slaptažodžiais komunikacijos metodas galės veikti dviem pasirenkamais rėžimais: naudojant HOTP arba TOTP vienkartinius slaptažodžius. Sprendimas įgyvendinti abu metodus buvo pasirinktas todėl, nes jie turės skirtingus architektūrinius reikalavimus, kurie ne visada bus įgyvendinami. Abu metodai turi savus pranašumus ir trūkumus:

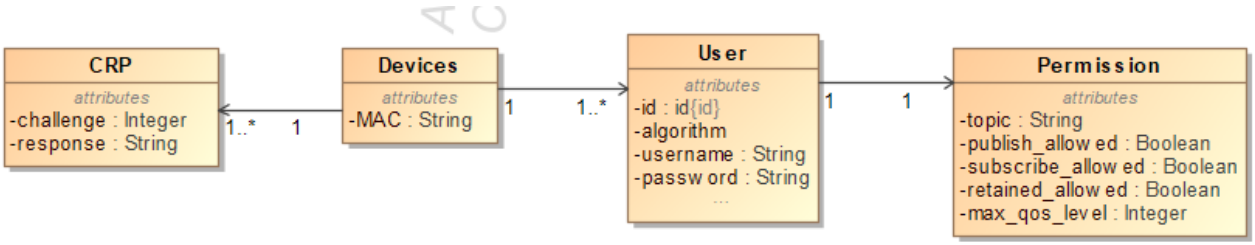
HOTP – šis metodas leidžia naudoti vienkartinius slaptažodžius, naudojant sinchronizuotą skaitiklį tarp kliento ir serverio. HOTP pranašumas yra sinchronizacijos lengvumas lyginant su TOTP – sinchronizacijos kintamasis gali būti skaitinė reikšmė, kuri didinama konstanta, pavyzdžiui, nuo vieno iki N. Šis metodas yra labiau tinkamas sistemose, kai įrenginiai negali turėti sinchronizuoto tarpusavio laiko kaip NTP serverius, arba tikslius vidinius laikrodžius. HOTP trūkumas – sinchronizacijos kintamasis gali nukrypti nuo serverio. Ši situacija pasireiškia, jei klientas bando siųsti vienkartinius slaptažodžius ir pastoviai didina savo sinchronizacijos skaitiklį, kai serveris nėra pasiekiamas ir nežino, jog reikia padidinti savo skaitiklį. Vienas dažniausiai naudojamų sprendimų išspręsti šią problemą yra serverio vienkartinių slaptažodžių reikšmių tikrinimas į priekį nuo esamo skaitiklio reikšmės. Pavyzdžiui, serveriui gavus netinkamą vienkartinį slaptažodį, jis gali patikrinti vienkartinius slaptažodžius, sugeneruotus su toliau einančiomis skaitiklio reikšmėmis, kartojant penkis kartus. Tai dalinai išspręs nesusinchronizuoto skaitiklio problemą. Šiai situacijai spręsti egzistuoja ir kitų sprendimų tačiau jie taip pat turi savus trūkumus ir tik dalinai išsprendžia sinchronizacijos problemą;

TOTP – panašus kaip HOTP metodas, kurio pagrindinis skirtumas yra sinchronizacijos skaitiklio pakeitimas į laiko žymą. TOTP išsprendžia sinchronizacijos problemą tarp kliento ir serverio jei jie turi tikslų laiką. Šio metodo problema – laiko nukrypimas. Problema dažiausiai pasireiškia kai įrenginiai nesugeba turėti tikslų vidinį laiką, pavyzdžiui kai kurie įterptiniai įrenginiai neturi vidinio laikrodžio, todėl jiems išsijungus, vidinis įrenginio laikas nebėra skaičiuojamas. Problema dažniausiai sprendžiama naudojant NTP serverį kuris teikia laiko sinchronizavimo mechanizmą kitiems įrenginiams. Toks sprendimas ne visada gali būti įgyvendinamas, ypač jei įrenginiai egzistuoja lokaliame tinkle ir esama architektūra negali suteikti NTP paslaugos;



1 pav. Patobulintos saugos MQTT protokolo komunikacijos topologijos pavyzdys

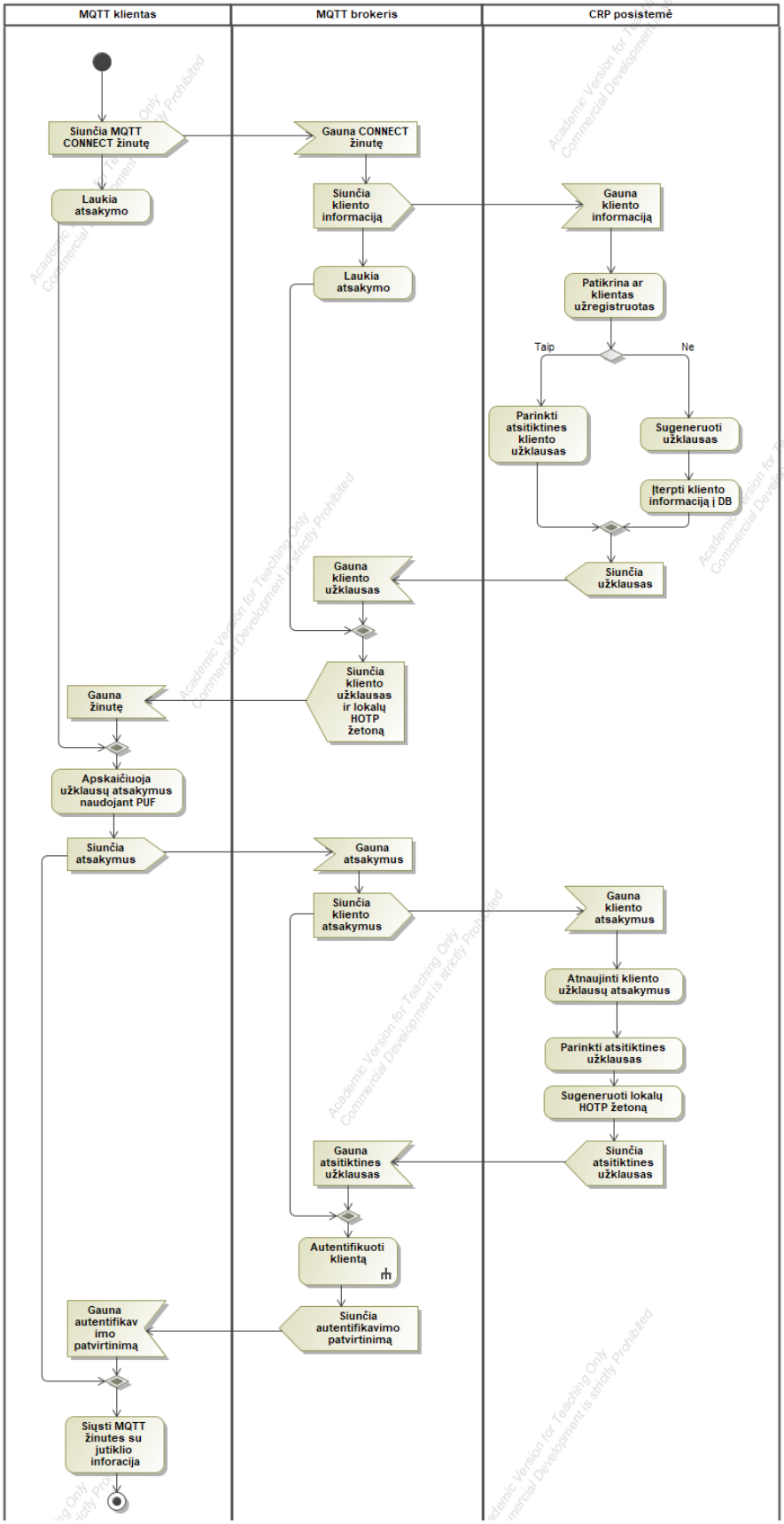
1 pav. pavaizduota siūloma sistema susidaro iš pagrindinės CRP valdymo sistemos, kuri atsakinga už papildomos MQTT klientų autentifikavimo informacijos valdymą tarp CRP posistemės bei MQTT brokerio. Ši sistema pritaikyta veikti kartu su specialiai sukonfigūruotu MQTT brokeriu, kuris perduos CRP valdymo sistemai reikalingą informaciją MQTT klientų prisijungimo metu. Norint naudoti TOTP autentifikacijos metodą papildomai į sistemą bus įdiegtas NTP serveris, kuris yra atsakingas už laiko sinchronizavimą jutikliuose. Siūlomas komunikacijos metodas taip pat skirtas veikti aplinkoje, kur jutikliai ir pagrindinis MQTT brokeris gali būti sujungti tiek lokaliame, tiek pasiekiami per išorinį tinklą.



2 pav. Vienkartinių slaptažodžių duomenų bazės schema

CRP duomenų bazę, pavaizduotą 2 pav. sudarys informacija apie registruotus naudotojus bei įrenginius. Ši informacija yra siunčiama MQTT „CONNECT“ kontrolinio paketo siuntimo metu kai klientas bando jungtis prie MQTT brokerio. Prie šio kontrolinio paketo prisegama informacija apie įrenginio MAC adresą ir papildomas autentifikavimo metodas. CRP valdymo sistema pagal gautą naudotojo informaciją tikrins ar besijungiantis naudotojas yra priregistruotas prie sistemos ir atitinkamai siųs instrukcijas sugeneruoti atsakymus į atsitiktinai parinktas užklausas, o pagal gautą

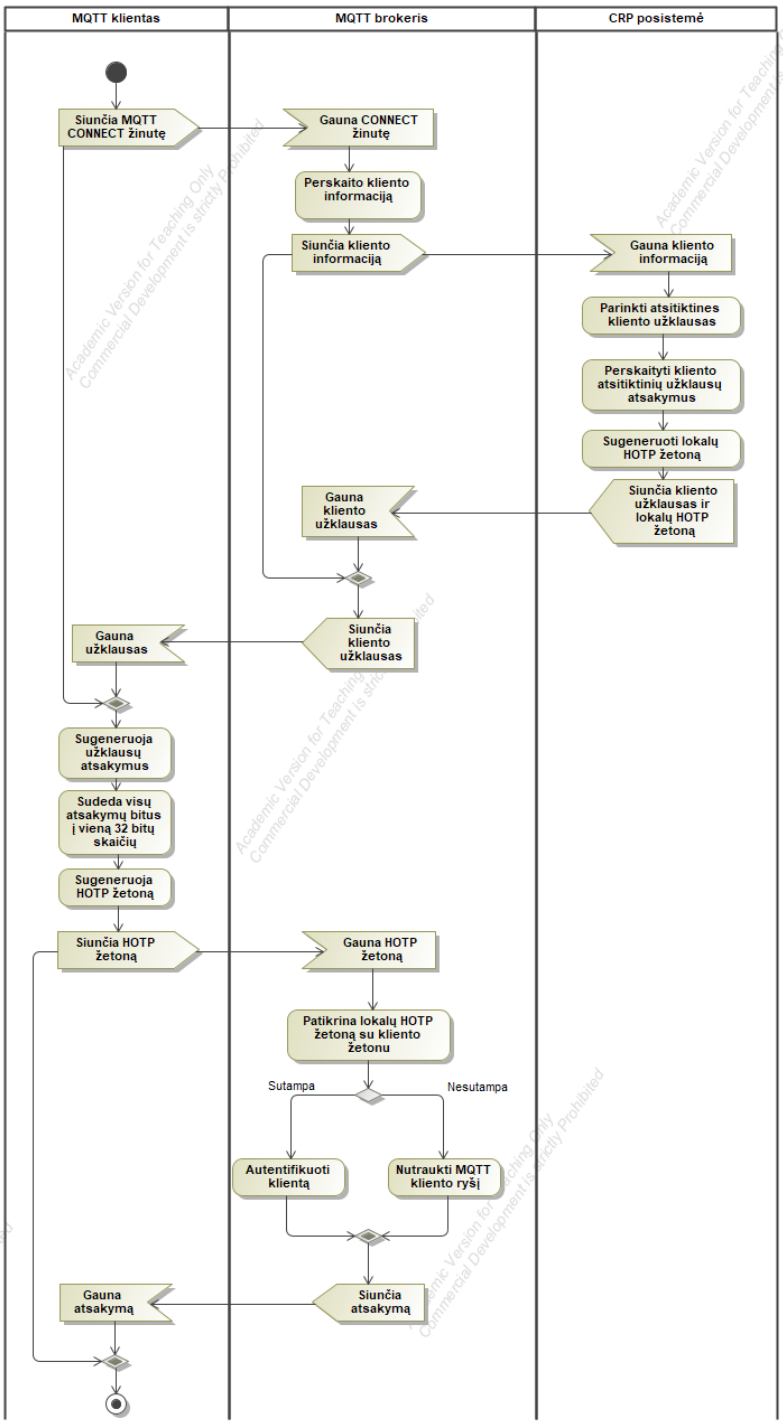
atsakymą bus generuojamas vienkartinis slaptažodis. Norint įgyvendinti privilegijų valdymo funkciją, duomenų bazėje papildomai saugoma informacija apie visų naudotojų privilegijas. Svarbu paminėti tai, jog fizinis įrenginys, kuris užregistruojamas su jo MAC adresu, gali turėti skirtingus MQTT naudotojus. Prieigos teisės susideda iš leidimų siųsti ir gauti žinutes iš nurodytų temų. Papildomai, kiekvienam naudotojui taikomas tikrinimas, kokie pagalbiniai MQTT parametrai yra leidžiami, kaip paslaugos kokybės lygis ir paskutinės žinutės saugojimo leidimas brokeryje.



3 pav. Patobulintos saugos MQTT klientų registracijos proceso diagrama

3 pav. pavaizduota diagrama nurodo jog MQTT klientas, norėdamas prisijungti prie brokerio patobulintu autentifikacijos metodu, privalo nurodyti naują autentifikavimo metodą naudojant papildomus „CONNECT“ kontrolinio paketo parametrus. Prie šios informacijos įrenginiai pridės

savo įrenginio MAC adresą. Norint patikrinti MQTT klientus užklausų atsakymų posistemė kuri saugos klientų informaciją kartu su jų užklausų atsakymų poromis. Brokeris, gavęs „CONNECT“ paketą su patobulintos autentifikacijos mechanizmu, perduos gautą įrenginio informaciją CRP posistemei. Jei įrenginys neegzistuoja, prasidės registravimo procesas: CRP posistemė sugeneruos įrenginiui užklausas, kurios bus perduotos „AUTH“ kontroliniais paketais atgal klientui. Gavus šią informaciją klientas privalės sugeneruoti atsakymus visoms užklausoms naudojant PUF funkcijas. Gauti atsakymai bus atgal persiunčiami CRP posistemei, kuri atnaujins kliento informaciją su gautais atsakymais. Šiuo žingsniu registracijos procesas bus baigtas ir bus pereinama prie kliento autentifikavimo proceso.

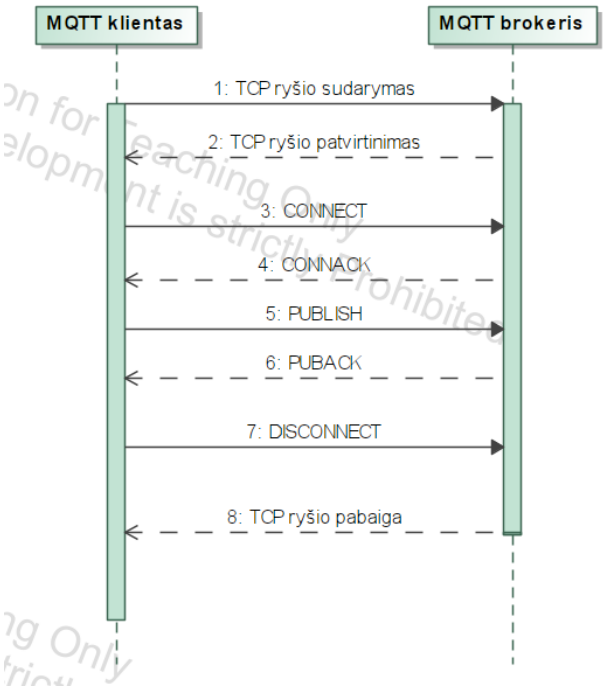


4 pav. Patobulintos saugos MQTT klientų autentifikacijos proceso diagrama

Kliento autentifikavimo metu CRP posistemė parinks dalį užregistruotų kliento užklausų ir jas persiųs atgal klientui, naudojant „AUTH“ kontroliniais paketais. Tuo pačiu metu, serveris galės apskaičiuoti, kokio HOTP žetono tikėtis, kadangi bus žinomi visi užklausų atsakymai. Klientas, gavęs užklausas, sugeneruos atsakymus ir remiantis jais, sugeneruos naują HOTP žetoną, kurį perduos MQTT brokeriui. Brokeris gavęs šią informaciją palygins ar žetonai sutampa ir pagal tai nuspręs ar klientas gali jungtis. Jei klientas autentifikuojamas, siunčiamas „CONNACK“ kontrolinis paketas po kurio klientas galės atlikti įprastą darbą siųsdamas MQTT žinutes. 4 pav. pavaizduotas minėtas procesas.

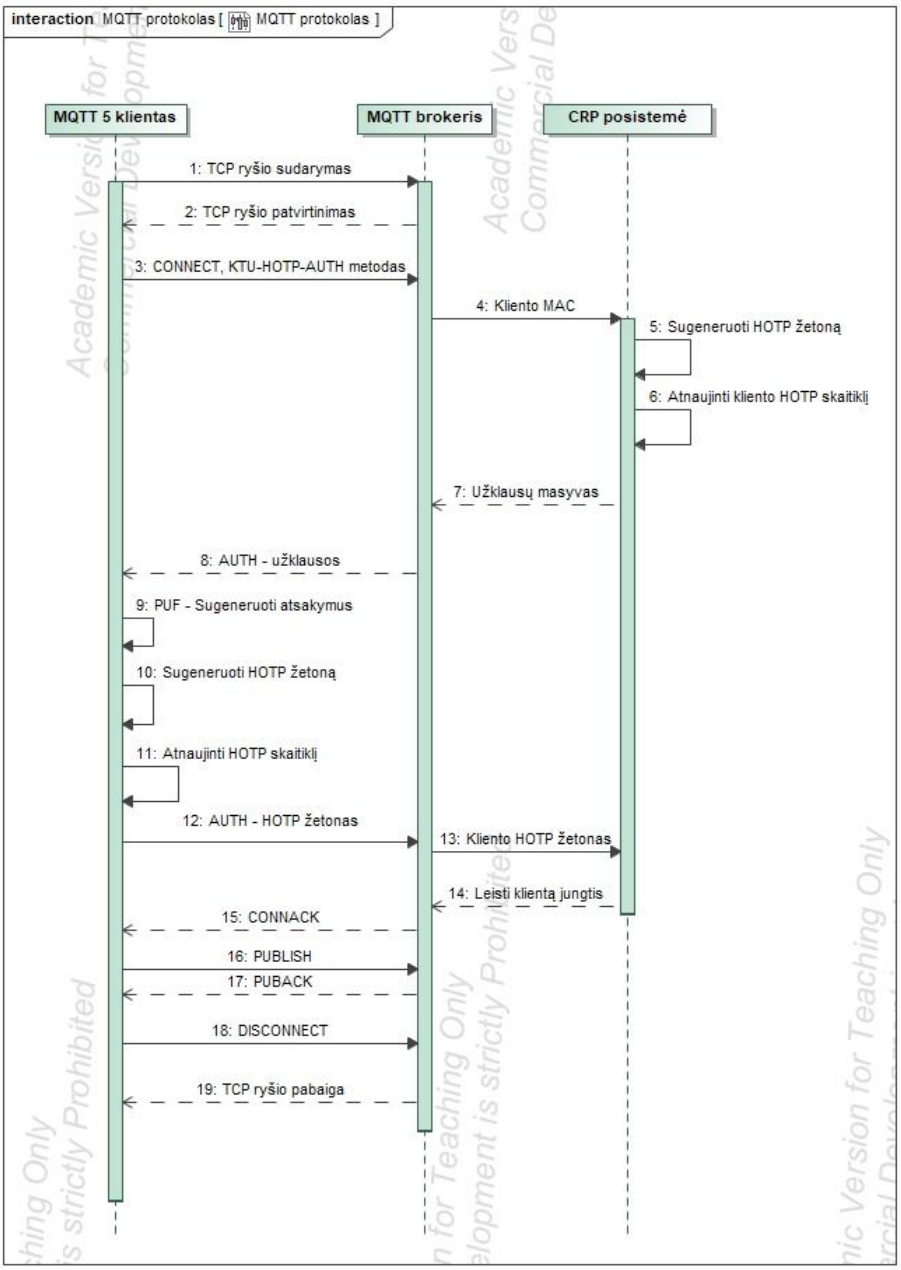
2.3. MQTT protokolo su vienkartiniais slaptažodžiais komunikacijos metodo architektūra

Norint įgyvendinti siūlomą autentifikacijos metodą MQTT protokolu bus reikalinga išnaudoti MQTT 5.0 protokolo versijos siūlomas naujas galimybes. Visi klientai, norintys naudotis šį autentifikacijos metodą, turės siųsti papildomą informaciją susijusią su OTP žetonais, PUF užklausų atsakymų poromis bei savo papildoma informacija MQTT ryšio sudarymo metu. Paprastai klientams tik siunčiant žinutes, protokolo gyvavimo ciklas susidaro iš TCP ryšio kanalo sudarymo tarp kliento ir brokerio. Toliau klientas jungdamasis prie brokerio siunčia tam kontrolinius paketus.



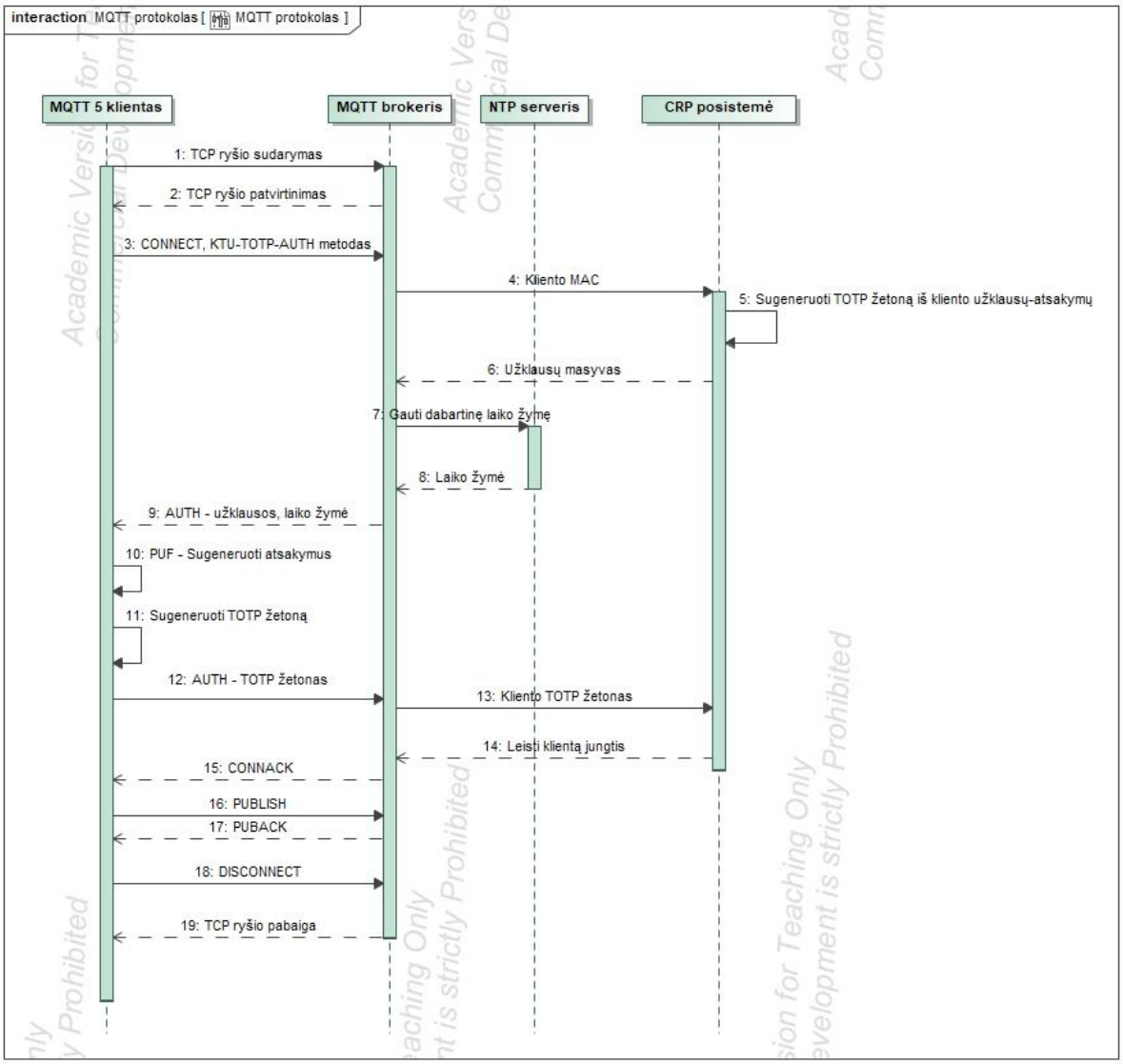
5 pav. Įprastas MQTT kliento prisijungimo procesas

5 pav. pavaizduota diagrama nurodo kaip įprastai atliekama MQTT komunikacija. Klientas siunčia „CONNECT“ paketą, kuris reiškia prašymą sudaryti MQTT protokolo ryšio kanalą su brokeriu. Esant teisingiems ryšio sudarymo parametrams, MQTT brokeris atsako kontroline žinute „CONNACK“ po kurios klientas gali arba užprenumeruoti brokerio temas su „SUBSCRIBE“ – „SUBACK“ arba, šiuo pavyzdžiu, tik siųsti žinutes su „PUBLISH“ – „PUBACK“ kontrolinėmis žinutėmis. MQTT ryšys užbaigiamas su „DISCONNECT“ kontroline žinute po kurios nutraukiamas TCP ryšys.



6 pav. Patobulintas MQTT protokolo autentifikacijos procesas HOTP metodu

Norint įgyvendinti OTP autentifikacijos metodą bus naudojamas naujas kontrolinis paketas – „AUTH“ kuris naudojamas tik MQTT 5.0 protokolo versijoje. Šis kontrolinis paketas siunčiamas iš kliento į serverį arba iš serverio į klientą kaip išplėsto autentifikavimo dalimi. „AUTH“ paketai siunčiami po „CONNECT“ kontrolinio paketo ir prieš „CONNACK“ paketą. „AUTH“ paketai gali būti siunčiami daug kartų jei serveris to prašo, pavyzdžiui, norint gauti papildomos informacijos. Naudojant HOTP vienkartinius slaptažodžius bus reikalinga sekti bendrą skaitiklio reikšmę pagal, kurią bus gaunamas vienkartinio slaptažodžio žetonas. 6 pav. matomi esminiai skirtumai tarp įprasto MQTT kliento komunikacijos eigos.



7 pav. Patobulintas MQTT protokolo autentifikacijos procesas TOTP metodu

7 pav. matomas autentifikavimo mechanizmas, naudojant TOTP metodu kurio metu pagrindinis skirtumas yra papildomai įdiegtas NTP serveris, kuris autentifikacijos metu atsiųs laiko žymę DI įrenginiui, pagal kurį bus generuojamas vienkartinio slaptažodžio žetonas kartu su gautomis užklausų atsakymų poromis. Lyginant su 6 pav. esminis skirtumas yra toks jog nebenaudojamas bendras skaitiklis. Norint pradėti MQTT išplėstinės autentifikacijos metodą klientas pridės papildomą autentifikacijos metodo informaciją „CONNECT“ pakete, šiuo atveju tai bus „KTU-TOTP-AUTH“ arba „KTU-TOTP-AUTH“, atitinkamai HOTP ir TOTP metodams. Autentifikavimo metodas yra susitarimas tarp kliento ir serverio dėl duomenų, siunčiamų autentifikavimo duomenyse ir bet kuriuose kituose „CONNECT“ kontrolinio paketo laukeliuose, reikšmės ir mainų bei apdorojimo, reikalingo kliento ir serverio autentifikavimui užbaigti. Klientas gali iš kart pridėti visą reikalingą informaciją „CONNECT“ pakete, tačiau pasirinkta pilnai išnaudoti „AUTH“ kontrolinio paketo galimybes kurio pagalba galima sukurti kelių žingsnių autentifikavimo mechanizmą. Būtent dėl šios priežasties atsiranda galimybė įgyvendinti sudėtingesnį autentifikavimo procesą kaip įrenginių registravimą ir papildomą laiko sinchronizavimą prieš klientams prisijungiant prie MQTT brokerio. „AUTH“ žinutėse naudojamas papildomas priežasties kodas, kuris nurodo ar reikalinga kitai pusei siųsti papildomas žinutes su 0x18 kodu, ar autentifikacija pavyko su 0x0 kodu. MQTT 5.0 standartas taip pat aprašo trečią priežasties kodą – 0x19, kuris reiškia atlikti pakartotinę autentifikaciją. Šį kodą

galima naudoti pridedant papildomą saugos priemonę, pavyzdžiui, apibrėžiant situacijas, kada prisijungę klientai gali būti prašomi atsiųsti naują vienkartinį slaptažodį po tam tikro laiko, kitu atveju atjungiant juos nuo serverio.

2 lentelė. MQTT kontrolinių žinučių autentifikacija kliento registracijos metu

Eil. nr.	**MQTT žinutės kryptis**	**Kontrolinės žinutės tipas**	**Papildoma kontrolinės žinutės informacija**
1.	Klientas serveriui	„CONNECT“	Autentifikacijos metodas – „KTU-HOTP-AUTH“; Autentifikacijos duomenys – Įrenginio MAC adresas
2.	Serveris klientui	„AUTH“	Autentifikacijos metodas – „KTU-HOTP-AUTH“; Autentifikacijos priežasties kodas – 0x18; Autentifikacijos duomenys – užklausų masyvas
3.	Klientas serveriui	„AUTH“	Autentifikacijos metodas – „KTU-HOTP-AUTH“; Autentifikacijos priežasties kodas – 0x18; Autentifikacijos duomenys – sugeneruotos užklausų atsakymų poros
4.	Serveris klientui	„AUTH“	Autentifikacijos metodas – „KTU-HOTP-AUTH“; Autentifikacijos priežasties kodas – 0x18; Autentifikacijos duomenys – atsitiktinai parinktų užklausų masyvas
5.	Klientas serveriui	„AUTH“	Autentifikacijos metodas – „KTU-HOTP-AUTH“; Autentifikacijos priežasties kodas – 0x18; Autentifikacijos duomenys – OTP žetonas
6.	Serveris klientui	„CONNACK“	Autentifikacijos metodas – „KTU-HOTP-AUTH“; Autentifikacijos priežasties kodas – 0x0;

Remiantis MQTT 5.0 standartu [6] sukuriami du nauji autentifikavimo mechanizmai pavadinimu „KTU-TOTP-AUTH“ ir „KTU-HOTP-AUTH“ priklausomai nuo vienkartinių slaptažodžių panaudojimo tipo. 2 lentelėje pavaizduojamas „KTU-HOTP-AUTH“ pavyzdys. Registracijos metu viso prisideda keturi papildomi „AUTH“ kontroliniai paketai lyginant su įprastu MQTT klientu.

2.4. Vienkartinių slaptažodžių generavimo algoritmai

Norint sukurti vienkartinius slaptažodžius tam realizuoti bus pasirinkti skirtingi slaptažodžių generavimo algoritmai atitinkamai ar bus naudojami laiko žyme pagrįsti arba skaitiklio reikšmėmis pagrysti vienkartiniai slaptažodžiai. Realizuoti HOTP algoritmą reikalingi šie parametrai remiantis [25] dokumentu:

- Slaptas raktas (*K*) – slapta reikšmė, kurią žinos tiek MQTT klientas tiek serveris. Kiekvienas klientas turės unikalų slaptą raktą.
- Skaitiklis (*C*) – skaitinė reikšmė kurios reikšmė turės būti tokia pati tiek kliente ir serveryje.
- Kriptografinė maišos funkcija – naudojama generuojant vienkartinį slaptažodį. HMAC naudos HMAC kartu su „SHA-1“ maišos funkcija. Ši funkcija paims slaptą raktą *K* ir skaitiklio reikšmę C kaip įvestis ir sugeneruos 20 baitų ilgio maišos reikšmę (*H*).
- Remiantis [25] dokumentu HOTP reikšmė iš maišos rezultato gaunamas dviem žingsniais. Maišos rezultato paskutinis baitas pasirenkamas kaip ataskaitos taško indeksas pagal kurį parenkami iš eilės einantys maišos reikšmės baitai. Šiems baitams atliekama dalybos liekanos operacija, iš kurios gaunamas vienkartinis slaptažodis susidedantis iš 6 skaitmenų.

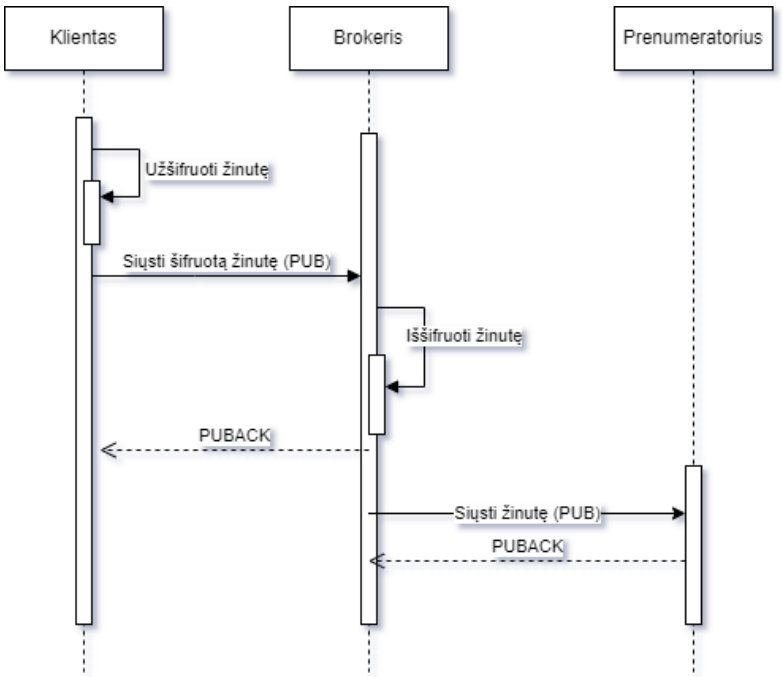
Laiko pagrindu generuojamas vienkartinis slaptažodis (TOTP) yra panašus į HOTP, tačiau vietoje skaitiklio naudoja laiką kaip įvestį, kad generuotų slaptažodžius, kurie galioja tik tam tikrą laiką. Realizuoti TOTP algoritmą reikalingi šie parametrai remiantis [26] dokumentu:

- Slaptas raktas (*K*) – slapta reikšmė kurią žinos tiek MQTT klientas tiek serveris. Kiekvienas klientas turės unikalų slaptą raktą.
- Laiko intervalas (*T*) - dabartinis laikas, padalintas į laiko intervalus, pavyzdžiui 30 sekundžių tarpas. Laiko intervalas yra naudojamas kaip kintamojo įvestis vietoj skaitiklio:
- Kriptografinė maišos funkcija – TOTP naudoja HMAC kartu su „SHA-1“ maišos funkcija, kaip ir HOTP. Ši funkcija paims slaptą raktą ir laiko intervalą kaip įvestis ir sugeneruos 20 baitų ilgio maišos reikšmę.
- TOTP reikšmė generuojama tokiu pačiu principu kaip HOTP.

Norint sėkmingai atlikti autentifikaciją vienkartinių slaptažodžių metodu, reikalinga serveriui, gavus kliento žetoną, sugeneruoti tokį patį lokalų žetoną ir tik tuo atveju jei jie sutampa, traktuoti jog klientas gali prisijungti prie MQTT brokerio. Šiam reikalavimui įgyvendinti reikalinga sąlyga serveriui žinoti kliento žetonų generavimo slaptąjį raktą. Siūlomame patobulintos saugos metode slaptasis raktas nebus saugomas statiniu pavidalu įrenginyje, o vietoje to, bus dinamiškai generuojamas, naudojant PUF. Kiekvienas MQTT kliento įrenginys autentifikacijos metu iššauks PUF mechanizmą, kuris, pagal iš anksto nustatytą užklausą, sugeneruos atsakymą kiekvienai užklausai. Kiekvienas atsakas bus sudarytas iš dvejetainio skaičiaus, o slaptasis raktas bus sudarytas iš 32 bendrai sudėtų PUF atsakų. Kiekviena PUF užklausa susidės iš aštuonių bitų nežymėto skaičiaus. Kiekvieno vienkartinio slaptažodžio slaptojo rakto atgaminimo metu bus naudojamos tik 32 užklausos iš 255 viso galimų užklausų, o tai reiškia, jog kiekvieną kartą generuojant vienkartinio slaptažodžio žetoną, slaptojo rakto reikšmė pastoviai keisis ir rakto reikšmė niekados nebus saugoma tiesiogiai DI įrenginio atmintyje, ar perduodama per tinklą. MQTT brokeris, norėdamas patikrinti kliento autentifikaciją, taip pat, atkurs slaptąjį raktą naudodamas žinomą užklausą ir įrenginio užklausos atsakymą, kuri buvo gauta registracijos metu. Gavus tą patį slaptą raktą, MQTT brokeris sugeneruos vienkartinio slaptažodžio žetoną ir palygins su kliento pateiktu žetonu. Jei jie sutampa, autentifikacija laikoma sėkminga.

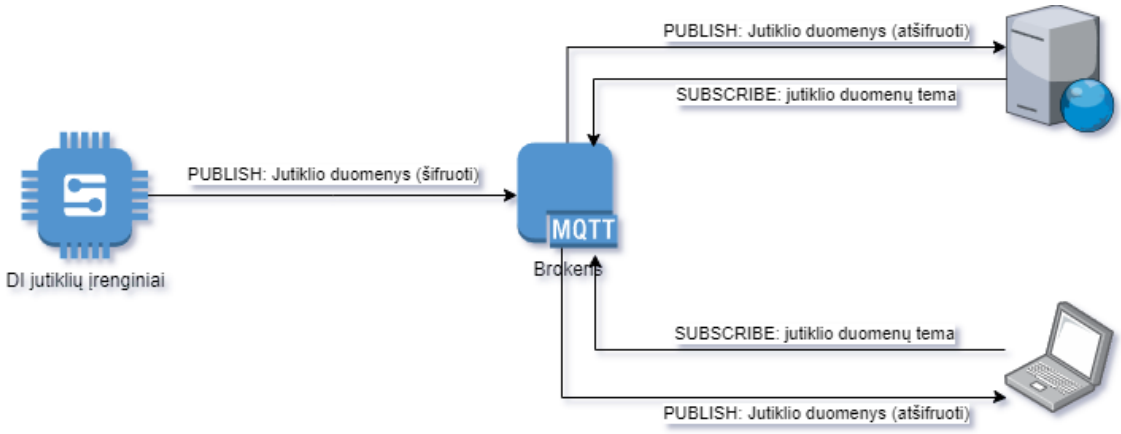
2.5. Papildomas MQTT protokolo srauto šifravimas

Iki šio momento siūlomas patobulintos saugos metodas prideda tik papildomą besijungiančių klientų autentifikacijos žingsnį, naudojant vienkartinius slaptažodžius. Norint toliau pagerinti metodo efektyvumą, bus pridėtas MQTT žinučių turinio šifravimo mechanizmas, pritaikytas negalingiems DI įrenginiams. Kadangi ankstesniuose skyriuose minėta architektūra suteikia sąlygą tiek kliento įrenginiui, tiek MQTT brokeriui dinamiškai sugeneruoti ir išgauti bendrą slaptą rakto reikšmę su PUF pagalba, šį raktą taip pat galima pritaikyti realizuojant šifravimo mechanizmą. Atsižvelgiant į bendrą MQTT protokolo architektūrą, žinučių turinio šifravimas bus tik tarp unikalaus kliento, kuris naudoja papildomą autentifikacijos mechanizmą, ir MQTT brokerio – visi kiti prie MQTT brokerio prisijungę klientai kurie norės gauti žinutes iš tos pačios informacijos temos, gaus jau atšifruotas žinutes. MQTT brokeris, gavęs užšifruotas žinutes iš unikalaus kliento atliks iššifravimo žingsnį prieš žinutės persiuntimą kitiems klientams.



8 pav. žinučių šifravimo pavyzdys

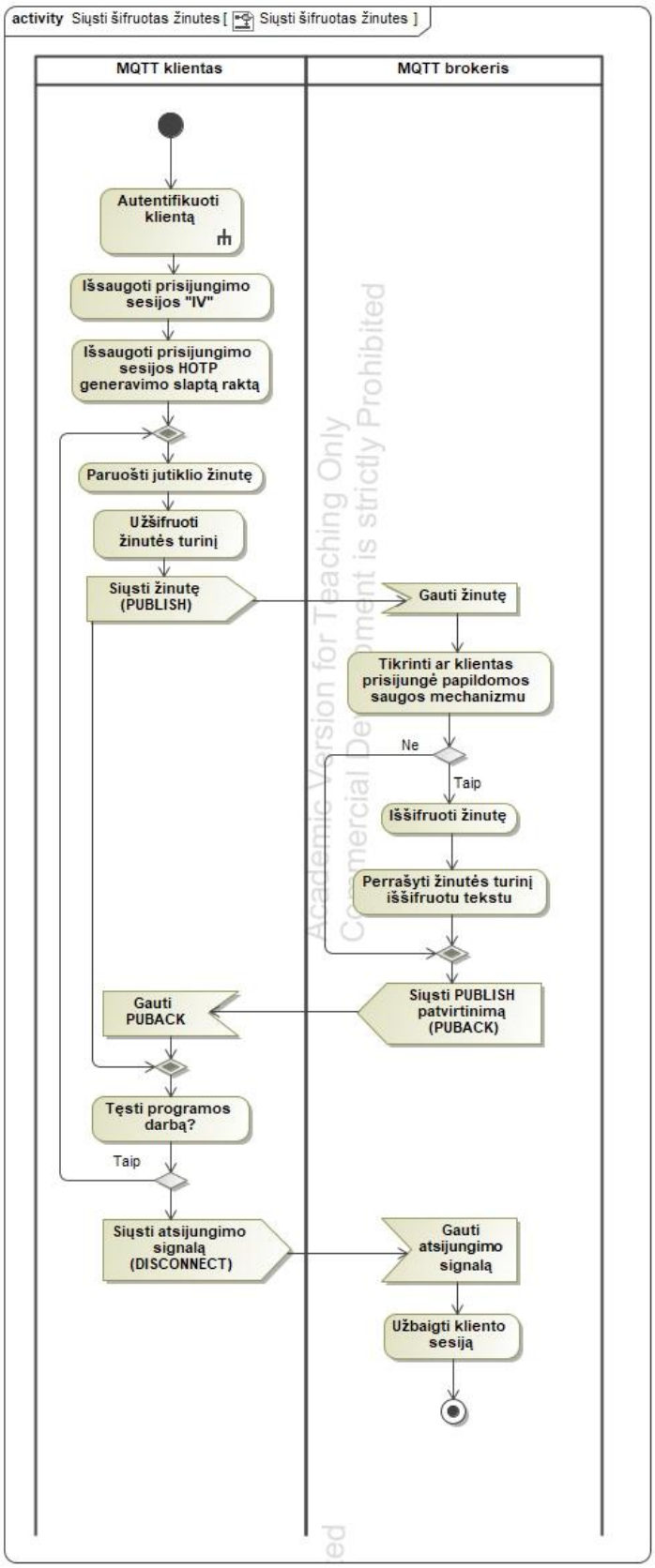
8 pav. pavaizduota, kaip unikalus MQTT klientas naudoja slaptą raktą žinučių turinio šifravimui prieš perduodant jas brokeriui, kuris savo ruožtu atlieka iššifravimą ir tik tada persiunčia informaciją kitiems prenumeratoriams. Toks sprendimas leidžia išlaikyti duomenų slaptumą perdavimo metu tarp konkretaus kliento ir brokerio, tuo pačiu nesukeliant papildomos šifravimo naštos visiems kitiems klientams. Svarbu pabrėžti, kad šis metodas ypač tinkamas resursų ribotiems DI įrenginiams, kadangi šifravimas ir iššifravimas atliekami tik dviem dalyviais – klientu ir MQTT brokeriu – taip išvengiant didelio krūvio daugeliui mazgų ar papildomų procesų visoje sistemoje.



9 pav. žinučių šifravimo architektūros pavyzdys

9 pav. diagrama pavaizduoja kaip atrodo tiesioginė žinučių iššifravimo eiga atsižvelgiant į tai, jog prie MQTT brokerio bus prisijungę ir kiti išoriniai klientai. Šifravimo algoritmų pasirinkimas šioje sistemoje turi būti orientuotas į efektyvų kompiuterinių resursų panaudojimą. Šiam tikslui įgyvendinti siūloma naudoti „AES-CBC“ metodą, kuris yra tinkamas mažos galios įrenginiams dėl jo efektyvumo ir paprastos realizacijos. „AES-CBC“ reikalauja tiek slaptosios rakto reikšmės, tiek unikalios

inicijuojančios vektoriaus „IV“ reikšmės kiekvienai žinutei, siekiant užtikrinti šifruotų duomenų saugumą ir užkirsti kelią šifruoto teksto pasikartojimui. Kadangi PUF mechanizmas suteikia galimybę dinamiškai generuoti slaptąjį raktą tiek kliento, tiek brokerio pusėje, tas pats raktas bus naudojamas žinučių turinio šifravimui ir iššifravimui. Klientas, prieš žinutės išsiuntimą brokeriui užšifruos turinį naudodamas „AES-CBC“. „IV“ reikšmė bus atsitiktinai sugeneruota kiekvienai žinutei ir siunčiama kartu su šifruotu turiniu, kad MQTT brokeris galėtų atkurti pradinį turinį.



10 pav. žinučių šifravimo sekos diagrama

10 pav. diagramoje pavaizduota, jog MQTT brokeris, gavęs šifruotą žinutę, naudos savo generuotą raktą ir „IV“ reikšmę iš žinutės antraštės, kad atliktų atšifravimo operaciją. Po sėkmingo atšifravimo, brokeris perduos jau iššifruotą žinutės turinį visiems prenumeratoriams, išlaikydamas MQTT

standartų atitiktį ir užtikrindamas, kad tik MQTT brokeris ir klientas, naudojantis PUF autentifikaciją, žino šifravimui naudojamą slaptą raktą.

3 lentelė. Patobulintos saugos MQTT protokolo sprendimo funkcijų palyginimas

Funkcija / **Sprendimas**	**Autentifikacijos informacijos apsikeitimas daugiau nei vienu paketu**	**Naudotojo-slaptažodžio autentifikacija**	**MQTT žinučių turinio šifravimas**	**MQTT kontrolinio paketo šifravimas**	**Temų prieigos teisių valdymas**	**MQTT 5.0 standarto atitikimas**
Įprastas MQTT klientas	-	+	-	-	-	+
Įprastas MQTT klientas su TLS	+	+	-	+	-	+
„Mosquitto“ patobulintos saugos įskiepis	-	+	-	-	+	+
[16]	-	+	-	-	+	-
[17]	-	+	-	-	+	+
Patobulintos saugos MQTT klientas	+	+	+	-	+	+

3 lentelėje palyginami anksčiau minėti sprendimai su siūlomu patobulintos saugos MQTT klientu. Pagrindiniai metodo skirtumai yra kelių žingsnių autentifikacijos mechanizmas bei žinučių šifravimas MQTT protokolo lygmenyje. Šios funkcijos leidžia pagerinti bendrą saugumo lygį, ypač pritaikant sistemą DI aplinkoje, kur dažnai naudojami ribotų resursų įrenginiai. Skirtingai nei įprastas MQTT klientas ar sprendimai, kurie pasikliauja TLS kaip išoriniu šifravimo sluoksniu, siūlomas patobulintos saugos klientas realizuoja žinučių turinio šifravimą MQTT protokolo viduje, naudojant tarp kliento ir MQTT brokerio dinaminiu būdu sugeneruotą slaptą raktą. Tokiu būdu išlaikomas duomenų konfidencialumas. Autentifikacijos procesas vykdomas per kelis žingsnius, įtraukiant papildomą kliento registravimo, bei vienkartinių slaptažodžių siuntimo žingsnius. Lyginant su ankstesniais sprendimais, taip pat išsaugotas MQTT 5.0 standarto atitikimas, todėl siūlomas klientas yra pilnai suderinamas su šiuolaikinėmis MQTT ekosistemomis ir gali būti lengvai integruotas į esamus tinklus.

2.6. Architektūros išvados

1. Norint geriau apsaugoti komunikacijos srautą, galima naudoti ir TLS protokolą kartu su patobulintos saugos MQTT klientu, jei tai leidžia naudojamų įrenginių galimybės. Tai reiškia, jog galima atlikti dvigubą šifravimą – MQTT žinučių šifravimą simetriniu šifravimo algoritmu, bei viso MQTT srauto šifravimą asimetriniu šifravimo algoritmu. Tai suteikia ženkliai didesnį lankstumą naudotojams, kurie nori papildomai sustiprinti savo komunikacijos protokolo saugumą. Naudojant tokį metodą, kiekvienas sistemos diegėjas ar administratorius gali pritaikyti skirtingus šifravimo algoritmus ar net papildomus apsaugos sluoksnius pagal savo poreikius, grėsmių modelį ir esamą infrastruktūrą. Tai tampa ypač naudinga sistemose, kuriose skiriasi įrenginių galimybės ar saugumo reikalavimai. Toks lankstumas leidžia ne tik padidinti saugumą, bet ir ateityje lengviau prisitaikyti prie naujų reikalavimų ar technologinių pokyčių;

2. Siūlomas patobulintos saugos sprendimas pranašesnis už kitus siūlomus MQTT sprendimus jei norima kuo efektyviau išnaudoti ribotus kompiuterinius resursus. Patobulintos saugos MQTT klientas siūlo kompromisą tarp siunčiamų duomenų privatumo ir sudėtingų šifravimų algoritmų, parenkant naudoti simetrinius algoritmus vietoje asimetrinių. Tai leidžia protokolą pritaikyti sistemose kuriose keliami griežti saugos reikalavimai išlaikant efektyvų įrenginių resursų sunaudojimą.

3. DI ir M2M sistemų protokolo saugaus komunikavimo metodo taikymas ir prototipas

Šiame skyriuje pristatomas patobulintos saugos MQTT protokolo taikymas ir prototipas. Pirmojoje skyriaus dalyje pateikiama prototipo vizija ir aukšto lygio topologija. Galiausiai pristatoma prototipo realizacija.

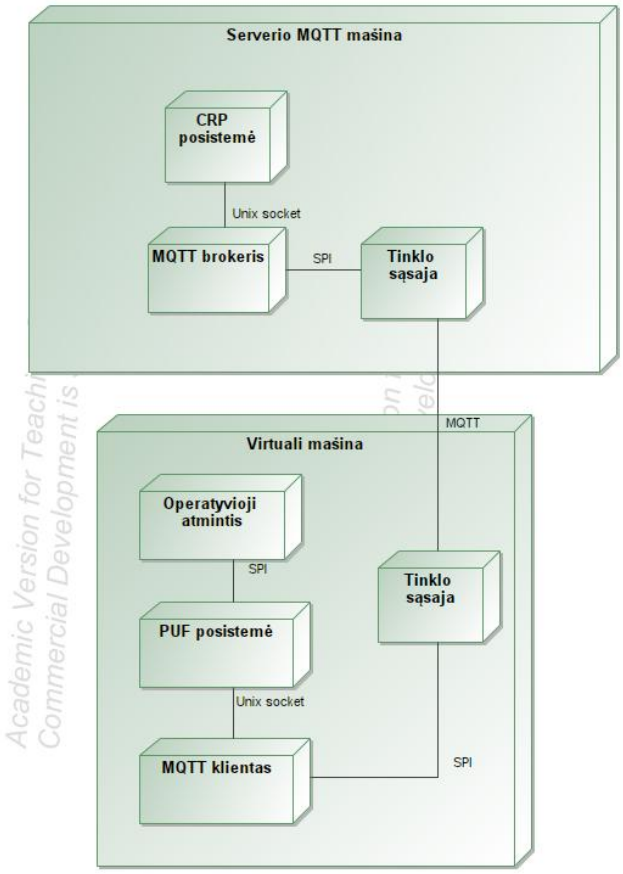
3.1. Prototipo realizacijos topologija

Sistemos architektūra susidės iš šių įrenginių: pagrindinis MQTT brokeris ir mažiausiai vienas DI įrenginys, jei naudojama HOTP vienkartinių slaptažodžių mechanizmas. Kadangi prototipo metu realizuojama HOTP vienkartinių slaptažodžių metodas, NTP serveris nereikalingas. Pagrindinis serveris susidės iš šių posistemių: MQTT brokeris – jis suteiks komunikacijos galimybę tarp DI įrenginių. MQTT brokeris palaikys patobulinto MQTT protokolo funkcijas pagal MQTT 5.0 standartą. Vienkartinių slaptažodžių tikrinimo posistemė veiks MQTT valdymo serverio viduje kaip atskira paslauga, kuri laikys informaciją apie visų užregistruotų įrenginių užklausų atsakymų poras, bei įrenginių sinchronizacijos skaitiklius, reikalingus HOTP autentifikavimo metodui. Šią paslaugą naudos MQTT brokeris gaudamas patobulinto saugos mechanizmo „AUTH“ kontrolinius paketus. CRP valdymo posistemė bus atskira paslauga pagrindiniame MQTT serveryje ir valdys visą susijusią informaciją apie kiekvieno įrenginio PUF funkcijas. Ši posistemė taip pat valdys PUF registracijos procesą, generuojant užklausas ir išsaugojant naujai registruojamo įrenginio atsakymų poras. Ši minėta paslauga sąveikaus su MQTT brokeriu, kuris užtikrins, jog visa reikalinga informacija bus perduota į abi puses „AUTH“ kontrolinių paketų metu. MQTT valdymo serveris bus laikomas kaip serverio lygmens įrenginys, turintis pakankamai kompiuterinių resursų atlikti sudėtingus kompiuterinius veiksmus bei turės pakankamai resursų aptarnauti daug tinkle esančių įrenginių.

DI įrenginiai bus atskiri fiziniai įrenginiai, turintys silpnesnę aparatinę įrangą. Pagrindiniai šių įrenginių reikalavimai yra turėti tinklo sąsają kurios pagalba galės komunikuotis tarpusavyje su kitais įrenginiais tinkle TCP/IP protokolu. Šie įrenginiai turės palaikyti MQTT 5.0 protokolo funkciją, tačiau jiems nebus keliamas reikalavimas palaikyti TLS/SSL protokolo galimybę. Patobulinto saugumo protokolui palaikyti bus keliamas reikalavimas turėti PUF posistemę kuri galės sąveikauti su SRAM arba kitokio tipo operatyvia atmintimi kuri yra rekomenduotina naudoti PUF įgyvendinti su pakankamu stabilumu.

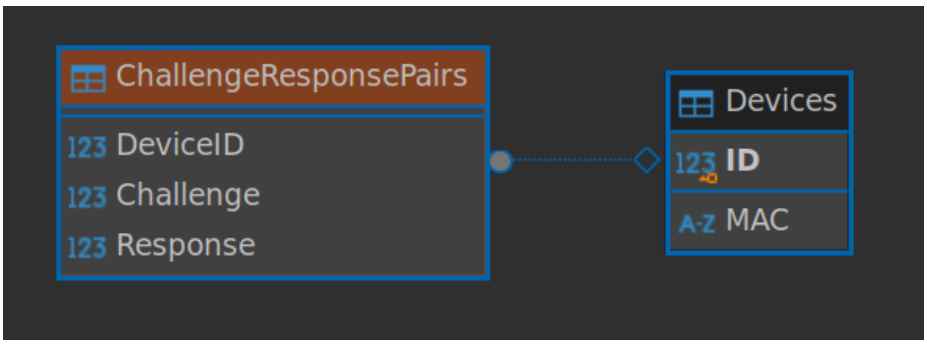
Patobulinto saugumo MQTT protokolui pasirinkimas naudoti PUF funkcijas kaip OTP žetono reikšmę, reiškia, jog MQTT klientai neturės priežasties saugoti šios rakto reikšmes. Šios reikšmės bus sugeneruojamos paties įrenginio kiekvieną kartą gavus atitinkamą iššūkio reikšmę iš MQTT brokerio. Svarbu paminėti, kad kiekvienas įrenginys turės potencialiai daug OTP žetono rakto reikšmių, su kuriomis galės generuoti pavienius OTP žetonus priklausomai nuo apibrėžto užklausų atsakymų. Dėl šio architektūrinio sprendimo pasirinkimo įrenginiai turės didesnį saugumo lygį ir bus atsparesni fizinėms atakoms – piktavalis net ir turintis fizinę prieigą prie įrenginio negalės sužinoti įrenginio OTP žetono rakto generavimo pagrindinės reikšmės, kadangi jie niekados nebus saugomi ilgalaikėje atmintyje. Svarbiausias šio metodo žingsnis, kuriame raktai gali būti sužinoti trečios šalies, yra įrenginio PUF registracijos procesas, kada MQTT brokeris atlieka registraciją užklausų atsakymų apsikeitimo metu. Pasirinkus atlikti registraciją MQTT protokolu, visa informacija bus siunčiama atviru tekstu, todėl svarbu užtikrinti, jog registracijos procesas vyks saugioje aplinkoje, pavyzdžiui, prieš diegiant DI įrenginius į fizines vietas savo darbui atlikti.

Patobulinto saugumo metodui keliami būtent MQTT 5.0 protokolo palaikymo reikalavimai. Nors ir visumoje, bus apsikeičiami vienkartiniai slaptažodžiai, kuriuos būtų galima naudoti vietoje slaptažodžio reikšmės, kuri palaikoma ir senesnėse MQTT protokolo versijose, tačiau šiam metodui reikalinga turėti daugiau parametrų, bei suteikti galimybę MQTT brokeriui siųsti papildomą informaciją klientui prieš jam siunčiant galutinį vienkartinį slaptažodį.



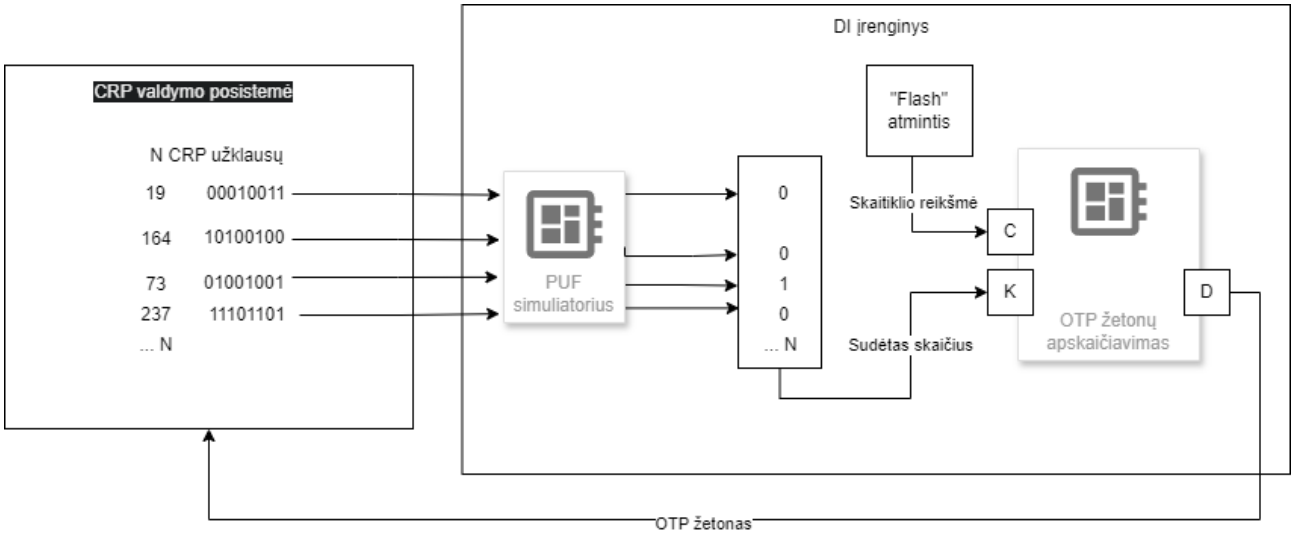
11 pav. Patobulintos MQTT protokolo saugos diegimo diagrama

11 pav. pateikiama diegimo diagrama kurioje pavaizduotas bendras protokolo veikimas. Visi įrenginiai turi tinklo plokštes su kuriomis bus vykdoma komunikacija tarp įrenginių. Serverio virtuali mašina atsakinga už MQTT brokerio veikimą bei patobulintos saugos autentifikavimo mechanizmą. PUF dalies veikimo palaikymui skirta CRP valdymo posistemė, kuri valdys informaciją apie besijungiančius MQTT klientus. Ši posistemė bendraus su MQTT brokeriu, bei perduos reikalingą informaciją apie registruotus įrenginius. Visi MQTT klientai bus atskiri įrenginiai virtualių mašinų pavidalu. Šiems įrenginiams keliamas reikalavimas palaikyti PUF funkcijas, kurios reikalingos atlikti registraciją su CRP posisteme. Realizuojant prototipą pasirinkta visus įrenginius įgyvendinti kaip virtualias mašinas, sujungtas į bendrą lokalų tinklą. PUF funkcijos MQTT klientų įrenginiuose bus imituojamos pagalbinių programų pagalba.



12 pav. CRP valdymo posistemės duomenų bazės schema

12 pav. pavaizduotoje CRP valdymo posistemei įgyvendinti reikalinga duomenų bazė kurioje bus laikoma informacija apie visus registruotus MQTT klientus. Kiekvienas klientas identifikuojamas fizinio įrenginio MAC adresu kuris išsiunčiamas „CONNECT“ kontroliniais paketais. Kiekvienam registruotam įrenginiui priskiriamas sąrašas užklausų-atsakymų porų. Užklausos laikomos kaip 8 bitų skaičiais, o kiekvienos užklausos atsakymas – 1 bitas. Užklausos generuojamos pačios CRP posistemės, o atsakymai apskaičiuojami kiekviename fiziniame įrenginyje su PUF imitatoriaus pagalba arba tikra PUF posisteme.



13 pav. PUF imitatoriaus veikimo pavyzdys

13 pav. pateikiamas pavyzdys kaip PUF imitatorius veiks kaip atskira Python programa kiekviename MQTT kliento įrenginyje. Imitatorius priims numatytą kiekį sugeneruotų užklausų kaip 8 bitų skaitmenis. Kiekviena užklausa bus paversta į atsakymo bitą. Kiekvienas įrenginio PUF imitatorius sukonfigūruotas su iš anksto įrašyta pradžios reikšme, o PUF generavimo algoritmas nustatytas į XOR „arbiter“ tipo funkcijas. Gavus visas užklausas iš serverio ir atlikus PUF bito apskaičiavimus kiekvienai užklausai visi bitai sudedami į vieną 32 bitų sveikąjį skaičių. Šis skaitmuo naudojamas kaip slaptos reikšmės įvestis įrenginio HOTP žetonų skaičiavimo algoritme. Šis skaičius traktuojamas kaip bendra paslaptis tarp autentifikuojančio įrenginio ir serverio. Antroji įvestis yra 8 baitų skaitiklio reikšmė kuris pastoviai kinta kiekvieno žetono apskaičiavimo metu.

3.2. Techninė įranga

Kadangi šis darbas susijęs su DI įrenginiais ir M2M komunikacija prototipo veikimas bus pritaikytas veikti įterptiniuose įrenginiuose kurie sąveikauja su tinklo įranga. Prototipas skirtas „OpenWRT“ operacinei sistemai. „OpenWRT“ yra atviro kodo projektas skirtas įterptiniems įrenginiams. Viena iš atskeltų projektų, paremtų „OpenWRT“ yra „RUTOS“ operacinė sistema naudojama Teltonika įrenginiuose. Šiam protokolui pasirinkta naudoti „Teltonika RUT241“ pramoninį maršrutizatorių. Šis įrenginys Teltonika įrenginių šeimoje yra mažiau galingas lyginant su kitais įrenginiais tačiau labiau finansiškai patrauklus priklausomai nuo sprendimų reikalavimų. Pagrindiniai įrenginio bruožai yra 4G LTE, 3G, 2G mobilaus ryšio galimybės, WiFi 4, vienas WAN ir LAN portas (10/100 MBps), procesorius „Mediatek“ 580 MHz, RAM atmintis 128 MB DDR2, vidinė atmintis 16 MB SPI flash, palaiko protokolus kaip IPv4, IPv6, „OpenVPN“, „IPsec“, „OPC UA“, „Modbus“, MQTT (brokeris ir klientas).

3.3. Programinė įranga

MQTT prototipui įgyvendinti pasirinkta „C“ programavimo kalba. Šis sprendimas paremtas tuo jog ši programavimo kalba yra viena iš plačiausiai naudojamų įterptinėse sistemose. „OpenWRT“ tuo tarpu ir RUTOS projektai pagrinde naudoja „C“ kalbą didžiojoje dalyje įeinančios programinės įrangos. Kadangi prototipas skirtas veikti „RUT241“ įrenginiuose kurie turi „MIPS“ architektūros CPU reikalingas kryžminio kompiliavimo aplinka kuri atliks kompiliavimo procesą teisingo tipo programinei aplinkai. Šiam tikslui įgyvendinti pasirinkta naudoti „buildroot“ kryžminio kompiliavimo sistema skirta „OpenWRT“ aplinkai. Šios aplinkos pagalba sukuriamas visas operacinės sistemos atvaizdas kartu su reikalingais įrankiais kaip pasirinktos architektūros Linux aplinka, jai reikalingi pagalbiniai įrankiai ir visos reikalingos bibliotekos kurios tarpe bus ir patobulintos MQTT komunikacijos posistemė.

MQTT patobulintos saugos mechanizmo protokolui įgyvendinti pasirinkta naudoti pagalbinės „C“ bibliotekos: MQTT protokolą įgyvendinanti biblioteka - „Mosquitto“; vienkartinių slaptažodžių generavimo tiek TOTP, tiek HOTP biblioteka „libcotp“; bendroji kriptografinių funkcijų pagalbinė biblioteka „openssl“; JSON formato apdorojimo biblioteka „cjson“; SQL duomenų bazės sąsajos biblioteka „sqlite3“. MQTT dalies programų kompiliavimui naudojamas „CMake“ įrankis aprašantis kompiliavimo instrukcijas. PUF imitatoriui sukurti pasirinkta naudoti Python aplikaciją kuri palaiko skirtingo tipo pagalbines bibliotekas prototipui įgyvendinti. Pasirinkta naudoti „pypuf“ pagalbinius įrankius. Kodo redagavimo įrankis – „VSCode“ papildomai naudoti įskiepiai palengvinantys darbą: „C/C++“ ir „clangd“ skirti apdoroti kuriamą „C“ programinį kodą, atlikti statinę kodo analizę bei suteikti išorinių bibliotekų funkcijų aprašymo funkciją; „doxygen“ skirtas standartizuoti funkcijų dokumentavimą komentarais; „Github“ skirtas integruoti kodo bazę su „Github“ kodo baze. Prototipo realizacijos metu programinis kodas buvo talpinamas į „Github“ kodo bazę siekiant lengviau sekti besikeičiantį programinį kodą. Prototipo testavimui papildomai naudojamas tinklo srauto stebėjimo įrankis „Wireshark“ kuris palaiko MQTT protokolo skaitymą.

3.4. Prototipo kodo struktūra

Prototipas suskaidytas į tris dalis: patobulinto MQTT saugos protokolo posistemė brokeriui ir klientui bei trečioji – PUF imitatorius. Brokerio posistemė įgyvendinta kaip bendro naudojimo biblioteka, parašyta „C“ programavimo kalba. Ši biblioteka įgyvendina visas pagrindines funkcijas, nurodytas pagal „Mosquitto“ dokumentacijos reikalavimus. MQTT brokeris aprašo kelias atgaliniu

būdu iškviečiamas funkcijas, kurios iškviečiamos tam tikru MQTT brokerio veikimo metu. Tai gali būti įvykiai, pavyzdžiui, naujos žinutės gavimo, išsiuntimo, atmetimo įvykiai. Prototipui kuriamos funkcijos kurios iškviečiamos kai jungiasi nauji MQTT klientai, kurie nurodo patobulintos saugos metodo autentifikacijos metodą, bei kai šie klientai bando siųsti žinutes. Šios funkcijos bus automatiškai iškviečiamos tuo atveju jei MQTT klientas prisegs papildomą autentifikavimo informaciją savo „CONNECT“ pakete. Prototipo brokerio bibliotekos funkcijos bus iškviečiamos su visa reikalinga informacija apie besijungiantį klientą įskaitant jo informaciją kaip IP adresas ir „AUTH“ kontrolinių paketų papildoma informacija. MQTT kliento aplikacija bus atskira sukompiliuota aplikacija, kuri palaikys patobulintą saugos metodą. PUF imitatorius bus atskira posistemė, įgyvendinta kaip „Python“ programa, kurios paskirtis bus PUF funkcijų simuliavimas. Ši programa priims bitų sekos įvestį kaip užklausą ir sugeneruos funkcijos atsakymą. PUF imitatoriaus aplikacija bus inicializuojama su atsitiktine unikalia reikšme kiekvienam įrenginiui, kurio pagrindu, bus generuojami skirtingi atsakymai toms pačioms užklausoms. Šiuo pagrindu bus simuliuojami aparatinės įrangos skirtumai tarp įrenginių. Ši aplikacija bus naudojama kartu su MQTT klientu: šios aplikacijos tarpusavyje komunikuos tuo atveju, kai MQTT brokeris atsiųs užklausų masyvą registracijos ar prisijungimo metu. Prototipo realizacijoje PUF posistemė, kartu su ja sąveikaujančia operatyviąja atmintimi, bus simuliuojama su minėtu PUF stimuliatoriumi. Kadangi šis imitatorius įgyvendintas kaip „Python“ programa, jis užims sąlyginai daug kompiuterinių resursų. Svarbu atsižvelgti į tai, jog programos keliami papildomi kompiuterinių resursų reikalavimai potencialiai bus didesni, nei įprastai naudojamuose negalinguose DI įrenginiuose, galimai didesni nei ir TLS/SSL protokolui keliami reikalavimai. Realioje aplinkoje, kurioje vietoje PUF imitatoriaus, veiks atskira posistemė, kuri komunikuos su įrenginio operatyviąja atmintimi reikalaus potencialiai mažiau kompiuterinių resursų.

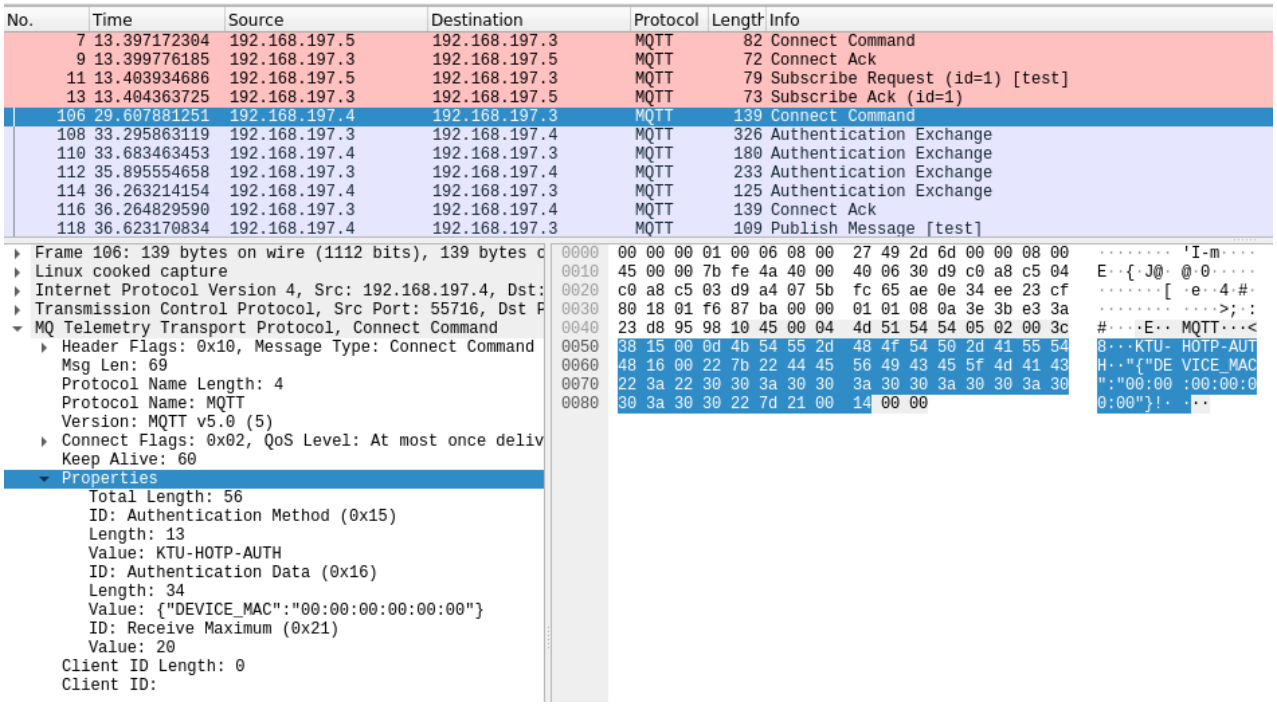
3.5. Prototipo veikimas

Prototipas susideda iš trijų programų: MQTT brokerio įskiepis, kuris valdo informaciją apie besijungiančius MQTT klientus. Įskiepis bus įdiegiamas „Mosquitto“ programos įsijungimo metu kuris bus sukonfigūruotas paleisti papildomus įskiepius; MQTT klientas – programa, kuri jungsis prie brokerio MQTT protokolu; PUF imitatorius – programa kuri apdoros pateiktą 8 bitų skaičių masyvą, kaip PUF įeities kintamuosius ir apskaičiuos visas išeitis. Šią programą kvies MQTT klientas, gavęs reikalavimą sugeneruoti užklausų atsakymus. Įjungus brokerį įjungiamas ir klientas, demonstracijos metu bus parodytos šios funkcijos: MQTT kliento jungties sudarymas naudojant „KTU-HOTP-AUTH“ autentifikacijos metodą; Kliento registracijos procesas; Serverio prašymas sugeneruoti vieną HOTP žetoną; Kliento HOTP žetono generavimo procesas; Serverio HOTP žetono tikrinimas ir sprendimas leisti arba uždrausti klientui prisijungti prie MQTT brokerio. Norint lengviau matyti prototipo veikimą naudojamas „Wireshark“ įrankis stebėti tinklo paketus, kurie bus matomi, kai klientas ir MQTT brokeris bandys tarpusavyje bendrauti. Prototipo demonstracijos metu naudotos trys virtualios mašinos.

4 lentelė. Prototipo demonstracijos metu naudotos virtualios mašinos

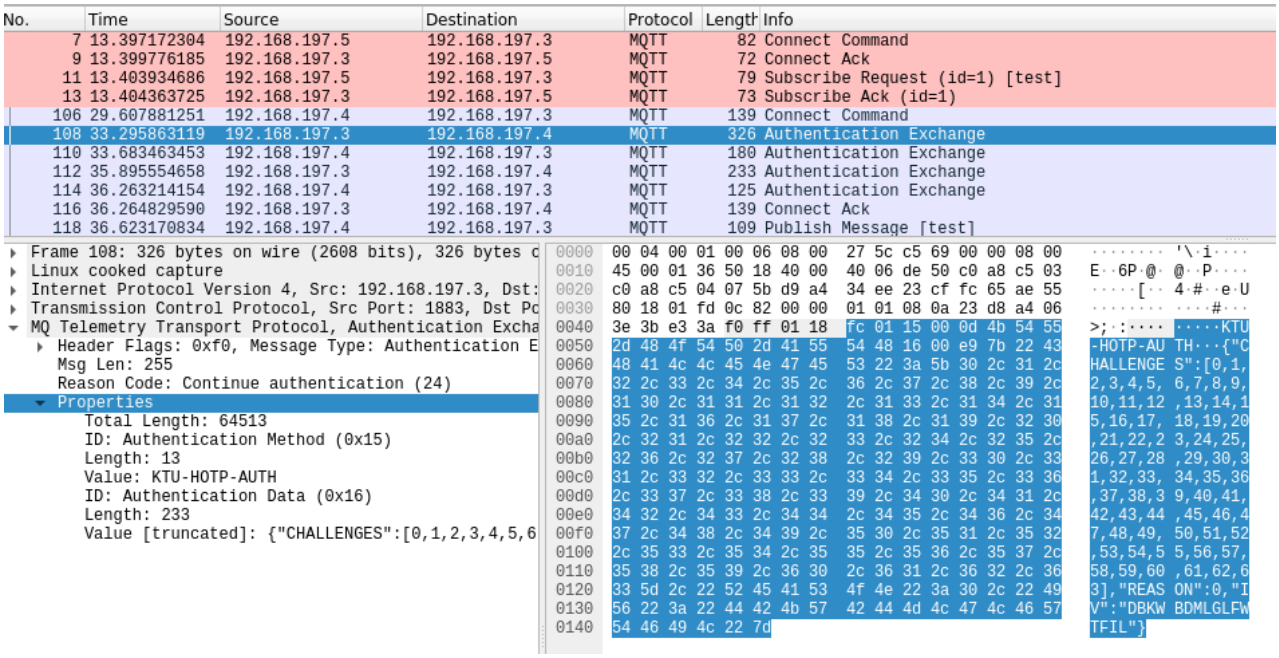
Eil. nr.	IP adresas	Aprašymas
1.	*192.168.197.3*	MQTT brokeris
2.	*192.168.197.4*	DI MQTT klientas naudojantis patobulintos saugos metodą, fiktyvus įrenginio MAC adresas: *00:00:00:00:00:00*
3.	*192.168.197.5*	Įprastas MQTT klientas nesinaudojantis papildomos saugos metodu

Eksperimentinė prototipo demonstracijos aplinka buvo sukurta naudojant tris „VirtualBox“ virtualias mašinas, kurios sujungtos į bendrą vietinį tinklą. Visų virtualių mašinų tinklo plokštės buvo sukonfigūruotos kaip tik įrenginio tinklo adapteriu, leidžiantis užtikrinti izoliuotą ryšį tarp virtualių mašinų ir pagrindinio kompiuterio. Visose VM taip pat buvo įjungtas praleidimo rėžimas su nustatymu leisti prisijungti kitoms virtualioms mašinoms, kad būtų galima pilnai stebėti srautą tarp visų įrenginių naudojant paketų analizės įrankius, tokius kaip „Wireshark“. Visos trys virtualios mašinos veikia tame pačiame potinklyje – 192.168.197.0/24. Papildomai, kiekviena virtuali mašina sukonfigūruota taip, kad palaikytų SSH prisijungimą iš pagrindinio kompiuterio bei atidaryti papildomi prievadai įrenginyje kuris veiks kaip MQTT brokeris: 1883 skirtas nešifruotai MQTT komunikacijai ir 8883 skirtas MQTT komunikacijai su TLS.



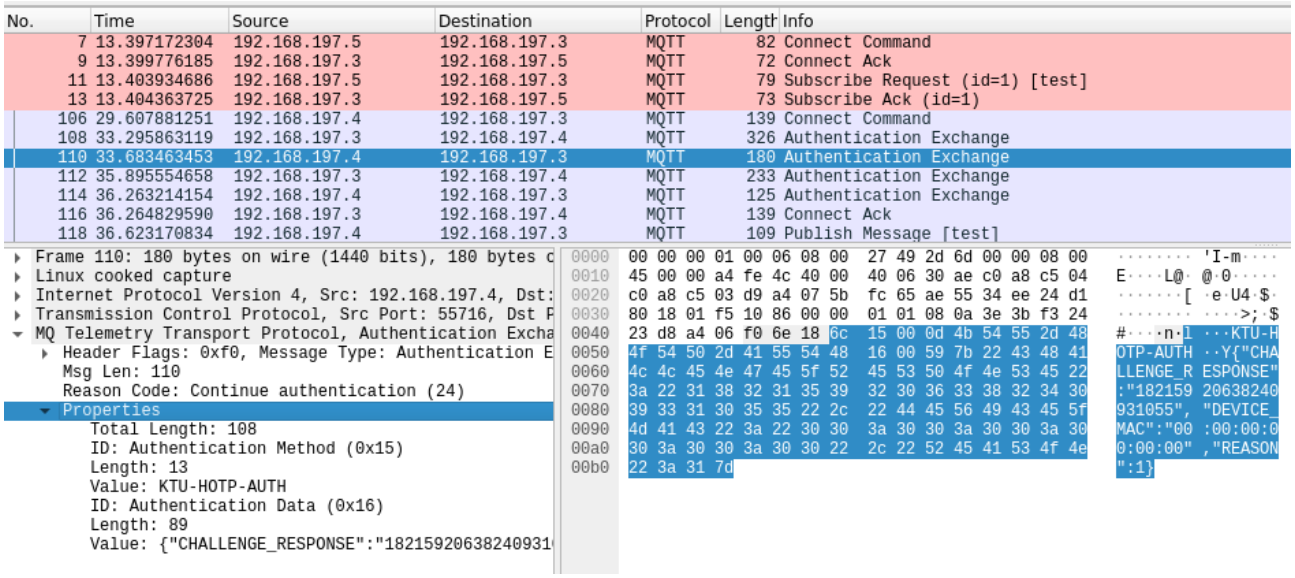
14 pav. MQTT kliento autentifikacijos pradžia

MQTT klientui pirmą kartą bandant jungtis prie MQTT brokerio, nustatomi papildomi parametrai „CONNECT“ kontroliniame pakete. 14 pav. pateikiamas paketo turinys, kuriame matomi papildomi parametrai: pridėtas ID parametras, kurio reikšmė 0x15, arba papildomas autentifikacijos metodas – „KTU-HOTP-AUTH”. Prie autentifikacijos metodo papildomai pridėta papildoma autentifikacijos informacija JSON formatu, kuriame įdedamas besijungiančio įrenginio MAC adresas.



15 pav. MQTT kliento registracijos užklausos

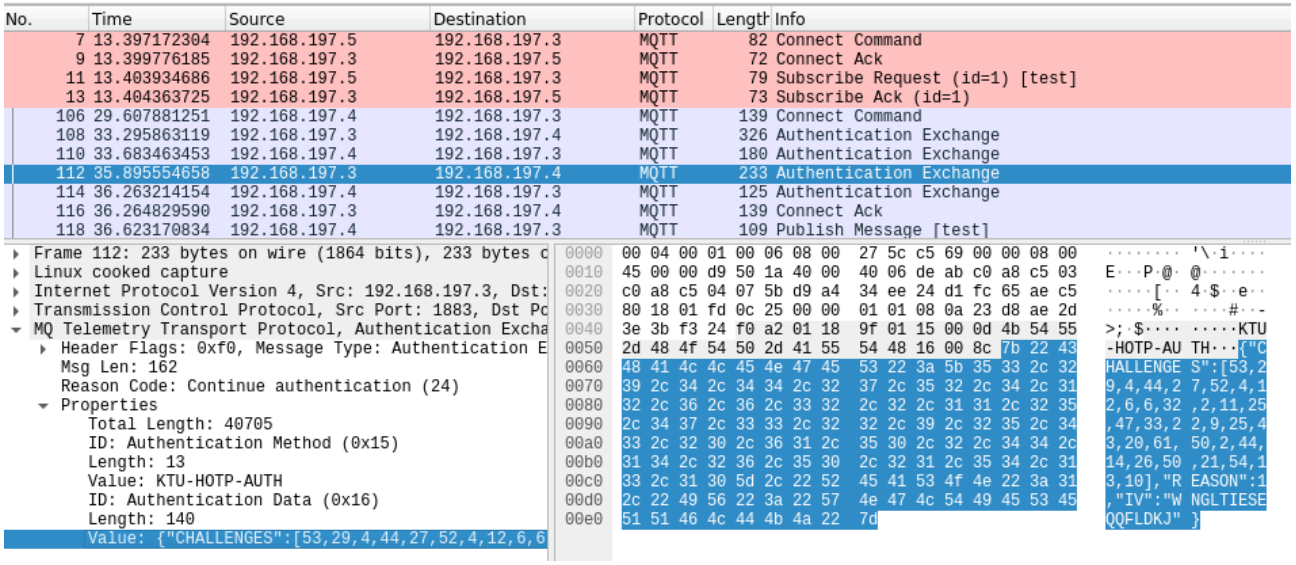
MQTT brokeriui, gavus užklausą iš kliento su patobulintos saugos mechanizmo autentifikavimo metodu, atliekama kliento įrenginio patikra: MQTT brokeris, gavęs įrenginio MAC adresą išsiunčia užklausą CRP valdymo posistemei. Prototipo demonstravimo metu, ši posistemė yra duomenų bazės pavidalu, kurios schema pavaizduota 12 pav. Jei įrenginys nėra rastas, pradedamas registracijos procesas. Prototipe, registracijos metu, serveris išsiunčia 64 užklausas. 15 pav. pavaizduotas „AUTH“ kontrolinis paketas siunčiamas iš serverio, skirtas klientui. Priežasties kodas nurodo, jog sėkmingam MQTT komunikacijos kanalo sudarymui reikalingi tolimesni papildomos autorizacijos žingsniai. Šiuo atveju siunčiamas CRP užklausų masyvas kuris matomas kaip JSON objektas „CHALLENGES“ objekte.



16 pav. MQTT kliento užklausų unikalūs atsakymai

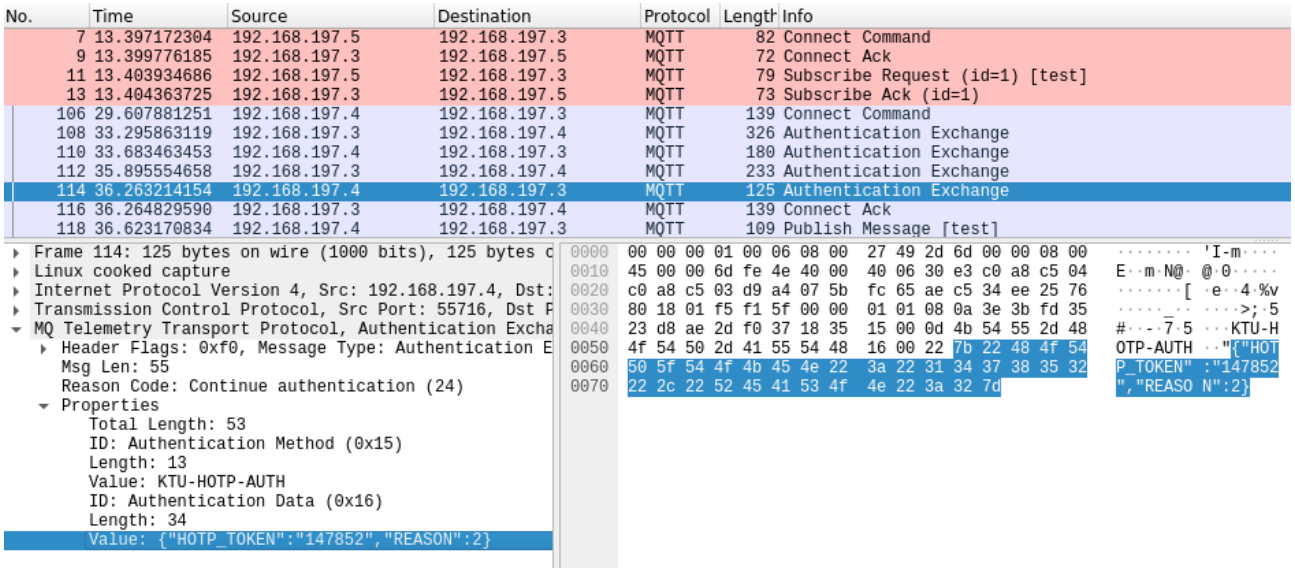
Klientui, gavus registracijos užklausą, perskaitomos užklausų masyvo reikšmės. Kiekvienas užklausos masyvo elementas traktuojamas kaip 8 bitų seka. Visos bitų sekos perduodamos vidiniam

PUF imitatoriui, kuris apskaičiuoja kiekvienai užklausai po atskirą atsakymą. Šis atsakymas skaitomas kaip unikalus kiekvienam fiziniam įrenginiui ir neturėtų būti stabiliai atkartojamas kituose įrenginiuose nuo gamybos proceso skirtumų tarp fizinių įrenginio komponentų kaip SRAM atmintis. Kadangi kiekvienai 8 bitų užklausai skiriamas vieno bito rezultatas, visi rezultatų bitai sujungiami į bendrą 64 bitų skaičių, atliekant bitų poslinkio veiksmą. Galutinis atsakymas išsiunčiamas brokeriui. 16 pav. matomas bendras sujungtas užklausų atsakymas, šiuo atveju, dešimtainė reikšmė „18215920638240931055“, arba dvejetainė reikšmė „11111100 11001011 11110011 01001110 11110011 00001110 10111100 11101111“. Papildomai prisegamas įrenginio MAC adresas bei papildomas priežasties kodas, kuris nurodo brokeriui, jog sėkmingai apskaičiuotas atsakymas.



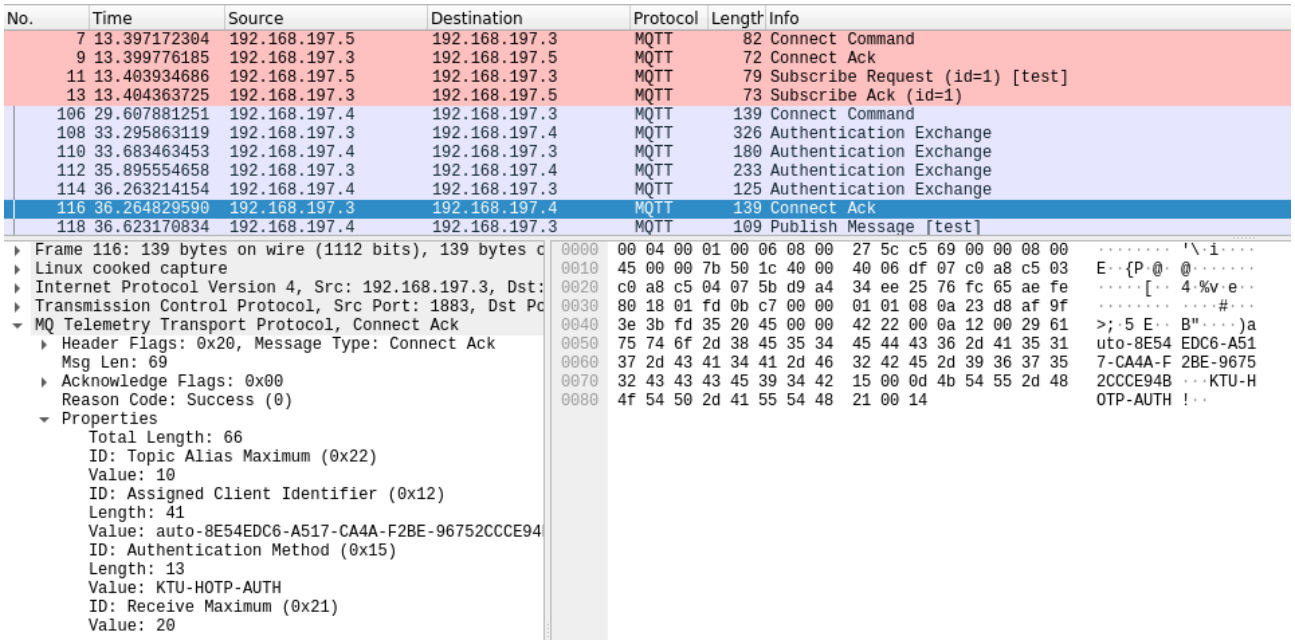
17 pav. MQTT klientui skirtos užklausos HOTP žetono generavimui

MQTT brokeris, gavęs užklausų atsakymus, paverčia dešimtainį skaičių atgal į dvejetainį ir kiekvieną bito reikšmę eilės tvarka užregistruoja CRP posistemėje, kaip atsakymo bito reikšmę atitinkamai užklausai. Registracijos procesui pasibaigus, serveris pereina į įprastą autentifikavimo rėžimą; atsitiktinai parenka 32 užklausas ir jų masyvą išsiunčia klientui, pavyzdys matomas 17 pav. kontrolinio paketo turinyje, „CHALLENGES“ objekte.



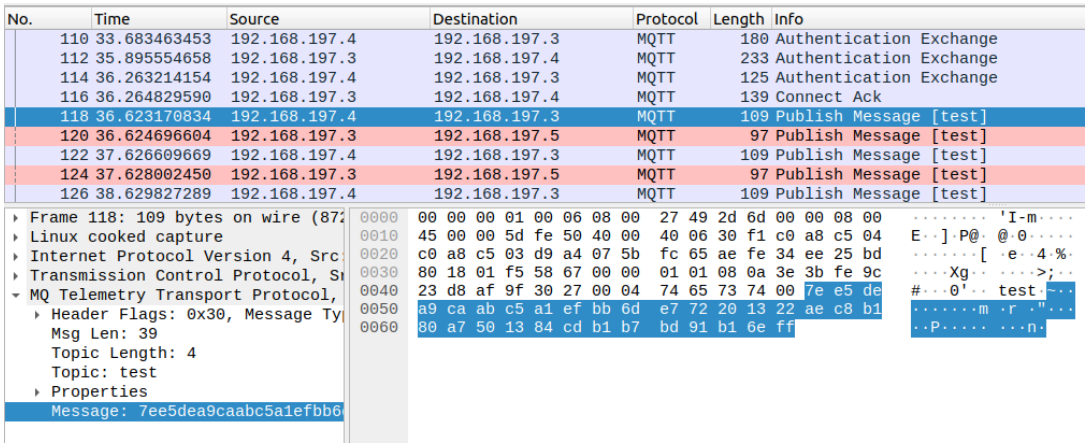
18 pav. MQTT kliento sugeneruotas HOTP žetonas

MQTT klientas kiekvienai gautai užklausai iš naujo apskaičiuoja po atskirą atsakymą ir visus atsakymo bitus sudeda į bendrą 32 bitų skaičių. Šis skaičius naudojamas kaip HOTP algoritmo slaptos reikšmės įvestis ir kartu su skaitiklio reikšme sugeneruoja HOTP žetoną kurio reikšmė matoma 18 pav. Priežasties kodas „2“ duoda serveriui žinoti jog žetonas sėkmingai apskaičiuotas.



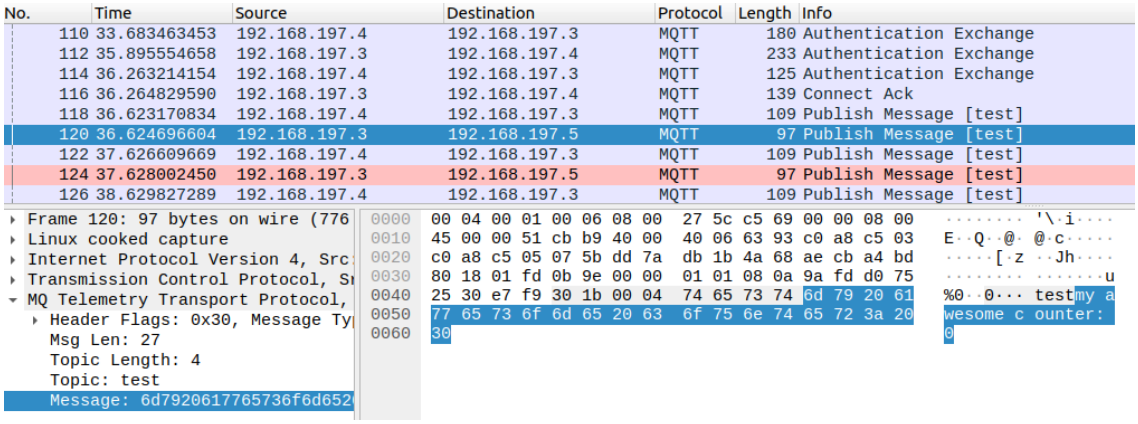
19 pav. kliento leidimas jungtis prie MQTT brokerio

MQTT brokeris, gavęs vienkartinio slaptažodžio žetoną sugeneruoja savo paties žetoną su tokiais pačiais užklausų masyvo rezultatais, kurie saugomi CRP posistemėje kartu su skaitiklio reikšme. Lokaliai sugeneruotas žetonas sulyginamas su kliento atsiųstu žetonu. Priklausomai, ar šios reikšmės sutampa, MQTT brokeris nusprendžia ar duoti leidimą klientui jungtis. 19 pav. pavaizduotas kontrolinis paketas „CONNACK“ nurodo, jog klientas leidžiamas prisijungti prie MQTT brokerio – baigiamas autentifikavimo procesas ir klientas pradeda įprastą darbą, pvz. jutiklio parametrų siuntimą.



20 pav. užšifruotos žinutės turinys

MQTT klientas, naudojantis siūlomą patobulintos saugos metodą, papildomai užšifruoja žinutės turinį naudodamas simetrinį šifravimo algoritmą „AES-CBC“ režimu. Tokiu būdu užtikrinamas papildomas duomenų konfidencialumo lygis, net ir tuo atveju, jei užšifruoti duomenys būtų perimti pakeliui tarp kliento ir MQTT brokerio. 20 pav. pateikiamas pavyzdys iš „Wireshark“ analizės, kuriame matoma, kad nors MQTT paketo struktūra – tokia kaip žinutės tipas, tema, paketo ilgis ir kiti MQTT antraštės parametrai yra lengvai nuskaitomi, pats žinutės turinys atrodo kaip atsitiktinių baitų seka ir negali būti suprastas ar interpretuotas be šifravimo rakto. Tai patvirtina, kad šifravimas buvo atliktas sėkmingai, o žinutės turinys yra apsaugotas nuo pasyvių tinklo stebėtojų.



21 pav. iššifruotos žinutės turinys

MQTT brokeris, gavęs žinutę ir nusprendęs jog ji atėjusi iš kliento, kuris naudoja patobulintos saugos metodą, jis atlieka žinutės turinio iššifravimą, o gautą rezultatą perrašo vietoje užšifruotos žinutės. 21 pav. pateikiamas paketo turinys, kuris skirtas papildomai prisijungusiam MQTT klientui. Šioje žinutėje turinys jau yra iššifruotas.

3.6. Prototipo realizacijos išvados

1. Kadangi įrenginio vienkartinių slaptažodžių žetonų gavimo algoritmui naudojama žetonų generavimo slaptoji reikšmė gaunama iš PUF imitatoriaus, kaip pavaizduota 13 pav. tai reiškia, jog DI įrenginys nesaugos šios jautrios informacijos ilgalaikėje atmintyje. Šis metodas suteikia didesnę apsaugą nuo atakų, kadangi net turint fizinę prieigą prie įrenginio, nebus įmanoma gauti šios jautrios informacijos. Papildomai, MQTT brokerio siunčiama informacija įrenginiui nėra

traktuojama kaip slapta informacija ne registracijos metu. Jis siunčia tik užklausas, kurias unikaliai gali apdoroti ir apskaičiuoti rezultatus, tik unikalios įrenginių PUF funkcijos.

2. Pastebėta jog išplėstos MQTT saugos autentifikavimo metodas gali būti patobulintas – siųsti „AUTH“ kontrolinio paketo papildomą informaciją, kaip JSON formato tekstą, nėra efektyvu, kadangi tai potencialiai reikalaus didesnio kiekio informacijos siuntimo tinklu. Vienas iš sprendimų yra patobulinti autentifikavimo protokolo veiksmų seką, bei iš anksto apibrėžti protokolo komunikacijos siunčiamų duomenų pavidalą. Vietoje JSON teksto galima aprašyti siunčiamus bitus ir jų visas galimas reikšmes.
3. Pastebėta, jog CRP posistemės užklausų atsakymų porų generavimą galima toliau patobulinti. Kadangi kiekvienam vienkartinio slaptažodžio žetonui reikalingas 32 bitų pagrindinis kintamasis, reikalinga turėti ne mažiau kaip 32 užklausų atsakymų porų. Registracijos metu reikalinga įvertinti, kiek reikės sugeneruoti užklausų, kad būtų užtikrintas pakankamai atsitiktinis PUF generavimas.

4. Patobulintos MQTT saugos metodo tyrimas

Tyrimo metu bus apibrėžti parametrai pagal kuriuos bus įvertintas siūlomo metodo prototipo efektyvumas. Taip pat pagal tokius pačius parametrus bus įvertinti bei palyginti kiti esami metodai.

4.1. Patobulintos MQTT saugos metodo tyrimo parametrai

Patobulintos MQTT saugos mechanizmo prototipo tyrimui bus taikomi šie parametrai:

1. Tinklo srauto sunaudojimas ir apkrova – šio parametru bus tikrinama kaip efektyviai siūlomas metodas naudos tinklo srautą, komunikuojant MQTT protokolu. Šis parametras ypač aktualus DI įrenginiuose, kurie turi ribotą tinklo duomenų kiekį, pavyzdžiui, įrenginiai, turintys modemą su „eSIM“, kuriame yra ribotas kiekis mobilių duomenų. Tyrimo metu bus tikrinamas srauto kiekio sunaudojimas esant šioms sąlygoms:
   1.1. MQTT kliento prisijungimas prie brokerio pirmą kartą, kai įrenginys nėra registruotas ir reikalingas registracijos procesas;
   1.2. MQTT kliento prisijungimas prie brokerio kai įrenginys yra jau užregistruotas;
   1.3. MQTT kliento šifruotų žinučių siuntimas brokeriui.
2. Siūlomo patobulintos saugos metodo greitaveika – šiuo parametru bus tikrinama kaip efektyviai siūlomas metodas išnaudos ribotą įrenginio procesoriaus laiką. Šis parametras svarbus atsižvelgiant į tai, jei procesas užims neproporcingai daug procesoriaus laiko, kiti procesai gali veikti mažesniu efektyvumu bei visiškai sustabdyti įrenginio darbą. Tyrimo metu bus tikrinami greitaveikos parametrai esant šioms sąlygoms:
   2.1. MQTT kliento prisijungimas prie brokerio pirmą kartą kai įrenginys nėra registruotas ir reikalingas registracijos procesas
   2.2. MQTT kliento prisijungimas prie brokerio kai įrenginys yra jau užregistruotas
   2.3. MQTT kliento šifruotų žinučių siuntimas brokeriui
3. Siūlomo patobulintos saugos metodo proceso atminties sunaudojimas – šiuo parametru bus tikrinama kaip efektyviai siūlomas metodas išnaudos ribotą DI įrenginio operatyvią atmintį. Kadangi siūlomas metodas pritaikytas veikti silpnuose įrenginiuose, svarbu atsižvelgti, jog aplikacijos neapkrautų įrenginio operatyvios bei ilgalaikės atminties. Neatsižvelgus į šiuos parametrus gali iškilti įrenginio problemos, kaip nenumatytas įrenginio persikrovimas, proceso nutraukimas ir kitos problemos, kurios kelia riziką bendrai sprendimo infrastruktūrai, kurioje įrenginiai įdiegti. Tyrimo metu bus tikrinamas greitaveikos parametrai esant šioms sąlygoms:
   3.1. MQTT kliento prisijungimas prie brokerio pirmą kartą kai įrenginys nėra registruotas ir reikalingas registracijos procesas
   3.2. MQTT kliento prisijungimas prie brokerio kai įrenginys yra jau užregistruotas
   3.3. MQTT kliento šifruotų žinučių siuntimas brokeriui

4.2. Patobulintos MQTT saugos mechanizmo tyrimo metodai

1. Tinklo srauto apkrovos tyrimas padės įvertinti, kiek papildomų duomenų generuoja patobulintas MQTT saugos mechanizmas bei kokia yra bendra tinklo apkrova esant skirtingoms operacijoms. Tyrimo metu bus naudojamas tinklo srauto stebėjimo įrankis „Wireshark“ kuris leis peržiūrėti MQTT paketus. Naudojant filtrus bus atskirti tik MQTT srauto paketai. Palyginimui bus išmatuota standartinio, nešifruoto MQTT srauto ir šifruoto srauto apkrova. Galiausiai bus

analizuojama kiek papildomų duomenų sugeneruoja PUF autentifikacija bei registracijos procesas.

2. Greitaveikos analizės tikslas – įvertinti, kiek procesoriaus laiko sunaudoja papildomi saugumo mechanizmai ir ar jie daro neigiamą įtaką įrenginio veikimui. Bendras proceso laiko matavimas. Šiuo metodu bus tikrinamas proceso veikimas laiko atžvilgiu. Tyrimo metu bus atlikti MQTT klientų programų pakeitimai, skirti matuoti sunaudotą proceso laiką tam tikrais programos vykdymo etapais. Laiko matavimas prasideda užfiksuojant sistemos laiko žymę milisekundžių tikslumu. Pasiekus nustatytą sąlygą, pavyzdžiui, sėkmingą kliento prisijungimą, užfiksuojama antroji laiko žymė. Gautas laiko tarpas apskaičiuojamas atimant pradinę reikšmę iš galutinės. Nors šis metodas yra pakankamai paprastas, jis laikomas efektyviu būdu tiksliai įvertinti atskirus programos vykdymo etapus, pasinaudojant standartinėmis „C“ kalbos sisteminėmis pagalbinėmis funkcijomis. Šiuo atveju, matuojamas sugaištas laikas atliekant žinučių šifravimo operacijas. Kiekvienas įrašas apskaičiuojamas atliekant 10,000 šifravimo operacijų tokiai pačiai žinutei. Šifravimo rakto ilgis pagal siūlomą metodą ir prototipą nesikeičia ir yra 32 baitų ilgio, o „IV“ – 8 baitų. Šifravimas matuojamas po kliento registracijos žingsnio kada išgaunama visa su šifravimu reikalinga informacija. Tyrimo metu keičiamas parametras yra simetrinio šifravimo algoritmas bei šifruojamos žinutės ilgis. Kiekvieno šifravimo algoritmo metu testuojama greitaveika trijų skirtingų ilgių žinutėmis – 7.5, 54 ir 228KB. Tyrimui parinkti šie šifravimo algoritmai: „AES-128-CBC“, „AES-256-CBC“, „AES-128-GCM“, „AES-256-GCM“, „DES-CBC“, „3DES-CBC“

3. Proceso atminties sunaudojimas. Atminties sunaudojimo tikrinimas bus atliktas prisegant šiuos proceso veiklos tyrimo įrankius prie MQTT proceso:

    3.1. „valgrind - memcheck“ – įrankis, skirtas aptikti proceso atminties sunaudojimo klaidas. Šis įrankis taip pat generuoja ataskaitas apie visus išnaudotus atminties baitus. Naudojantis ataskaita galima daryti išvadas apie programos atminties valdymo kokybę, taip pat nustatyti galimus atminties nutekėjimus. „Memcheck“ klasifikuoja atminties nutekėjimus į keturis tipus. Vis dar pasiekiama (angl. „Still reachable“) - tai atvejai, kai vis dar egzistuoja pradinis rodyklės taškas arba jų grandinė nukreipta į atminties bloką. Kadangi į bloką vis dar rodo rodyklė, teoriškai programuotojas galėjo jį atlaisvinti prieš programos pabaigą. Tokie blokai dažni ir dažnai nelaikomi klaidomis todėl „Memcheck“ jų pagal numatytuosius nustatymus atskirai nepraneša; Tikrai prarasta (angl. „Definitely lost“) - tai reiškia, kad į atminties bloką nerasta jokios rodyklės. Kadangi rodyklė neegzistuoja, programuotojas negalėjo jo atlaisvinti. Tokie atvejai dažniausiai rodo klaidą kai rodyklė buvo pamesta anksčiau programos vykdymo metu ir turi būti ištaisyta; Netiesiogiai prarasta (angl. „Indirectly lost“) - tai atvejai kai blokas prarandamas ne tiesiogiai, o todėl, kad visi blokai, kurie į jį rodo taip pat yra prarasti. Pavyzdžiui jei prarandamas dvejetainio medžio šakninis mazgas visi jo vaikų mazgai taip pat bus netiesiogiai prarasti. Kadangi problema išnyksta pataisius pagrindinį, tikrai prarastą bloką, „valgrind-memcheck“ tokių atvejų paprastai atskirai nerodo; Galimai prarasta (angl. „Possibly lost“) - tokiu atveju rasta rodyklių grandinė į atminties bloką, bet bent viena iš rodyklių yra vidinė – rodo į bloko vidų. Tai gali būti atsitiktinė reikšmė atmintyje, pataikiusi į bloko vidų. Todėl tokio atvejo nerekomenduojama laikyti tvarkingu. Prie šios ataskaitos papildomai „valgrind-memcheck“ pateikia informaciją apie iš viso atliktų operacijų kiekį susijusių su atminties išskyrimu. Naudojant šia informacija galima daryti išvadas kaip efektyviai programa valdo kompiuterio atminties resursus. Paprastai šis įrankis

naudojamas ieškant atminties nutekėjimo klaidų. Ataskaitoje galima matyti specifines programos kodo eilučių vietas kur potencialiai buvo neišlaisvinta atmintis. Įrankis sugeba nustatyti ir klaidas kurios atsirado iš išorinių bibliotekų, pavyzdžiui, naudojant „Mosquitto“ biblioteką pagrindinėje programoje galima matyti vietas iš kurios bibliotekos funkcijos kilo atminties sunaudojimo klaidos. Norint efektyviai išnaudoti visas funkcijas reikalinga atlikti pagrindinės programos ir visų susietų bibliotekų kompiliavimą su papildomais derinimo nustatymais. Norint pasinaudoti šią funkciją reikalinga atlikti bibliotekų kompiliavimą su papildomais derinimo nustatymais kaip „CFLAGS+=ggdb”. Taip pat, dažniausiai reikalingas ir papildomas „C“ standartinės bibliotekos perkompiliavimas su papildoma derinimo informacija. Pavyzdžiui, naudojant „OpenWRT“ paremtomis sistemomis paprastai reikalinga perkompiliuoti „libc“ standartinę biblioteką ir perkelti reikalingus failus tiesiai į įrenginį. Taip pat, dažnai reikalinga išjungti binarinio kodo sutraukimą (angl. „binary-stripping“) norint sklandžiai atlikti derinimą kadangi tai panaikina sukompiliuotų programų derinimo informaciją bei nereikalingus meta-duomenis norint maksimaliai sumažinti programų dydį negalinguose įrenginiuose. . Svarbu atsižvelgti į tai jog „valgrind-memcheck“ pritaikytas veikti tam tikrų architektūrų įrenginiuose. Pavyzdžiui, įrenginiuose kaip „Teltonika RUT241“ negalima naudotis šio įrankiu kadangi įrenginio CPU architektūra yra MIPS 24KEc [https://teltonika-networks.com/products/routers/rut241], tai nėra viena iš palaikomų architektūrų „valgrind-memcheck“.

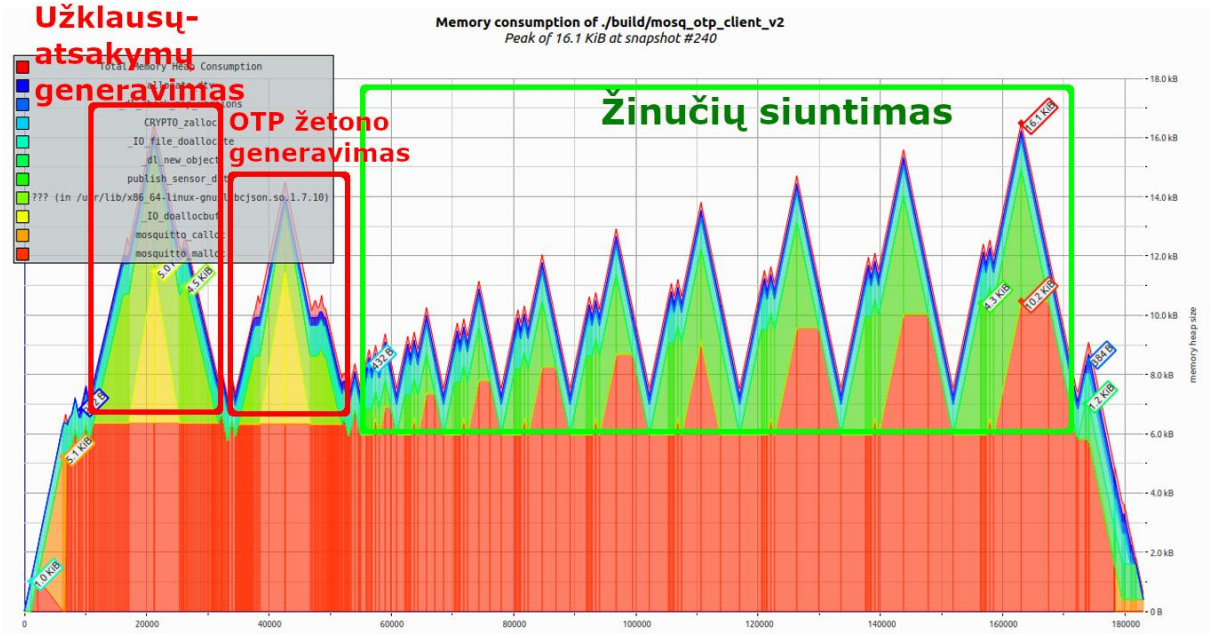
3.2. „valgrind – massif“ – įrankis skirtas matuoti proceso „kupetos“ atminties sunaudojimą per laiką. Prisegus šią programą prie patobulintos saugos MQTT proceso, darbo pabaigoje sugeneruojama ataskaita kurioje pateikiama informacija apie atminties sunaudojimą viso proceso veikimo metu. „massif“ pagal nustatytą laiko tarpą atlieka atminties sunaudojimo tikrinimą daug kartų viso proceso veikimo metu. Tai leis nustatyti kurie programos komponentai sunaudoja daugiausia operatyvios atminties. Šio tyrimo metu pradedamos MQTT klientų programos su skirtingomis konfigūracijomis. Procesai pradedami iš „valgrind-massif“ įrankio. Gauti ataskaitų failai atvaizduoja atminties sunaudojimo kiekį skirtingu metu. Įrankis papildomai sukonfigūruotas išsaugoti atminties vaizdą maksimaliai 500 kartų bei atlikti vaizdo saugojimo žingsnį kiekvieną kartą kai atliekama atminties priskyrimo operacija. Gautas ataskaitos failas yra tekstinio pavidalo nurodantis baitų kiekį kiekvieno atminties vaizdo išsaugojimo metu. Norint aiškiau suprasti atminties sunaudojimo ataskaitą naudojamas papildomas įrankis „massif-vizualizer“ kuris tekstinį failą paverčia į grafiką. Sugeneruotas grafikas leidžia lengvai matyti bendrą proceso atminties sunaudojimą. Papildomai „massif-vizualizer“ leidžia išskaidyti grafiką į sudedamąsias dalis – galima matyti kiek atminties naudoja susietos bibliotekos, pavyzdžiui Mosquitto, cJSON, OpenSSL ir kitos. Grafiką galima dar detaliau skaidyti į individualias kiekvienos bibliotekos dalis, tokiu būdu matant iš kokių specifinių bibliotekų funkcijų atliekamos atminties išskyrimo operacijos.

Naudojant MQTT klientą su TLS protokolu, prieš pradedant darbą būtina atlikti papildomus parengiamuosius žingsnius. Visų pirma, reikia sugeneruoti PEM formato sertifikatus tiek serveriui, tiek klientui. Kad klientas ir serveris galėtų tarpusavyje patvirtinti tapatybę, būtinas ir tėvinis sertifikatas, kuriuo bus pasirašyti abu – kliento ir serverio – sertifikatai. Šiam tikslui „Ubuntu“ sistemoje naudojama „openssl“ komandinė priemonė, su kuria sugeneruojami viešieji ir privatūs tėvinio, serverio bei kliento sertifikatų raktai. Kadangi šie sertifikatai bus naudojami tik prototipo

tyrimui, tėvinis sertifikatas pasirašomas savęs paties. Tada sukuriami sertifikatų pasirašymo prašymai „CSR“ tiek klientui, tiek serveriui. Šiuose prašymuose užpildoma pagrindinė identifikacinė informacija: šalis, organizacijos padalinys bei bendrasis vardas. Bendrojo vardo lauke, serverio atveju, įrašomas jo IP adresas. Sukūrus CSR failus, jie pasirašomi tėviniu sertifikatu, taip sukuriant galiojančius klientų ir serverio sertifikatus. Kai sertifikatai paruošti, papildomai sukonfigūruojamas MQTT brokeris. Šiame etape išjungiama patobulintos saugos įskiepio funkcija ir vietoje to įjungiamas TLS protokolo palaikymas, nurodant atitinkamus failus: serverio privatų raktą, viešąjį sertifikatą bei tėvinį sertifikatą. Brokerio konfigūracijoje taip pat nurodoma, kad priimami tik tie klientai, kurie taip pat naudoja TLS. Galiausiai, kliento programa pritaikoma naudoti TLS protokolą – tai nesudėtinga atlikti su „mosquitto“ biblioteka, tiesiog nurodant kliento sertifikatų failus bei tėvinį sertifikatą. Taip pat pakeičiamas naudojamas prievadas – vietoje numatyto 1883 prievado naudojamas 8883 prievadas, skirtas TLS ryšiui.

4.3. Patobulintos MQTT saugos mechanizmo tyrimo rezultatai

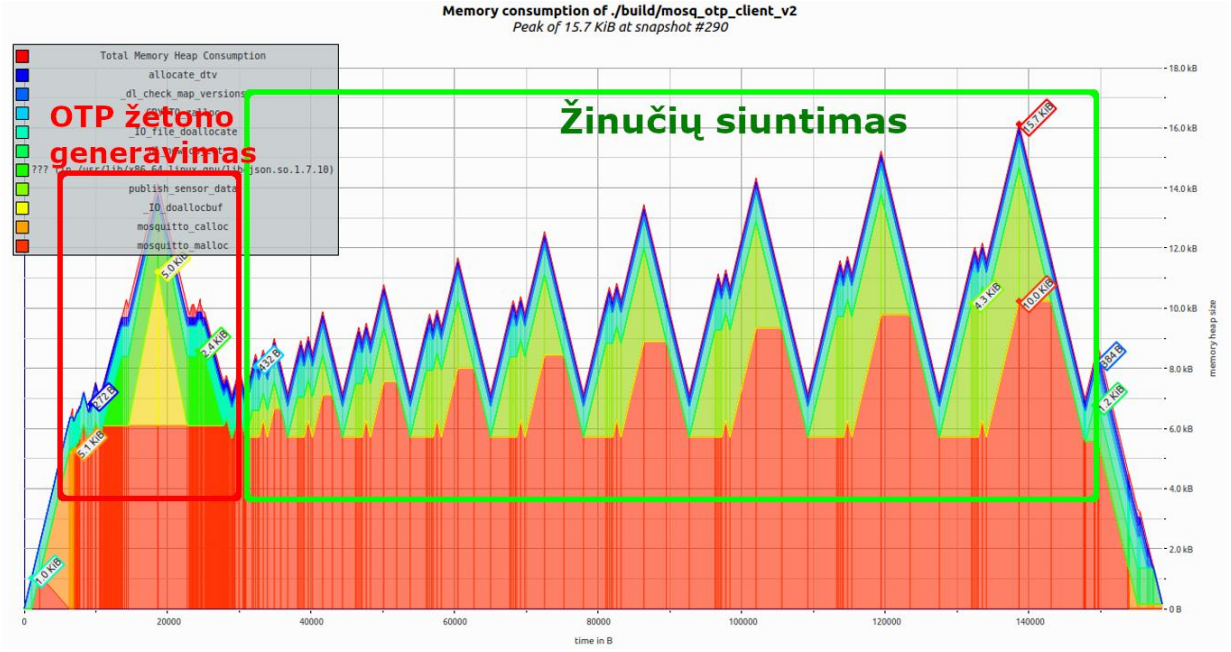
Svarbu paminėti jog atminties sunaudojimas yra objektyviau vertinamas parametras, kadangi įrenginio centrinio procesoriaus charakteristikos nedaro tiesioginės įtakos šiam rodikliui. Tuo tarpu greitaveikos testų rezultatai gali ženkliai skirtis priklausomai nuo realaus naudojimo scenarijaus. Taip pat svarbu atkreipti dėmesį, jog testuojant sprendimą buvo naudojamas tik vietinis (LAN) tinklas – ryšio kokybė per platesnio masto tinklus (WAN), ypač naudojant mobilųjį ryšį, nebuvo vertinama. Dėl to našumas gali kisti priklausomai nuo tinklo sąlygų, kurios yra sunkiai prognozuojamos ir kontroliuojamos.



22 pav. patobulintos saugos metodo MQTT kliento atminties sunaudojimo grafikas su registracija

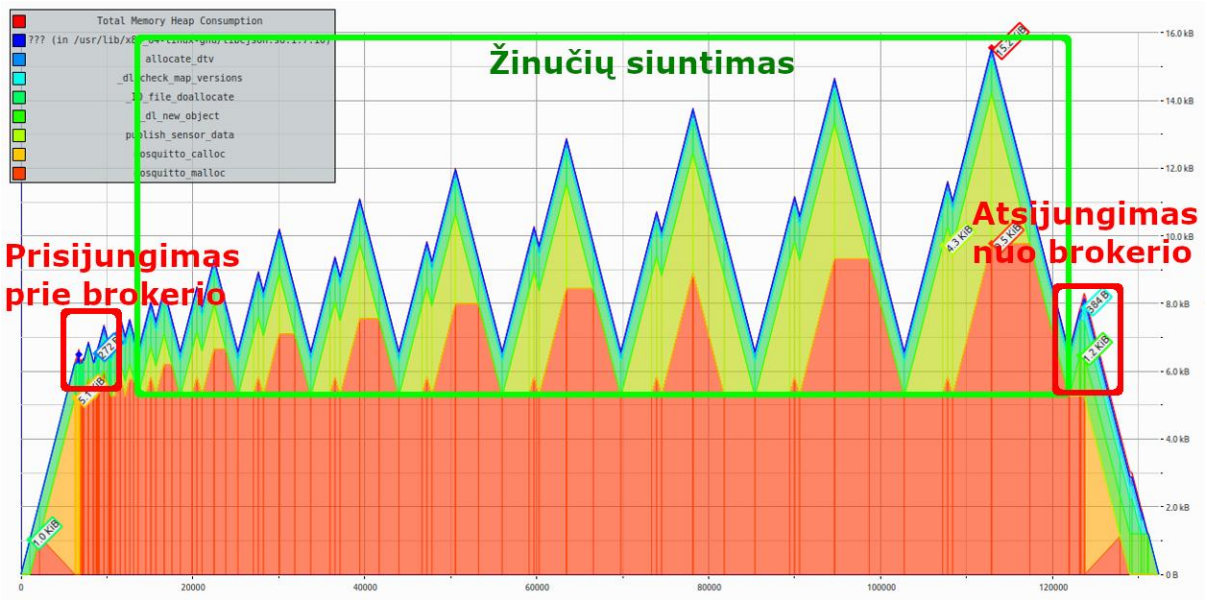
Pirmame scenarijuje, kuris pavaizduotas 22 pav. testuojamas klientas su patobulintos saugos autentifikavimu kuris nėra užregistruotas. Autentifikacijos metu gavus instrukciją sugeneruoti užklausų atsakymus iš MQTT brokerio matoma jog proceso atmintis išauga iki ~16.1 kB. Šiame proceso momente buvo išskirta daugiau proceso atminties norint apdoroti gautą užklausų masyvą iš brokerio tai pat ir gautas sugeneruotas atsakymų masyvas bei sudėta atsakymo žinutė susidedanti. Kitas sunaudotos atminties šuolis yra OTP žetono generavimo metu, kai MQTT brokeris iš naujo atsiunčia tik dalį užklausų. Kadangi užklausų kiekis yra mažesnis nei visų galimų užklausų, matomas

ir mažesnis sunaudotos atminties kiekis, ~14.2 kB. Klientui prisijungus prie brokerio pradedamas žinučių siuntimas. Visuose scenarijuose siunčiama po 10 žinučių. Kiekvieno žinutės išsiuntimo metu didinamas vidinis skaitiklis pagal kurį padidinamas žinutės turinys ~1024 raidžių. 18 pav. matomi laiko momentai kada atliekama žinutės formavimo operacija bei išsiuntimas į brokerį. Paskutinės didžiausios žinutės išsiuntimo metu buvo išskirta ~16.1 kB atminties.



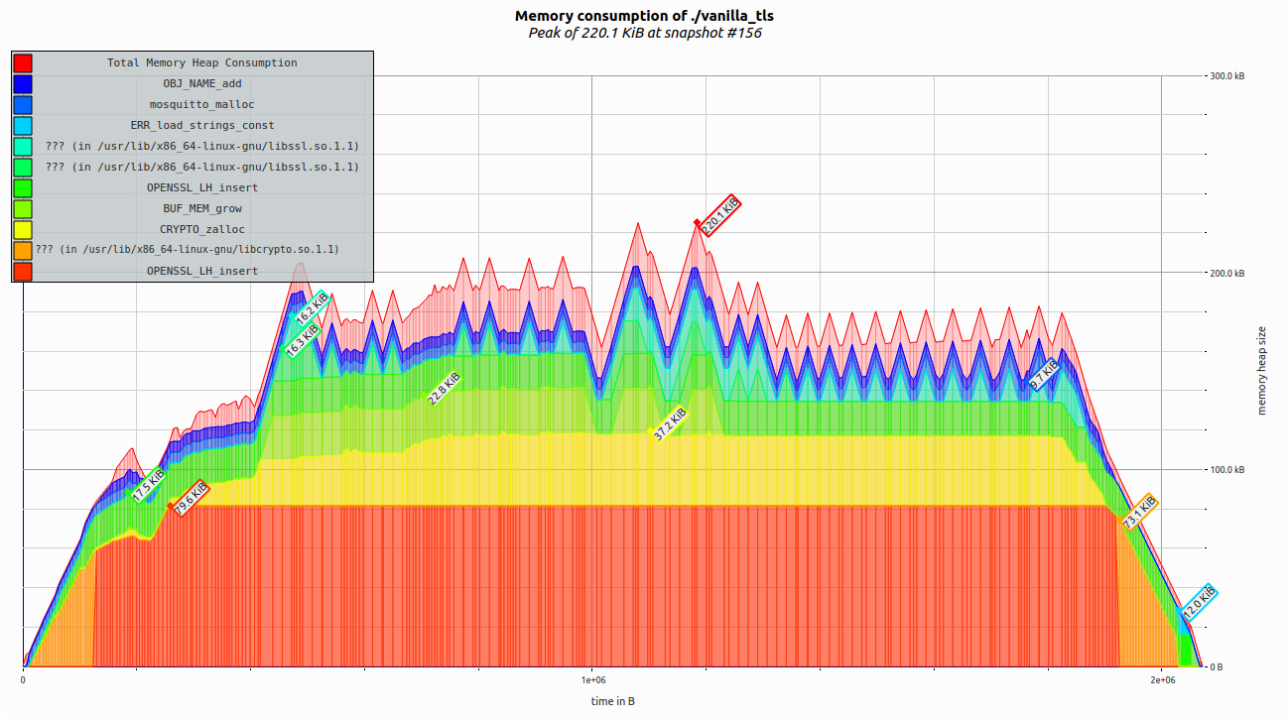
23 pav. patobulintos saugos metodo MQTT kliento atminties sunaudojimo grafikas be registracijos

Antrame scenarijuje testuojama tokia pati programa kaip ir pirmame scenarijuje, tačiau šiuo atveju klientas bus jau iš anksto užregistruotas prie PUF posistemės. Kadangi kliento autentifikavimo metu nereikalingas registracijos žingsnis atliekamas tik OTP žetono generavimas. 23 pav. matoma, jog nebėra atminties išskyrimo šuolio kuris buvo matomas pirmame scenarijuje. OTP žetono generavimo metu išskirta atmintis siekė ~13.8 kB. Toliau vėl siunčiamos tokios pačios šifruotos žinutės su kylančiu teksto kiekiu. Paskutinės žinutės išsiuntimo metu išskirta ~15.7 kB atminties



24 pav. įprasto nešifruoto MQTT kliento atminties sunaudojimo grafikas

Trečiame scenarijuje naudojama skirtinga programa kuri jungiasi prie brokerio nenaudojant papildomos saugos mechanizmo. 24 pav. klientui prieš siunčiant skirtingo dydžio žinutes nematomas išskirtos atminties šuolis, prisijungiant prie MQTT brokerio programos atmintis neperžengė 8 kB išskirtos atminties ribos. Siunčiant žinutes matomi panašūs atminties išskyrimo bruožai kaip ir praeituose scenarijuose, didžiausios žinutės siuntimo metu sunaudota ~15.2 kB atminties.



25 pav. įprasto MQTT kliento su TLS atminties sunaudojimo grafikas

Paskutiniame scenarijuje naudojamas MQTT klientas su TLS protokolu. Atminties sunaudojimo grafike 25 pav. matomas skirtingas grafikas nei anksčiau naudoti. Kadangi „massif“ įrankis išsaugojo visas vietas, kuriose vyko atminties išskyrimo operacijos, matoma, jog prieš žinučių siuntimo operacijas buvo atlikta daugiau papildomo darbo, susijusio su autentifikacija TLS protokolo

lygmenyje. Kadangi ši dalis buvo atliekama automatiniu būdu su „mosquitto“ bibliotekos pagalba, daug išvadų apie atliktus veiksmus negalima daryti. Svarbu atsižvelgti jog grafiko horizontali ašis neišreiškia tolygaus laiko ėjimo, nors ir klientas prisijungė prie MQTT brokerio greičiau nei patobulintos saugos kliente, tačiau per tą laiką buvo atlikta daug daugiau vidinių operacijų, susijusių su TLS protokolu. Atliekant žinučių siuntimą matoma, jog bendras aplikacijos atminties sunaudojimas stabiliai kito dešimt kartų pagal didėjančias žinutes, tačiau bendras atminties sunaudojimas buvo žymiai didesnis nei kituose scenarijuose. Svarbu paminėti tai, jog atminties regionai, susiję su žinučių siuntimu nebuvo stipriai didesni, tačiau kitos programos dalys, susijusios su TLS protokolo palaikymu tuo tarpu užėmė didžiąją dalį bendros programos atminties.

5 lentelė. MQTT klientų programų laiko užtrukimo rezultatai

Eil. nr.	Programa	Matuotas laiko tarpas	Užtruktas laikas (ms)
1.	Patobulintos saugos metodo MQTT klientas su registracija	Kliento prisijungimas prie MQTT brokerio („CONNACK“)	4432.41
2.	Patobulintos saugos metodo MQTT klientas be registracijos	Kliento prisijungimas prie MQTT brokerio („CONNACK“)	196.52
3.	Įprastas MQTT klientas	Kliento prisijungimas prie MQTT brokerio („CONNACK“)	1.81
4.	Įprastas MQTT klientas su TLS	Kliento prisijungimas prie MQTT brokerio („CONNACK“)	54.92
5.	Patobulintos saugos metodo MQTT klientas su registracija	Vidutinis žinučių išsiuntimo laikas („PUBACK“)	0.35
6.	Patobulintos saugos metodo MQTT klientas be registracijos	Vidutinis žinučių išsiuntimo laikas („PUBACK“)	0.56
7.	Įprastas MQTT klientas	Vidutinis žinučių išsiuntimo laikas („PUBACK“)	2.38
8.	Įprastas MQTT klientas su TLS	Vidutinis žinučių išsiuntimo laikas („PUBACK”)	0.73
9.	Patobulintos saugos metodo MQTT klientas su registracija	Visos programos veikimo laikas („DISCONNECT“)	14483.82
10.	Patobulintos saugos metodo MQTT klientas be registracijos	Visos programos veikimo laikas („DISCONNECT“)	10029.82
11.	Įprastas MQTT klientas	Visos programos veikimo laikas („DISCONNECT“)	11202.70
12.	Įprastas MQTT klientas su TLS	Visos programos veikimo laikas („DISCONNECT“)	11034.48

5 lentelėje pateikiami skirtingų MQTT klientų programų veikimo laiko matavimai trijose pagrindinėse kategorijose: kliento prisijungimas prie brokerio („CONNACK“), vidutinis žinučių išsiuntimo laikas („PUBACK“) bei visas programos veikimo laikas iki atsijungimo („DISCONNECT“). Palyginus rezultatus, matyti tam tikrų netikėtų tendencijų. Buvo tikimasi, kad įprastas MQTT klientas veiks greičiausiai, o siūlomas patobulintos saugos metodo kliento programa užtruks ilgiau nei įprasto MQTT kliento, bet trumpiau nei naudojant TLS protokolą. Kaip ir tikėtasi, įprastas MQTT klientas prisijungia greičiausiai – vos per 1.81 ms. Naudojant TLS prisijungimo laikas išauga iki 54.92 ms, dėl papildomų kriptografinių skaičiavimų bei rankos paspaudimo proceso. Tuo tarpu patobulintos saugos kliento su registracija prisijungimo laikas yra 4432.41 ms – tai atspindi

papildomus autentifikacijos žingsnius kurie užtrunka MQTT kliento įrenginyje. Buvo pastebėta jog didžiąją dalį laiko užtrunka „Python“ programėlės veikimas kurio metu generuojamos PUF užklausų-atsakymų poros ir papildomai šis procesas kartojamas du kartus – pirmas kartas generuojant visas galimas, šiuo atveju, 64 užklausų-atsakymų poras, o antru atveju kai generuojama dalis užklausų-atsakymų porų norint išgauti OTP generavimo reikšmę. Be registracijos ši vertė sumažėja iki 196.52 ms, bet vis tiek išlieka didesnė nei TLS atveju kadangi reikalinga iš suvykdyti PUF generavimo programėlę. Lyginant žinučių išsiuntimo laiką pastebima netikėta situacija – nors patobulintos saugos metodo klientas siunčia užšifruotas žinutes, o MQTT brokeris realiuoju laiku jas iššifruoja naudodamas papildomą įskiepį, vidutinis PUBACK užtrukimas yra mažiausias iš visų klientų – netgi mažesnis už TLS ir įprastą klientą. Tai gali būti susiję su specifiniu brokerio konfigūravimu, virtualios aplinkos ypatumais ar įskiepio veikimu. Galutinis programos veikimo laikas nuo prisijungimo iki atsijungimo neatskleidžia ryškios loginės sekos. Pavyzdžiui, įprasto MQTT kliento veikimo trukmė siekia 11202.70 ms, nors tai teoriškai turėtų būti mažesnė nei papildomus veiksmus atliekančio patobulintos saugos kliento. Šie rezultatai rodo, kad šis rodiklis galėjo būti paveiktas papildomų aplinkos faktorių.

6 lentelė. Tinklo resursų sunaudojimo rezultatai

Eil. nr.	**Programa**	**Srauto kiekis iš kliento į MQTT brokerį (baitai)**	**Srauto kiekis iš MQTT brokerio į klientą (baitai)**
1.	Patobulintos saugos metodo MQTT klientas su registracija	30676	1693
2.	Patobulintos saugos metodo MQTT klientas be registracijos	30028	1235
3.	Įprastas MQTT klientas	25657	987
4.	Įprastas MQTT klientas su TLS	26692	5254

6 lentelėje pateikiami tinklo resursų sunaudojimo rezultatai. Šio tyrimo metu buvo leidžiamos visos programėlės iš naujo tuo pačiu tikrinant tinklo srauto veiklą naudojant „Wireshark“. Gavus ataskaitas buvo pasirinkta sekti TCP/IP lygmens pokalbis tarp kliento ir MQTT brokerio. Tai reiškia jog gauti rezultatai susideda tiek iš MQTT protokolo lygmens tiek iš TCP/IP protokolo bei TLS paskutiniame tyrimo variante. Iš gautų rezultatų matoma jog patobulintos saugos metodo MQTT klientas sunaudoja 648 ir 458 daugiau baitų atliekant registracijos žingsnį nei lyginant tą patį klientą be registracijos. Ši papildoma informacija susideda iš MQTT 5.0 „AUTH“ kontrolinių paketų. Kaip ir tikėtasi patobulintos saugos metodo MQTT klientas bendrai sunaudoja daugiau tinklo srauto duomenų siunčiant žinutes nei lyginant su įprastu MQTT klientu, tačiau iš rezultatų matoma jog ir kliento siunčiamų duomenų kiekis buvo didesni už MQTT klientą su TLS. Iš kitos pusės bendras sunaudotų duomenų kiekis nebuvo viršytas jei atsižvelgiama į sunaudotą srauto kiekį iš MQTT brokerio į klientą. Naudojant TLS šis kiekis stipriai išaugo. Tai yra todėl kadangi vykdoma papildoma komunikacija TLS protokolo lygyje kada atliekamas rankos paspaudimo žingsnis.

7 lentelė. MQTT klientų atminties sunaudojimo rezultatai

Eil. nr.	Programa	Atminties priskyrimo operacijų kiekis	Priskirtų baitų kiekis	Didžiausias atminties sunaudojimo kiekis (KiB)
1.	Patobulintos saugos metodo MQTT klientas su registracija	260	89,001	16,1
2.	Patobulintos saugos metodo MQTT klientas be registracijos	159	77,548	15,7
3.	Įprastas MQTT klientas	57	65,425	15,2
4.	Įprastas MQTT klientas su TLS	6522	911,451	200,1

Visi scenarijai vykdomi iš naujo paleidžiant programas su „valgrind-memcheck“ įrankiu, kurio pagalba matoma detalesnė informacija apie atminties sunaudojimą programos pabaigoje. Matoma jog patobulintos saugos metodo MQTT klientas su registracija sunaudojo daugiausiai atminties baitų ir atminties priskyrimo operacijų. Lyginant su scenarijumi be registracijos žingsnio, buvo suvykdyta 101 daugiau atminties priskyrimo operacijų bei priskirta 11,453 daugiau baitų.

8 lentelė. Patobulintos saugos metodo MQTT žinučių šifravimo greitaveikos rezultatai

Eil. nr.	Šifravimo algoritmas	Šifruojamos žinutės ilgis	Šifravimo vidutinis laikas (sekundės)
1	„AES-128-CBC“	7.5KB	0.082970
2	„AES-256-CBC“	7.5KB	0.107849
3	„AES-128-GCM“	7.5KB	0.049331
4	„AES-256-GCM“	7.5KB	0.051368
5	„DES-CBC“	7.5KB	0.955403
6	„3DES-CBC“	7.5KB	2.502073
7	„AES-128-CBC“	54KB	1.362100
8	„AES-256-CBC“	54KB	1.666550
9	„AES-128-GCM“	54KB	1.145062
10	„AES-256-GCM“	54KB	1.160736
11	„DES-CBC“	54KB	8.074314
12	„3DES-CBC“	54KB	19.198858
13	„AES-128-CBC“	228KB	16.997466
14	„AES-256-CBC“	228KB	17.330477
15	„AES-128-GCM“	228KB	15.046200
16	„AES-256-GCM“	228KB	15.483306
17	„DES-CBC“	228KB	40.663288
18	„3DES-CBC“	228KB	85.494747

Atlikus šifravimo našumo matavimus virtualioje mašinoje, buvo palyginti keli simetrinio šifravimo algoritmai – „AES“ („CBC“ ir „GCM“ režimai), „DES“ ir „3DES“ – šifruojant skirtingo dydžio (7.5 KB, 54 KB ir 228 KB) žinutes. Rezultatai, pavaizduoti 8 lentelėje rodo aiškius našumo skirtumus tarp skirtingų algoritmų bei režimų. „AES“ algoritmas pasižymi geriausiu našumu. „AES-GCM“ režimas buvo greitesnis nei „AES-CBC“. Tai gali būti paaiškinama tuo, jog „GCM“ režimas

leidžia efektyviau apdoroti duomenis bei turi įmontuotą autentifikavimą, todėl nereikia papildomų žingsnių saugumui užtikrinti. „AES-128“ šiek tiek greitesnis nei „AES-256“, kaip ir tikėtasi, dėl mažesnio rakto ilgio, tačiau skirtumas nėra labai didelis, ypač „GCM“ režime. „DES“ ir „3DES“ algoritmai – ženkliai lėtesni. Nors „DES“ šifravimas su mažomis žinutėmis (7.5 KB) truko mažiau nei sekunde, didėjant duomenų kiekiui, laikas stipriai išauga. Pvz., 228 KB žinutės šifravimas truko virš 40 sekundžių. „3DES“ šifravimo trukmė buvo dar ilgesnė – virš 85 sekundžių 228 KB žinutei, o tai dar kartą patvirtina, kad šis algoritmas nėra tinkamas naudoti ribotų resursų ar realaus laiko sistemose. Šifruojamos žinutės dydis daro tiesioginę įtaką šifravimo laikui - visų algoritmų atveju šifravimo laikas didėjo proporcingai žinutės dydžiui, tačiau efektyvesni algoritmai, ypač „AES-GCM“, pasižymėjo geresniu efektyvumu. Pavyzdžiui, „AES-128-GCM“ šifravimo laikas nuo 0.049 s (7.5 KB) iki 15.046 s (228 KB) parodė žemesnį augimo šuolį lyginant su „3DES“ ar „DES“.

4.4. Tyrimo išvados

1. „valgrind-memcheck“ ataskaitoje buvo pastebėta jog patobulintos saugos metodo MQTT kliento programos turėjo atminties nutekėjimo problemų. Dėl šių problemų buvo pastebėtas nestandartinis atminties sunaudojimas iš „valgrind-massif“ kadangi atminties sunaudojimas kilo visos programos veikimo metu net kai ir buvo vykdomos atminties išvalymo operacijos. Išsprendus šias klaidas testai buvo paleisti iš naujo iki kol nebeliko jokių prarastų atminties regionų programos veikimo pabaigoje.

2. Prototipas įgyvendintas sėkmingai – pagerintas MQTT protokolo saugumas atsižvelgiant į ribotų resursų įrenginius panaudojant vienkartinius slaptažodžius. Išnaudotos naujausios MQTT protokolo versijos funkcijos kurios leidžia išplėsti autentifikacijos mechanizmą nepažeidžiant standarto taisyklių. Prototipe MQTT klientas ir brokeris palaiko pagrindines funkcijas – įrenginių registravimą bei priregistruotų įrenginių autentifikaciją. Naudojant PUF toliau pagerinamas sistemos saugumas leidžiantis apsaugoti įrenginių jautrią informaciją net ir gaunant įrenginio fizinę prieigą. Siūlomas patobulintos saugos metodas autentifikacijos metu apkrauna įrenginį labiau nei įprasti MQTT klientai tačiau yra žymiai lengvesnis atsižvelgiant į sunaudotos atminties suvartojimą su klientais kurie naudoja TLS protokolą, tokiu padarant metodą tinkamu silpnesniems įrenginiams kurie nepalaiko TLS protokolo funkcijų.

3. Siekiant gauti tikslesnius ir objektyvesnius rezultatus, rekomenduojama eksperimentus atlikti realiose sistemose, kuriose veikia Linux operacinės sistemos, o ne virtualiose mašinose. Pavyzdžiui, MQTT klientų programas paleisti atskiruose įterptiniuose įrenginiuose, o MQTT brokerį talpinti dedikuotame fiziniame serveryje. Tokiu būdu būtų galima eliminuoti virtualizacijos poveikį ir tiksliau įvertinti kiekvieno kliento našumą realiomis sąlygomis laiko atžvilgiu. Taip pat, autentifikacijos proceso trukmę naudojant patobulintos saugos MQTT klientą galima optimizuoti tobulinant PUF Python kliento programos įgyvendinimą.

Išvados

1. Išlaikytas modulinis dizainas leidžia lanksčią integraciją. Siūlomas saugos sprendimas išlaiko MQTT protokolo modulinio dizaino principą – papildomas šifravimas taikomas tik tarp unikalaus kliento ir brokerio, o kiti esami funkcionalumai, tokie kaip MQTT tilto mechanizmas ar TLS protokolas, gali būti laisvai naudojami kaip papildomos, neprivalomos apsaugos priemonės. Tai leidžia sprendimą integruoti į esamas architektūras nesulaužant jų struktūros ir neužkertant kelio naudoti kitus, aukštesnio lygmens saugumo sluoksnius.
2. Efektyvumo ir saugumo balansas pasiektas. Naudojant siūlomą sprendimą, veikimo greitis yra ženkliai geresnis nei pilnai TLS protokolu paremtų klientų, tačiau tik šiek tiek mažesnis nei visai be šifravimo veikiančių sistemų. Šis kompromisas leidžia pasiekti balansą tarp saugumo lygio ir prietaisų apkrovos – tai ypač svarbu ribotų išteklių DI įrenginiams. Atminties sunaudojimas taip pat išlieka minimalus, todėl sprendimas yra tinkamas ir silpnesniems mikrovaldikliams.
3. MQTT 5.0 standartas suteikia galimybių kurti išplėstinius autentifikacijos metodus. Vienas iš MQTT 5.0 standarto privalumų yra naujai įtrauktas „AUTH“ kontrolinis paketas, kuris leidžia kurti ir integruoti nestandartinius, kelių žingsnių autentifikacijos metodus. Ši funkcija buvo panaudota siūlomame sprendime tam, kad būtų galima įgyvendinti dinaminį, kontekstinį autentifikavimą, kuris geriau atitinka DI aplinkos reikalavimus nei tradiciniai naudotojo–slaptažodžio modeliai.
4. PUF technologijos stiprina įrenginių saugumą. Naudojant PUF technologiją, išsprendžiama viena iš didžiausių DI saugumo problemų – statinių slaptų raktų laikymas įrenginiuose. PUF leidžia dinaminiu būdu sugeneruoti unikalias, iš klonuoti neįmanomas rakto reikšmes tiesiogiai iš įrenginio fizinių savybių, taip užtikrinant, kad net jei atmintis būtų pažeista ar nuskaityta, slaptas raktas nebūtų tiesiogiai prieinamas. Tai ženkliai sumažina riziką susijusią su rakto nutekėjimu ar atakomis prieš atmintį.
5. MQTT išlieka bendro naudojimo protokolu. Nepaisant siūlomų patobulinimų, MQTT protokolas išlieka lengvas, universalus ir plačiai taikomas įvairiose srityse. Šis sprendimas nekeičia esminių protokolo principų, todėl gali būti diegiamas ten, kur reikalingas standartinis suderinamumas ir sąveika tarp skirtingų gamintojų įrenginių.
6. Tradiciniai saugos sprendimai dažnai netinka ribotų išteklių DI aplinkai. Atlikta analizė parodė, kad daugelis esamų MQTT saugos mechanizmų nėra pilnai pritaikyti naudojimui DI aplinkoje. Vieni iš jų, pavyzdžiui, leidimas laisvai prenumeruoti temas ar paprasta slaptažodžio autentifikacija, kelia grėsmių saugumui. Tuo tarpu kiti, kaip TLS, nors ir labai saugūs, dažnai būna per daug apkraunantys mažos galios įrenginiams dėl sudėtingų kriptografinių operacijų. Tai rodo poreikį kurti tarpinius, specifinius DI sprendimus, kurie derina saugumą ir efektyvumą.

Literatūros sąrašas

[1] V. Hassija, V. Chamola, V. Saxena, D. Jain, P. Goyal, and B. Sikdar, "A Survey on IoT Security: Application Areas, Security Threats, and Solution Architectures," *IEEE Access*, vol. 7, pp. 82721–82743, 2019, doi: 10.1109/ACCESS.2019.2924045.

[2] R. Rajora, A. Rajora, B. Sharma, P. Aggarwal, and S. Thapliyal, "Unveiling the Synergy: IoT and Digital Twins in Transforming Industries," in *2023 3rd International Conference on Smart Generation Computing, Communication and Networking (SMART GENCON)*, Dec. 2023, pp. 1–5. doi: 10.1109/SMARTGENCON60755.2023.10442175.

[3] R. O. Andrade, S. G. Yoo, L. Tello-Oquendo, and I. Ortiz-Garcés, "A Comprehensive Study of the IoT Cybersecurity in Smart Cities," *IEEE Access*, vol. 8, pp. 228922–228941, 2020, doi: 10.1109/ACCESS.2020.3046442.

[4] F. Katulić, D. Sumina, S. Groš, and I. Erceg, "Protecting Modbus/TCP-Based Industrial Automation and Control Systems Using Message Authentication Codes," *IEEE Access*, vol. 11, pp. 47007–47023, 2023, doi: 10.1109/ACCESS.2023.3275443.

[5] C. Ian, "The Origin of MQTT," HiveMQ. Accessed: May 05, 2025. [Online]. Available: https://www.hivemq.com/blog/the-history-of-mqtt-part-1-the-origin/#heading-pre-standardization-1999-2012

[6] A. Banks, E. Briggs, K. Borgendale, and R. Gupta, *MQTT Version 5.0*, Mar. 07, 2019.

[7] H. W. van der Westhuizen and G. P. Hancke, "Practical Comparison between COAP and MQTT - Sensor to Server level," in *2018 Wireless Advanced (WiAd)*, Jun. 2018, pp. 1–6. doi: 10.1109/WIAD.2018.8588443.

[8] B. Mishra and A. Kertesz, "The Use of MQTT in M2M and IoT Systems: A Survey," *IEEE Access*, vol. 8, pp. 201071–201086, 2020, doi: 10.1109/ACCESS.2020.3035849.

[9] A. R. Alkhafajee, A. M. A. Al-Muqarm, A. H. Alwan, and Z. R. Mohammed, "Security and Performance Analysis of MQTT Protocol with TLS in IoT Networks," in *2021 4th International Iraqi Conference on Engineering Technology and Their Applications (IICETA)*, Sep. 2021, pp. 206–211. doi: 10.1109/IICETA51758.2021.9717495.

[10] E. Baranauskas and J. Toldinas, "Daiktų interneto MQTT protokolo saugos ir energijos sąnaudų tyrimas," 2019.

[11] E. B. Sanjuan, I. A. Cardiel, J. A. Cerrada, and C. Cerrada, "Message Queuing Telemetry Transport (MQTT) Security: A Cryptographic Smart Card Approach," *IEEE Access*, vol. 8, pp. 115051–115062, 2020, doi: 10.1109/ACCESS.2020.3003998.

[12] Ö. Şeker, G. Dalkılıç, and U. C. Çabuk, "MARAS: Mutual Authentication and Role-Based Authorization Scheme for Lightweight Internet of Things Applications," *Sensors*, vol. 23, no. 12, Art. no. 12, Jan. 2023, doi: 10.3390/s23125674.

[13] C.-S. Park and H.-M. Nam, "Security Architecture and Protocols for Secure MQTT-SN," *IEEE Access*, vol. 8, pp. 226422–226436, 2020, doi: 10.1109/ACCESS.2020.3045441.

[14] S. Shin and K. Kobara, "A Secure MQTT Framework from PUF-based Key Establishment," in *2017 International Conference on Computational Science and Computational Intelligence (CSCI)*, Las Vegas, NV, USA: IEEE, Dec. 2017, pp. 1296–1301. doi: 10.1109/CSCI.2017.339.

[15] M. Calabretta, R. Pecori, and L. Veltri, "A Token-based Protocol for Securing MQTT Communications," in *2018 26th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, Split: IEEE, Sep. 2018, pp. 1–6. doi: 10.23919/SOFTCOM.2018.8555834.

[16] Ö. Yerlikaya and G. Dalkılıç, "Authentication and Authorization Mechanism on Message Queue Telemetry Transport Protocol," in *2018 3rd International Conference on Computer Science and Engineering (UBMK)*, Sep. 2018, pp. 145–150. doi: 10.1109/UBMK.2018.8566599.

[17] D. Hästbacka, M. Tran, P. Kannisto, M. Filppula, and P. Varga, "External Token-Based Authorization of Data-Driven Integrations and Service Compositions in MQTT 5," in *IECON 2023-*

49th Annual Conference of the IEEE Industrial Electronics Society, Oct. 2023, pp. 1–6. doi: 10.1109/IECON51785.2023.10311779.

[18] A. Ali-Pour and J. Gascon-Samson, "PECMQ: Private and Encrypted Communications in IoT Systems Using PUFs and MQTT," in *2024 IEEE 44th International Conference on Distributed Computing Systems Workshops (ICDCSW)*, Jul. 2024, pp. 64–74. doi: 10.1109/ICDCSW63686.2024.00017.

[19] S. Babkin and A. Epishkina, "Authentication Protocols Based on One-Time Passwords," in *2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, Jan. 2019, pp. 1794–1798. doi: 10.1109/EIConRus.2019.8656839.

[20] F. Buccafurri, V. De Angelis, and R. Nardone, "Securing MQTT by Blockchain-Based OTP Authentication," *Sensors*, vol. 20, no. 7, Art. no. 7, Jan. 2020, doi: 10.3390/s20072002.

[21] H. Kim, J. Han, C. Park, and O. Yi, "Analysis of Vulnerabilities That Can Occur When Generating One-Time Password," *Appl. Sci.*, vol. 10, no. 8, Art. no. 8, Jan. 2020, doi: 10.3390/app10082961.

[22] K. Lounis and M. Zulkernine, "T2T-MAP: A PUF-Based Thing-to-Thing Mutual Authentication Protocol for IoT," *IEEE Access*, vol. 9, pp. 137384–137405, 2021, doi: 10.1109/ACCESS.2021.3117444.

[23] X. Gong, T. Feng, and M. Albettar, "PEASE: A PUF-Based Efficient Authentication and Session Establishment Protocol for Machine-to-Machine Communication in Industrial IoT," *Electronics*, vol. 11, no. 23, Art. no. 23, Jan. 2022, doi: 10.3390/electronics11233920.

[24] M. Bender, E. Kirdan, M.-O. Pahl, and G. Carle, "Open-Source MQTT Evaluation," in *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*, Jan. 2021, pp. 1–4. doi: 10.1109/CCNC49032.2021.9369499.

[25] D. M'Raihi, D. M'Raihi, F. Hoornaert, D. Naccache, M. Bellare, and O. Ranen, "HOTP: An HMAC-Based One-Time Password Algorithm," Internet Engineering Task Force, Request for Comments RFC 4226, Dec. 2005. doi: 10.17487/RFC4226.

[26] D. M'Raihi, S. Machani, M. Pei, and J. Rydell, "TOTP: Time-Based One-Time Password Algorithm," RFC Editor, RFC6238, May 2011. doi: 10.17487/rfc6238.