Kauno technologijos universitetas

Informatikos fakultetas

Saugus duomenų perdavimo metodas daiktų interneto įrenginiams

Baigiamasis magistro studijų projektas

Povilas Malakas

Projekto autorius

Prof. Egidijus Kazanavičius

Vadovas

Kaunas, 2024

Kauno technologijos universitetas

Informatikos fakultetas

Saugus duomenų perdavimo metodas daiktų interneto įrenginiams

Baigiamasis magistro projektas

Informacijos ir informacinių technologijų sauga (6211BX008)

Povilas Malakas

Projekto autorius

Prof. Egidijus Kazanavičius

Vadovas

Doc. Gedeiminas Činčikas

Recenzentas

Kaunas, 2024

Kauno technologijos universitetas

Informatikos fakultetas

Povilas Malakas

Saugus duomenų perdavimo metodas daiktų interneto įrenginiams

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.

Povilas Malakas

Patvirtinta elektroniniu būdu

Malakas Povilas. Saugus duomenų perdavimo metodas daiktų interneto įrenginiams. Magistro baigiamasis projektas / vadovas prof. Egidijus Kazanavičius; Kauno technologijos universitetas, Informatikos fakultetas.

Studijų kryptis ir sritis (studijų krypčių grupė): informatikos inžinerija.

Reikšminiai žodžiai: saugus duomenų perdavimo metodas daiktų interneto įrenginiams, ESP-BLE-MESH, ChaCha20-Poly1305.

Kaunas, 2024. 61 p.

Santrauka

Darbe pristatomas saugus duomenų perdavimo metodas daiktų interneto įrenginiams. Įrenginių prijungtų prie interneto kiekiui augant didėja ir duomenų, perduotų daiktų interneto tinkluose, kiekis. Saugūs duomenų perdavimo metodai yra receptyvūs sunaudojamos energijos atžvilgiu, o daiktų interneto įrenginiai pasižymi ribotais resursais. Darbo metu bus tiriama saugų duomenų perdavimą užtikrinančių metodų, kurie būtų mažiau receptyvūs energijos sunaudojimui, problema.

Darbo analizės dalyje apžvelgiamas daiktų interneto ir šios technologijos saugos aktualumas, problemos, tinklų architektūros, duomenų perdavimo metodai, ESP32 mikrovaldikliai daiktų interneto tinkluose, esami saugumo sprendimai ir palyginami daiktų interneto duomenų perdavimo protokolai. Sprendimo dalyje pateikiamas siūlomas daiktų interneto įrenginių ESP32 saugaus duomenų perdavimo metodas. Prototipo realizacijos skyriuje apžvelgiamas prototipas, daiktų interneto tinklo potinkliai ir konfigūracijos. Tyrimo dalyje pateikiamas kokybinis tinklų palyginimas ir eksperimentinis tyrimas įrenginių naudojamos energijos našumui palyginti. Darbo pabaigoje pateikiamos išvados.

Malakas Povilas. Secure Data Transmission Method for Internet of Things Devices. Master's Final Degree Project / supervisor prof. Egidijus Kazanavičius; Faculty of Informatics, Kaunas University of Technology.

Study field and area (study field group): Informatics Engineering.

Keywords: secure data transmission method for internet of things devices, ESP-BLE-MESH, ChaCha20-Poly1305.

Kaunas, 2024. 61.

Summary

The paper presents a secure data transmission method for Internet of Things devices. As the number of devices connected to the Internet grows, so does the amount of data transmitted over Internet of Things networks. Secure data transmission methods are receptive to energy consumption, and IoT devices are characterized by limited resources. The problem of methods ensuring secure data transmission, which would be less receptive to energy consumption, will be investigated.

The analysis part of the work reviews the relevance of the Internet of Things and its security, problems, network architectures, data transmission methods, ESP32 microcontrollers in Internet of Things networks, existing security solutions and compares Internet of Things data transmission protocols. The solution part presents the proposed ESP32 secure data transmission method for Internet of Things devices. The implementation section reviews the prototype, IoT network subnets and configurations. The research part provides a qualitative comparison of networks and an experimental study to compare the energy performance of devices. Conclusions are presented at the end of the work.

Turinys

Lentelių sąrašas

Paveikslų sąrašas

Santrumpų ir terminų sąrašas

Terminai:

Bluetooth SIG – „Bluetooth Special Interest Group“ yra standartų organizacija, kuri prižiūri „Bluetooth“ standartų kūrimą ir „Bluetooth“ technologijų bei prekių ženklų licencijavimą gamintojams.

PDU (angl. *Protocol Data Unit*) – protokolo duomenų vienetas yra vienas informacijos vienetas, perduodamas tarp lygiaverčių kompiuterių tinklo objektų. Jį sudaro su protokolu susijusi valdymo informacija ir vartotojo duomenys.

TransMIC (angl. *The Message Integrity Check for Transport*) – 32 arba 64 bitų laukas transportavimo sluoksnio pranešimo vientisumui patikrinti.

Nonce - Kriptografijoje nonce yra savavališkas skaičius, kurį galima naudoti tik vieną kartą kriptografiniam ryšiui.

API (angl. *Application Programming Interface*) – taikomųjų programų programavimo sąsaja.

Įvadas

Nepaisant pasaulį sukrėtusių dalykų, kurie paveikė produktus ir tiekimo grandines Daiktų Internetas (DI toliau) nenustojamai auga ir plečiasi. Tikimasi, kad iki 2025 m. prijungtų DI įrenginių bus maždaug 27 mlrd. [24]. Ši technologija veikia prijungdama įvairius dalykus prie interneto, pvz. renka jutiklių surinktus duomenis ir leidžia nuotoliniu būdu valdyti bei stebėti aplinką. Tai leidžia DI pritaikyti įvairiose srityse siekiant piniginės ar asmeninės naudos.

Įrenginių prijungtų prie interneto kiekiui augant ir populiarėjant daugėja atvejų kada duomenys gali būti nutekinti vykstant prietaisų komukacijai tarpusavyje. Siekant apsaugoti tinklą reikia įdiegti saugų duomenų perdavimo metodą, tačiau duomenų saugumą užtikrinantys protokolai yra imlūs energijai. Darbo metu bus tiriama saugų duomenų perdavimą užtikrinančių metodų, kurie būtų mažiau receptyvūs energijos sunaudojimui, problema.

Darbo tikslas - sukurti saugų ir taupų sunaudojamai energijai metodą duomenų perdavimui DI įrenginių ESP32 tinkle. Tikslui pasiekti išsikelti uždaviniai:

1. Išanalizuoti DI tinklų architektūras, ESP32 tinklo protokolus ir saugumo problemas;
2. Pasiūlyti saugų duomenų perdavimo protokolą;
3. Praktiškai realizuoti DI įrenginių tinklą, naudojantį pasiūlytą saugų duomenų perdavimo protokolą;
4. Atlikti eksperimentinį tyrimą šio metodo energijos efektyvumui įvertinti;

Darbas sudarytas iš analizės, sprendimo, prototipo realizacijos ir tyrimo skyrių. Analizės dalyje apžvelgiamas daiktų interneto ir šios technologijos saugos aktualumas, problemos, tinklų architektūros, duomenų perdavimo metodai, ESP32 mikrovaldikliai daiktų interneto tinkluose, esami saugumo sprendimai ir palyginami daiktų interneto duomenų perdavimo protokolai. Sprendimo dalyje pristatomas siūlomas daiktų interneto įrenginių ESP32 saugaus duomenų perdavimo metodas, apžvelgiami reikalingi atlikti pakeitimai. Realizacijos skyriuje apžvelgiamas daiktų interneto tinklo prototipas, tinklo potinkliai ir konfigūracijos. Tyrimo dalyje pateikiamas kokybinis tinklų palyginimas ir eksperimentai įrenginių naudojamos energijos našumui palyginti. Darbo pabaigoje pateikiamos darbo eigoje prieitos išvados.

1. Daiktų interneto aktualumas

Daiktų interneto (toliau DI) rinka sparčiai plėtėsi per pastaruosius keletą metų po padidėjusios paklausos įvairių prietaisų komunikacija ir valdymu. Pagrindinis reikalavimas, taikomas šiuolaikiniams DI įrenginiams, yra suteikti veiksmingą ryšį, kad būtų užtikrintas patikimas nuotolinis valdymas ir duomenų perdavimas belaidėje aplinkoje.

DI technologija daro didelę įtaką žmonių elgesiui ir gyvenimo būdui tiek darbe, tiek buityje. Pažangios komunikacijos galimybės gerokai pakeičia pramonės automatizavimo ir gamybos, verslo ir procesų valdymo, intelektualių transporto sistemų, logistikos ir kt. savybes ir veikimą. Kasdieniams vartotojams DI suteikia išmaniuosius namus ir pristato naujas komunikacija paremtas technologijas kaip domotika, pagalbinis gyvenimas, e. sveikata, el. mokymasis ir kt.

1.1. Daiktų interneto tinklų saugos aktualumas

DI tinklo saugumas yra esminis plačiai paplitusios DI technologijos aspektas. Vis daugiau įrenginių jungiasi prie interneto ir vienas prie kito, todėl saugumo pažeidimų ir duomenų vagysčių rizika didėja. Todėl labai svarbu suprasti su DI tinklais susijusią saugumo riziką ir imtis veiksmų jai sumažinti.

Vienas iš pagrindinių dalykų, lemiančių DI tinklo saugumo svarbą, yra jautri informacija, kuri dažnai perduodama šiais tinklais. DI įrenginiai naudojami duomenims iš įvairių šaltinių rinkti ir perduoti, įskaitant asmeninę informaciją, finansinius duomenis ir pramonės valdymo sistemų informaciją. Jei šiuos duomenis perims ar galės prieiti neįgaliotos šalys, tai gali turėti rimtų pasekmių, tokių kaip finansiniai nuostoliai, duomenų pažeidimai ir net fizinė įrangos žala.

Kita DI tinklo saugumo svarbos priežastis yra neteisėtos prieigos prie pačių įrenginių rizika. DI įrenginiai dažnai yra prijungti prie interneto ir yra pasiekiami iš bet kurios pasaulio vietos. Jei užpuolikas gali gauti prieigą prie vieno iš šių įrenginių, jis gali jį panaudoti tolimesnėms atakoms pradėti arba pasiekti kitas tinklo dalis.

Labai svarbu suprasti su DI tinklais susijusias saugumo grėsmes ir stengtis sumažinti kylančią riziką. Didėjant prijungtų įrenginių skaičiui, svarbu įdiegti tinkamas saugumo priemones, skirtas apsaugoti asmens duomenis, įmonės duomenis ir užkirsti kelią neteisėtai prieigai prie įrenginių ir tinklo.

1.1.1. Daiktų interneto saugumo problemos

Be įprastų saugumo spragų, kurios veikia ne tik DI tinklus, tokių kaip tinkamos saugos konfigūracijos, priežiūros ir tinkamo tinklo segmentavimo trūkumas, silpni arba lengvai atspėjami slaptažodžiai, nepakankamas stebėjimas ir sistemos įvykių registravimas galima išskirti šias problemas, veikiančias DI tinklų įrenginius:

- Nesaugus ryšys: DI įrenginiai gali naudoti nešifruotus arba prastai užšifruotus ryšio protokolus, kurie gali leisti užpuolikams perimti ir nuskaityti tinkle perduodamus slaptus duomenis.
- Saugumo trūkumas programinės aparatinės įrangos atnaujinimų metu: tai gali leisti užpuolikams įrenginyje įdiegti kenkėjišką programinę įrangą, kuri gali suteikti jiems nuolatinę prieigą prie įrenginio ir tinklo.

– Tinkamos prieigos kontrolės trūkumas: netinkamai sukonfigūruoti prieigos kontrolės mechanizmai gali leisti neteisėtai pasiekti įrenginį ir taip kompromituoti įrenginį ir tinklą.

Saugaus ryšio neužtikrinimas gali sukelti rimtų saugumo problemų, įskaitant duomenų pažeidimus, neteisėtą prieigą ir teisinių nuostatų nesilaikymą. Svarbu užtikrinti, kad DI tinkle naudojami ryšio protokolai būtų saugūs, tačiau dažnai saugumą užtikrinantys protokolai gali būti nepaprastai sekinantys ribotų išteklių įrenginiams.

1.2. Daiktų interneto tinklų architektūra

DI tinklo architektūra reiškia bendrą įrenginių, protokolų ir komunikacijos mechanizmų, sudarančių DI sistemą, struktūrą ir organizavimą. DI tinklo architektūrą paprastai sudaro keli pagrindiniai komponentai:

– DI įrenginiai: fiziniai įrenginiai, sudarantys DI tinklą, pvz., jutikliai, pavaros ir tinklų sąsajos. Jie renka ir perduoda duomenis, taip pat gali gauti komandas ir valdyti kitus įrenginius.
– Tinklo sąsajos: įrenginiai, kurie veikia kaip tiltas tarp DI įrenginių ir likusio tinklo. Paprastai jie turi daugiau apdorojimo galios ir saugyklos nei DI įrenginiai ir yra atsakingi už duomenų rinkimą iš įrenginių, vietinį apdorojimą ir duomenų persiuntimą į debesį ar kitas galines sistemas.
– Debesis arba galinės sistemos: servisai, kuriuose saugomi ir apdorojami DI įrenginių surinkti duomenys. Tai gali būti duomenų bazės, duomenų analizės platformos ir kitos programos, kurios naudoja duomenis įvairioms funkcijoms atlikti.
– Tinklo protokolai: protokolai, naudojami prietaisams, tinklo sąsajoms ir galinėms sistemoms sujungti vienas su kitu. Juose gali būti laidinių ir belaidžių technologijų, tokių kaip Wi-Fi, *Bluetooth*, *Zigbee*, *MQTT* ir kt.
– Sauga ir valdymas: tai apima saugos protokolus, prieigos kontrolės mechanizmus ir valdymo įrankius, kurie naudojami DI tinklui apsaugoti ir sklandžiam veikimui užtikrinti.

DI tinklo architektūra sukurta taip, kad skirtingi įrenginiai, tinklo sąsajos, debesų sistemos ir protokolai galėtų bendrauti ir dirbti kartu, tuo pačiu užtikrinant tinklo ir jo komponentų saugumą. Architektūra gali skirtis priklausomai nuo konkretaus naudojimo atvejo ir tinklo topologijos pasirinkimo. Toliau apžvelgsime populiarias DI tinklų topologijas.

1.2.1. Daiktų interneto tinklo topologija

Egzistuoja kelių tipų DI tinklo topologijos, skirtos tinklui kurti, iš kurių labiausiai paplitusios yra tinklelio (angl. *mesh*), žvaigždės (angl. *star*), taškas į tašką (angl. *point-to-point*) ir hibridinė (angl. *hybrid*). Priklausomai nuo kuriamo tinklo specifikos ir tikslo pasirenkama point-to-point skirtinga tinklo topologija. Taškas į tašką topologija skirta dviejų įrenginių komunikacijai tarpusavyje, kuriai vykstant duomenų srautas gali keliauti viena kryptimi arba į abi puses priklausomai nuo konfigūracijos. Detaliau apžvelgiamos tik tinklelio, žvaigždės ir hibridinė topologijos, kadangi darbe bus kuriamas DI tinklas iš daugiau nei dviejų įrenginių.

1.2.1.1. Tinklelio tinklo topologija

Tinklelio topologijoje mazgai yra visiškai sujungti vienas su kitu per tam skirtą ryšį, per kurį informacija keliauja iš mazgo į mazgą. Tinklelio topologijoje yra $n(n-1)/2$ ryšių, jei yra n mazgų. Pramonės standartai, pagrįsti tinklelio topologija, apima *ZigBee, Z-Wave* ir *Thread*.

Pagrindinis tinklelio tinklo topologijos pranašumas yra tai, kad jis turi mažą perdavimo galią ir trumpesnius ryšius (< 30 metrų), o tai užtikrina gana ilgą baterijos veikimo laiką ir leidžia perkelti daug duomenų tinkle.

Pagrindinis tinklelio tipo topologijos trūkumas yra tas, kad diapazonas tarp dviejų tinklo mazgų yra gana ribotas. Tai reiškia, kad į tinklą gali tekti pridėti papildomų mazgų, kurie nėra griežtai būtini, pvz., papildomą DI įrenginį su tam tikru davikliu, tik tam, kad galėtumėte palaikyti ryšį tinkle. Be to, dėl susieto tinklelio tinklo pobūdžio, jei mazgas, esantis pozicijoje, kuri užtikrina ryšį tinkle, nėra aktyvus arba negali perduoti duomenų, tinklas gali sugesti.

1.2.1.2. Žvaigždės tinklo topologija

Žvaigždės topologijoje mazgai yra prijungti prie centrinio šakotuvo arba maršrutizatoriaus, kuriame informacija keliauja iš centrinio šakotuvo arba maršrutizatoriaus į visus mazgus. Žvaigždės topologijoje yra n ryšių, jei yra n mazgų. Mazgai yra visiškai sujungti su pagrindiniu mazgu per tam skirtą ryšio kanalą, kuriame informacija keliauja tarp mazgų.

Žvaigždės topologijos pranašumas yra tas, kad visas tinklo sudėtingumas nukreipiamas į centrinį mazgą, todėl visi kiti mazgai turi bendrauti tik savo laiko arba dažnio tarpsniu. Tai, kaip jie bendrauja, priklauso nuo to, ar belaidis tankinimas atliekamas naudojant dažnio padalijimo daugybinę prieigą, daugkartinę laiko dalijimosi prieigą ar daugybinę kodo dalijimosi prieigą.

Pagrindinis žvaigždės tinklo topologijos trūkumas yra tas, kad radijo ryšys tarp tinklo sąsajos ir galinio mazgo arba terminalo gali būti labai ilgas, o tai reiškia, kad kuo toliau mazgas yra nutolęs nuo tinklo sąsajos, tuo daugiau energijos reikia pranešimui perduoti. Tačiau skirtingai nei tinklelio mazgas, kuris turi būti nuolat aktyvus, žvaigždės topologijos mazgai gali ilsėtis tarp pranešimų perdavimo ir padeda sutaupyti bendrą kiekvieno mazgo išeikvojamą energijos kiekį.

1.2.1.3. Hibridinė tinklo topologija

Tai yra dviejų pirmiau minėtų topologijų derinys. Tai žvaigždės ir tinklelio topologijų derinys, kai kurie įrenginiai yra prijungti prie centrinio šakotuvo arba mazgo, o kiti įrenginiai yra sujungti vienas su kitu tinklelio tinkle. Ši topologija gali būti lankstesnė ir turi privalumų, lyginant su kitomis topologijomis. Ši topologija leidžia lengvai bendrauti tarp įrenginių ir sukurti tvirtesnį tinklą, kuris gali toliau veikti, net jei vienas įrenginys sugenda.

Pagrindinis hibridinių topologijų pranašumas yra jų lankstumas. Konkrečiai, hibridinės topologijos gali derinti skirtingų tipų tinklus, kad atitiktų specifinius delsos, mastelio keitimo ir energijos vartojimo efektyvumo reikalavimus. Be to, hibridinės topologijos paprastai yra labai patikimos. Taip yra todėl, kad gana lengva aptikti ir išskirti gedimus nepažeidžiant likusių hibridinio tinklo dalių. Taip pat hibridiniai tinklai yra labai keičiamo dydžio, nes nauji komponentai gali būti įtraukti į segmentą iš esmės nepažeidžiant kitų segmentų veiklos.

Hibridinius tinklus sudėtinga projektuoti ir paruošti naudojimui. Tokių tinklų projektavimas reikalauja didelių pastangų ir didesnių diegimo išlaidų. Daugeliu atvejų hibridiniams tinklams reikalingos brangios kapitalo investicijos į tokius elementus kaip išmanieji centrai, aušinimo infrastruktūra ir sudėtingi tinklo įrenginiai. Apskritai hibridinės topologijos yra daugiau tinkamos didelio masto diegimams ir mažiau – paprastesniems, mažesnio masto projektams.

1.2.1.4. Daiktų interneto tinklo topologijos pasirinkimas

Nuo tinklo topologijos pasirinkimo priklauso viso tinklo našumas, įrenginių sunaudojamos energijos kiekis, informacijos perdavimo vėlavimas ir paketų praradimas. I. M. Reza Permana ir kitų autorių 2019 m. moksliniame darbe išmaniųjų gaisro signalizacijų tobulinimui buvo atlikta tinklelio ir žvaigždės topologijų lyginamoji analizė. Tyrimas parodė, kad tinklelio topologijos tinklas yra jautresnis, atsparesnis gedimams ir prarado mažiau duomenų paketų, dėl to gali pateikti tikslesnius duomenis [5].

Atlikus energijos suvartojimo našumo testą, žvaigždės topologijos tinklas užtikrino iki dviejų kartų mažiau energijos sunaudojimo negu tinklelio topologijos tinklas. Visgi, darbe nepatikslinta koks ryšio protokolas buvo naudojamas tarp įrenginių. Ryšio protokolo pasirinkimas gali lemti ir tinklo topologijos pasirinkimą bei suvartojamą energijos kiekį, tinklo saugumą ir duomenų perdavimo efektyvumą. Toliau apžvelgsime dažniausiai naudojamus DI komunikavimo protokolus.

1.2.2. Daiktų interneto duomenų perdavimo protokolai

Svarbiausia renkantis duomenų perdavimo protokolus – susiaurinti savo reikalavimus, kad būtų galima sutelkti dėmesį tik į perspektyvias technologijas. Saugumas, veikimo diapazonas, duomenų perdavimo greitis, energijos suvartojimas ir kaina yra pagrindiniai kriterijai renkantis belaidį komunikacijos protokolų rinkinį. DI tinkluose yra daug komunikacijos protokolų rinkinių, kurie gali būti skirstomi į 5 grupes pagal savo savybes:

- P2P (angl. *peer-to-peer*) technologijos;
- vietinio tinklo (angl. *local area network* (LAN)) technologijos;
- tolimų nuotolių korinio ryšio (angl. *Long-distance cellular*) technologijos;
- mažos galios tolimojo susisiekimo (angl. *Low-power Long-distance*) technologijos;
- trumpo atstumo tinklelio tinklo (angl. *short range mesh*) technologijos.

P2P technologijos leidžia tiesiogiai bendrauti dviems įrenginiams, kurie yra sujungti kartu. Protokolų rinkiniai kaip *Bluetooth Classic*, *WiFi Direct*, NFC sudaro šią grupę. Paprastai, tik du įrenginiai gali dalyvauti *peer-to-peer* ryšyje, todėl šios grupes ryšio protokolų toliau nenagrinėsime.

1.2.2.1. Vietinio tinklo (LAN) technologijos

Vietinio tinklo komunikacijos technologijos, tokios kaip vietinis belaidis tinklas naudojamos, kai gaminiams reikalinga prieiga prie interneto su sąlyga, kad jis visada bus naudojamas šalia vietinio belaidžio tinklo prieigos taško. Vietinis belaidis tinklas yra žinomas kaip vietinio tinklo technologija dėl savo vidutinės aprėpties zonos. Ši ryšio technologija yra greita, pigi ir lengvai įgyvendinama, turi gerą veikimo diapazoną ir yra plačiai prieinama, tačiau didžiausias trūkumas, bent jau mobiliesiems įrenginiams, yra energijos suvartojimas. Dėl didelio energijos kiekio suvartojimo dažniausiai geriau naudoti kitas belaides technologijas, jeigu kuriamas tinklas nereikalauja didelio našumo.

1.2.2.2. Tolimų nuotolių korinio ryšio technologijos

Įrenginiams, kuriems reikia prieigos prie debesies, tačiau nuolat šalia neturės vietinio belaidžio tinklo prieigos taško, greičiausiai reikės tolimų nuotolių korinio radijo ryšio. Reikalingas korinio ryšio technologijos tipas priklauso nuo to, kaip greitai reikia perduoti duomenis, ir kiek mažiau nuo to, kur įrenginys bus naudojamas.

Ilgą laiką GSM (angl. *Global System for Mobile Communication*) kartu su GPRS (angl. *General Packet Radio Service*) duomenims perduoti buvo dažniausiai naudojama korinio ryšio technologija gaminiams, kuriems nereikia didelio duomenų perdavimo kiekio. Taip pat ši technologija yra populiari dėl didelio GSM / GPRS aparatinės įrangos prieinamumo ir santykinai mažų techninės įrangos sąnaudų.

Tačiau dauguma mobiliojo ryšio operatorių visame pasaulyje jau laipsniškai atsisakė GSM arba netrukus tai padarys, kad būtų galima padidinti pralaidumą 4G ir 5G išmaniesiems telefonams, kuriems reikalingas didžiulis duomenų perdavimo kiekis. Daugeliui gaminių LTE-M yra geriausia moderni alternatyva GSM programoms, kurioms taikomi vidutiniškai maži duomenų perdavimo spartos reikalavimai. Tuo tarpu gaminiams, kuriems reikalingas didelis duomenų perdavimo greitis, LTE (angl. *Long Term Evolution*) greičiausiai yra geriausia technologija.

LTE yra 4G korinio ryšio technologija, kuri palaiko daug didesnį duomenų greitį nei GSM. Ši technologija yra pagrįsta paketų komutavimu, o tai reiškia, kad duomenys perduodami mažais paketais, o ne tam skirtu ryšiu. Tai leidžia efektyviau naudoti tinklo išteklius ir įgalina didesnį duomenų perdavimo greitį. Tačiau LTE moduliai yra brangesni palygus su GSM įterptiniais moduliais. LTE operatorių paslaugų kaina taip pat yra žymiai didesnė nei GSM. Keliolikos įrenginių tinklo sudarymo kaina būtų svariai didesnė negu naudojant kitus duomenų perdavimo protokolus.

Žvelgiant iš saugumo pusės duomenys perduodami 5G, LTE ir 4G ryšiais yra užšifruoti, tapatybė autentifikuota ir apsaugota, taip pat naudojamasi jau sukurta ir gerai prižiūrima infrastruktūra. Apie aptiktas saugos spragas pranešama atitinkamoms telekomunikacijų įmonėms, tokioms kaip 3-osios kartos partnerystės projektas (3GPP) ir GSM asociacija, kurios imasi reikiamų priemonių 5G, 4G ir LTE saugumui gerinti. Korinio ryšio protokolų saugumas pažeidžiamas daug rečiau negu vietinių tinklų komunikacijos protokolų [12].

1.2.2.3. Mažos galios tolimojo susisiekimo technologijos

Šiais komunikacijos protokolais paremti tinklai paprastai vadinami LPWAN (angl. *low-power wide area network*) arba mažos galios plataus masto tinklais. Protokolai užtikrina didelį veikimo diapozoną, mažus energijos kaštus, tačiau negali pasigirti duomenų perdavimo greičiu.

SigFox yra mažos galios technologija, skirta belaidžiam ryšiui su įvairiais mažai energijos naudojančiais objektais, pvz., jutikliais ir mašina prie mašinos (angl. *machine to machine*) M2M programomis. Tai leidžia transportuoti nedidelius duomenų kiekius iki 50 kilometrų. *SigFox* naudoja UNB (angl. *Ultra Narrow Band*) technologiją. Ši technologija skirta tik mažam duomenų perdavimo greičiui nuo 10 iki 1000 bitų per sekundę ir gali veikti su maža baterija. *SigFox* palaiko žvaigždės tinklo topologiją [7].

LoRa (*Long-Range* trumpinys) kai kuriose srityse leidžia palaikyti ryšį daugiau nei 9 km, tuo pačiu sunaudojant mažai energijos. Tai patentuota belaidė technologija, kurią Semtech įsigijo 2012 m. LoRa naudoja įvairias dažnių juostas, priklausomai nuo veikimo regiono. Šiaurės Amerikoje naudojamas 915 MHz, o Europoje dažnis yra 868 MHz. Kitos sritys taip pat gali naudoti 169 MHz ir 433 MHz dažnius. LoRa nurodo pagrindinę technologiją ir gali būti tiesiogiai naudojama P2P ryšiams. LoRaWAN nurodo viršutinio lygmens tinklo protokolą. Norint, kad įrenginys galėtų prisijungti prie esamo LoRaWAN tinklo reikalingas brangesnis modulis su įtrauktu tinklo sluoksnio palaikymu. LoRa nesinaudoja korinio ryšio protokolais dėl to yra pigesnė alternatyva užtikrinanti veikimą dideliame diapozone, tačiau aprėptis yra ribota palyginti su korinio ryšio technologijomis. Įrenginių prisijungimas prie interneto įmanomas tik tinklo topologijoje turint tinklo sąsają užtikrinantį mazgą, kuris prisijungia prie interneto ir palaiko ryšį su visais nuotoliniais LoRa įrenginiais. Visa komunikacija interneto protokolais vyksta per tinklo sąsajos mazgą.

Skirtingai nuo LoRa / LoRaWAN, NB-IOT (angl. *Narrowband Internet of Things*) yra 4G korinio ryšio technologija, vienas iš LTE pogrupių dar žinomų kaip *LTE Cat-NB1*. Tai reiškia, kad šis protokolų rinkinys yra sudėtingesnis, brangesnis ir sunaudoja daugiau energijos, tačiau leidžia korinio ryšio dažnių juostomis sujungti platų spektrą įrenginių ir paslaugų. Ši technologija fokusuojasi į programas, kurioms reikalinga didelė paslaugų kokybė (angl. *Quality of Service* (QoS)), mažas delsimas ir tiesioginė prieiga prie interneto [9]. Korinio ryšio technologijų saugumo aspektas buvo apžvelgtas anksčiau 1.2.2.2 skyriuje.

Tolimojo ryšio prieiga su didesniu duomenų perdavimo greičiu negu LoRa arba NB-IOT palaiko gali suteikti LTE-M. LTE-M yra *LTE Cat-M1* santrumpa. Ši technologija skirta DI įrenginiams, kuriems reikia tiesiogiai prisijungti prie 4G mobiliojo ryšio tinklo. Tai LTE korinio ryšio technologijos pogrupis, optimizuotas mažos duomenų perdavimo spartos įrenginiams, veikiantiems iš mažų baterijų. LTE-M skiriasi nuo standartinio LTE keliomis svarbiomis savybėmis. Pirma, LTE-M aparatinė įranga yra pigesnė, nes galima naudoti paprastesnius lustus, kurių užtenka ribotesniam pralaidumui. Antra, šis protokolų rinkinys yra optimizuotas siekiant sumažinti energijos suvartojimą, kad greitai neišsikrautų mažos baterijos. Galiausiai, korinio ryšio paslaugų palaikymo išlaidos yra žymiai mažesnės, nes naudojamas pralaidumas neviršija standartinės LTE ribos.

MQTT (angl. *Message Queuing Telemetry Transport*) yra lengvas, publikavimo-prenumeratos (angl. *publish-subscribe*) modeliu pagrįstas ryšio protokolas, skirtas naudoti mažos galio įrenginiuose ir tinkluose su ribotu pralaidumu. Dėl savo energijos taupumo ir saugumo savybių yra plačiai naudojamas DI tinkluose.

Pagrindinė MQTT ypatybė – publikavimo-prenumeratos modelis, leidžiantis įrenginiams siųsti ir gauti tik jiems reikalingus duomenis, taip sumažinant perduodamų ir gaunamų duomenų kiekį. Tai gali padėti sumažinti įrenginių energijos suvartojimą ir padaryti tinklą efektyvesnį. MQTT naudoja brokeriu pagrįstą architektūrą, kurioje centrinis serveris, žinomas kaip brokeris, valdo ryšį tarp įrenginių. MQTT palaiko pramonės standartinius šifravimo metodus, tokius kaip TLS (angl. *Transport Layer Security*) [3], todėl jis puikiai tinka naudoti DI tinkluose, kuriuose ypatingai svarbu užtikrinti tinklo saugumą.

1.2.2.4. Trumpo atstumo tinklelio tinklo technologijos

Yra keturios dažniausiai naudojamos technologijos skirtos mažos galios, mažai duomenų naudojančiam tinklui sukurti: *Bluetooth Low-Energy* (BLE), *Zigbee, Z-Wave* ir *6LoWPAN*. Šie komunikacijos protokolų rinkiniai skirti įrenginiams su ribota baterijos energija, kuriems reikia nedidelius duomenų kiekius perduoti nedideliu atstumu. Pagrindinė funkcija, kurią palaiko visos keturios technologijos, vadinama tinklelio tinklu 1.2.1.1. Įrenginiams nereikia duomenų perduoti dideliu atstumu, todėl kiekvienam įrenginiui reikalinga galia yra daug mažesnė. Tai leidžia įrenginiams sumažinti energijos suvartojimo kaštus.

BLE taip pat žinomas kaip *Bluetooth smart* arba *Bluetooth 5*, kuris yra svarbus protokolas DI programose. Kurtas ir tobulintas mažo diapazono, mažo pralaidumo ir mažos delsos DI programoms užtikrinti. BLE pranašumai, palyginti su klasikiniu *Bluetooth*, apima mažesnį energijos suvartojimą, trumpesnį sąrankos laiką ir žvaigždžių tinklo topologijos palaikymą su neribotu mazgų skaičiumi.

Bluetooth 5 siūlo keturis skirtingus duomenų perdavimo greičius, kad atitiktų įvairius perdavimo diapazonus: 2Mbps, 1Mbps, 500kbps, 125kbps. BLE suteikiamas lankstumas duomenų perdavimo spartoje leidžia mažos galios gaminiams siųsti dar sudėtingesnius duomenis galutiniam vartotojui ir pritaikyti šį duomenų perdavimo protokolų rinkinį įvairios paskirties aplikacijom [11]. Energijos suvartojimas gali siekti dešimt kartų mažiau nei klasikinis *Bluetooth*, o tinklo delsos laikas gali būti trumpesnis iki 15 kartų [16]. Tai lengviausiai įdiegiama belaidė technologija, ji sunaudoja labai mažai energijos ir yra plačiausiai palaikoma.

ZigBee Alliance sukūrė *ZigBee* protokolą, pagrįstą mažos galios belaidžių IEEE802.15.4 tinklų standartu. *ZigBee* sukurtas kaip standartas, tinkantis aukšto lygio nebrangiems ryšio protokolams, kuriant asmeninius tinklus iš mažo dydžio, mažos galios skaitmeninių radijo imtuvų, perduodančių duomenis didesniais atstumais, tuo pat metu jis naudojamas programose, kurioms aktualu mažas duomenų perdavimo greitis, ilgesnis baterijos veikimo laikas ir saugūs tinklo įrenginiai. Be to, *ZigBee* gali palaikyti įvairių tipų topologijas, tokias kaip žvaigždžės, medžio ir tinklelio. *ZigBee* saugos architektūra papildo arba pagerina IEEE 802.15.4 standarto sluoksnių saugą. *ZigBee* standartas apibrėžia dviejų tipų simetrinius raktus, kurių kiekvienas yra 128 bitų ilgio, naudojamas šifruotam ryšiui [27].

Z-Wave yra patentuota belaidė technologija (2018 m. įsigyta *Silicon Labs*), kuri namų automatikos rinkoje pirmiausia konkuruoja su anksčiau apžvelgtais *ZigBee* ir BLE. Skirtingai nuo BLE ir *ZigBee*, kurie naudoja populiarią 2,4 GHz dažnių juostą, *Z-Wave* naudoja žemesnes 1 GHz juostas. Tiksli juosta įvairiose šalyse skiriasi, o tai gali sukelti komplikacijų plačiam šio protokolų rinkinio naudojimui. JAV *Z-Wave* veikia 908 MHz dažniu, o Europoje – 868 MHz. Kitos šalys ir regionai naudoja nuo 865 MHz iki 921 MHz dažnių juostas. Yra du reikšmingi žemesnio dažnio pranašumai: didesnis diapazonas ir mažesni trukdžiai. Žemesnio dažnio radijo bangos sklinda toliau. *Z-Wave* naudojamos dažnių juostos paprastai yra daug mažiau perpildytos, tačiau duomenų perdavimas yra daug kartų lėtesnis nei *Bluetooth 5*. Kaip ir *ZigBee* protokolų rinkinys *Z-Wave* naudoja išplėstinio šifravimo standartą AES (angl. *Advanced Encryption Standard*) simetrinio bloko šifravimo algoritmą naudojant 128 bitų rakto ilgį [28].

1.2.2.5. Duomenų perdavimo protokolų rinkinių lyginamoji analizė

DI tinkle yra daug belaidžių technologijų, kiekviena iš jų turi tam tikrų specifikacijų ir privalumų. Tinkamo protokolo pasirinkimui glaustai apžvelgiami ir lyginami bendrieji DI ryšio protokolai. Duomenų perdavimo protokolams palyginti naudojami skirtingi kriterijai. Darbe siekiama atrasti energijai taupų, tačiau saugumą užtikrinantį metodą, tuo pačiu atsižvelgiant ir į veikimo diapozoną, duomenų perdavimo greitį ir įrenginių, komunikacijos palaikymo kainą.

Vietinio belaidžio tinklo technologiją naudojantys įrenginiai naudoja daug energijos, ypač jeigu siekiama užtikrinti saugumą, todėl šios kaip atskiros technologijos lyginamojoje analizėje nepateiksime. Dėl energijos suvartojimo kriterijaus korinio ryšio technologijų GSM ir LTE taip pat nelyginsime. Visgi labiau specializuoti ir DI pritaikyti LTE pogrupiai NB-IOT ir LTE-M bus įtraukti į analizę.

Pagal pasirinktus kriterijus bus palyginti aštuoni duomenų perdavimo protokolai dažniausiai naudojami DI tinkluose. Pagal kiekvieną 1 lentelė pateiktą charakteristiką apžvelgsime ir atrinksime protokolus. Du svarbiausi faktoriai į kuriuos norime fokusuotis yra saugumas ir taupumas energijos suvartojimui.

1 lentelė. DI duomenų perdavimo protokolai ir jų charakteristikos

-	**SigFox**	**LoRa**	**NB-IOT**	**LTE-M**	**MQTT**	**BLE**	**ZigBee**	**Z-Wave**
Saugumas	Dalinai atsižvelgta [8]	AES-128 [17]	AES-128 [23]	AES-128 [23]	AES-256	AES-128 [25]	AES	AES-128
Energijos suvartojimas	10 mW-100 mW [4]	Mažas-Vidutinis (10 - 125mA)	Didelis (167mA)	Didelis (202mA)	Mažas (70mA)	Mažas (15mA)	Mažas (30mA)	Mažas (23mA)
Diapozonas	10-50km[4]	5-15km [4]	1-10km [21]	1-10km [18]	Priklauso nuo tinklo	240m [11]	10 - 100 m [1]	30 - 100m [1]
Duomenų perdavimo greitis	600 bps [4]	37.5 kbps [4]	66 kbps [19]	1119 kbps [19]	Priklauso nuo tinklo	0.25 - 2 Mbps	250 kbps [10]	40 kbps [13]
Kaina	Didelė	Didelė	Didelė	Didelė	Maža	Maža	Maža	Vidutinė

Energijos suvartojimas: *SigFox* ir LoRa suvartoja mažai energijos, o NB-IOT ir LTE-M – daugiau. BLE, *Zigbee* ir *Z-Wave* yra mažos galios protokolai. MQTT reikalingas nedidelis duomenų kiekis, o energijos suvartojimas yra mažas.

Saugumas: *SigFox*, LoRa, NB-IOT, LTE-M ir MQTT naudoja saugumą užtikrinančius protokolus, tokius kaip AES ir TLS, kad užšifruotų duomenis. BLE, *Zigbee* ir *Z-Wave* saugos lygiai skiriasi,

priklausomai nuo konkretaus įgyvendinimo, tačiau yra galimybė naudoti saugumą užtikrinančius protokolus.

Diapazonas: *SigFox* ir LoRa yra tolimojo ryšio protokolai, o BLE, *Zigbee* ir *Z-Wave* yra trumpojo nuotolio protokolai. NB-IOT ir LTE-M yra korinio ryšio protokolai, kurių diapazonas priklauso nuo korinio tinklo aprėpties. MQTT diapazonas nustatomas pagal tinklą, kuriame jis veikia.

Duomenų perdavimo greitis: LTE-M ir BLE technologijos gali suteikti didžiausią duomenų greitį. Mažiausią duomenų greitį turi tolimojo ryšio protokolai *SigFox* ir LoRa.

Kaina: *SigFox* ir LoRa įrenginiai yra lyginamai brangūs, palyginti su BLE, *Zigbee* ir *Z-Wave* įrenginiais. NB-IOT ir LTE-M įrenginiai taip pat yra gana brangūs. MQTT įrenginiai gali būti palyginti pigūs, nes jiems nereikia konkrečios aparatinės įrangos.

Pagal pasirinktus kriterijus labiausiai atitinkantys duomenų perdavimo protokolas DI tinkle yra BLE. Šis protokolas yra efektyviausias energijos vartojimo aspektu ir gali užtikrinti pakankamą saugumą, taip pat yra lankstus ir suteikia pakankamą diapozoną, jeigu tinklas yra sudarytas tinklelio principu. Verta paminėti, kad MQTT naudojimas kartu su BLE gali suteikti papildomą DI įrenginių saugos sluoksnį, tačiau tai reikalauja papildomų energijos resursų, todėl rekomenduojama MQTT naudoti mazguose, kurie turi interneto prieiga ir komunikuoja su serveriais.

MQTT užtikrina saugų ryšį šifruodamas duomenis, perduodamus tarp įrenginio ir brokerio, naudojant TLS protokolą. Taip išvengiama siunčiamų duomenų pasiklausymo ir klastojimo. Naudojant MQTT ir BLE kartu, išoriniam aktoriui sunkiau perimti ir sugadinti perduodamus duomenis, nes abu protokolai suteikia skirtingus šifravimo sluoksnius. Tačiau svarbu pažymėti, kad nors MQTT ir BLE gali užtikrinti papildomą saugumą, jie negali užkirsti kelio visoms saugumo spragoms. Pavyzdžiui, jei įrenginyje yra pažeidžiamumas arba jei raktai tvarkomi netinkamai, ryšys vis tiek gali būti pažeistas.

1.3. ESP32 mikrovaldiklis daiktų interneto įrenginiuose

Toliau plėtoti DI ir išplėsti taikymo sritis būtini galingi, nebrangūs ir mažai energijos naudojantys sprendimai. Įrenginys turi turėti mažą formos koeficientą, kuo mažesnis dydis ir įrenginio svoris, tuo platesnė jo taikymo sritis. ESP32 sistemos mikrovaldikliai yra vis dažniau naudojami DI įrenginiuose, kadangi atitinka aukščiau išvardintus reikalavimus, be to yra palyginamai pigūs. Dėl paminėtų savybių ir pranašumų ESP32 sistemos mikrovaldikliai bus naudojami šiame darbe kuriant saugų ir energijos atžvilgiu taupų DI tinklą. Toliau apžvelgsime ESP32 sistemos mikrovaldiklio tinklo architektūrą, taikomus protokolus ir saugumo problemas.

1.3.1. ESP32 mikrovaldiklio tinklo architektūra

ESP32 tinklo architektūra sukurta aplink 2,4 GHz vietinio belaidžio tinklo modulį, kuris palaiko ir infrastruktūrinį, ir decentralizuotą režimus. Jame taip pat yra dviejų režimų *Bluetooth* modulis. vietinio belaidžio tinklo modulis turi integruotą protokolų rinkinį, kuris palaiko įvairius interneto protokolus, tokius kaip TCP, UDP, HTTP, HTTPS, MQTT ir kt. *Bluetooth* modulis palaiko ir klasikinį *Bluetooth*, ir BLE protokolus.

ESP32 apima daugybę integruotų periferinių įrenginių, tokių kaip skaitmeninis signalo procesorius DSP (angl. *Digital Signal Processors*), didelės spartos UART, I2C ir SPI sąsajos, taip pat daug

bendrosios paskirties įvesties / išvesties jungčių. Šie įrenginiai gali būti naudojami sąsajai su išoriniais jutikliais ir kitais įrenginiais. Mikrovaldiklis taip pat apima daugybę integruotų saugos funkcijų, tokių kaip saugus įkrovimas, „flash“ šifravimas ir saugi raktų saugykla. Šios funkcijos padeda apsisaugoti nuo įvairių tipų atakų, pvz., tų, kuriomis bandoma pavogti neskelbtiną informaciją arba pažeisti įrenginį.

Be to, ESP32 turi integruotą realaus laiko operacinę sistemą RTOS (angl. *Real Time Operating System*), kuri palaiko daugiafunkcinį darbą ir leidžia lengvai valdyti įvairius sistemos išteklius, tokius kaip atmintis ir procesoriaus laikas. ESP32 tinklo architektūra sukurta taip, kad būtų galinga ir įvairiapusė platforma prijungtiems įrenginiams ir programoms kurti, taip pat suteiktų patikimų saugos funkcijų, apsaugančių nuo įvairių tipų atakų.

1.3.2. ESP32 mikrovaldiklio saugos funkcijos

ESP32 apima daugybę integruotų saugos funkcijų, apsaugančių nuo įvairių tipų atakų. ESP32 saugos funkcijos:

- Saugus paleidimas (angl. *Secure boot*): ši funkcija padeda užtikrinti, kad įrenginyje veiktų tik gamintojo skaitmeniniu parašu pasirašyta programinė įranga. Tai neleidžia užpuolikams įkelti kenkėjiškos programinės įrangos į įrenginį.
- *Flash* šifravimas: funkcija užšifruojanti įrenginyje saugomą programinę įrangą, kad užpuolikams būtų sunkiau iš įrenginio išgauti konfidencialią informaciją.
- Saugi raktų saugykla: ESP32 turi integruotą saugaus rakto saugojimo modulį, kuris gali būti naudojamas saugoti jautrius raktus, tokius kaip kriptografiniai raktai, naudojami šifravimui ir iššifravimui.
- *WiFi Protected Access* (WPA) ir WPA2 protokolai, kurie užtikrina aukštesnį belaidžių tinklų saugumo lygį, taip pat palaiko AES ir laikinojo rakto vientisumo protokolo (TKIP) šifravimo algoritmus.
- ESP32 gali palaikyti abipusį įrenginio ir prieigos taško autentifikavimą, o tai padeda išvengti vyro viduryje MITM (angl. *Man-in-the-Middle*) atakų.
- Integruotas palaikymas TLS ir SSL (angl. *Secure Sockets Layer*) protokolams šifruoti tinklo ryšius ir apsaugoti nuo pasiklausymo ir klastojimo.

Šios funkcijos padeda apsaugoti įrenginį ir jo tvarkomą saugią informaciją nuo dažniausiai pasitaikančių kibernetinių atakų prieš DI įrenginius. Tačiau svarbu pažymėti, kad sistemos saugumo lygis priklauso nuo įdiegimo, funkcijų įgalinimo ir tinkamo jų naudojimo. Įrenginys su visomis saugos funkcijomis vis tiek gali būti pažeidžiamas, jei jis nėra tinkamai sukonfigūruotas, ir panašiai, įrenginys be visų saugos funkcijų, bet su tinkama konfigūracija gali būti saugesnis nei įrenginys su visomis funkcijomis, bet netinkamai sukonfigūruotas.

1.3.3. Saugumo sprendimai daiktų interneto ESP32 mikrovaldiklių tinkluose

ESP32 mikrovaldiklių gamintoja *Espressif Systems* privati puslaidininkių įmonė yra sukūrusi tris belaidžio ryšio protokolus ESP-BLE-MESH, ESP-WIFI-MESH ir ESP-NOW [14]. Kaip galima nuspręsti iš pavadinimų, jų savybės ir naudojimo atvejai skiriasi.

ESP-BLE-MESH yra BLE pagrįstas protokolas tinklelio topologijos tinklams kurti. Tai leidžia įrenginiams bendrauti tarpusavyje tinkle, kur kiekvienas įrenginys gali veikti kaip perdavimo mazgas kitiems įrenginiams. ESP-BLE-MESH sukurtas mažos galios ir mažo pralaidumo

programoms ir tinka mažų įrenginių tinklams, kuriems taikomi riboto diapazono ir mažo duomenų perdavimo spartos reikalavimai, kurti.

ESP-WIFI-MESH yra panašus ESP-BLE-MESH protokolui, tik yra grįstas vietinio belaidžio tinklo technologija. Šis protokolas užtikrina didesnį pralaidumą, bet tuo pačiu reikalauja didesnės galios. Ši technologija tinka kurti didesnių įrenginių tinklus, kuriems taikomi didesni duomenų perdavimo spartos reikalavimai, pavyzdžiui, vaizdo transliacijai ir failų perdavimui.

ESP-NOW yra patentuotas *Espressif Systems* sukurtas protokolas, skirtas kurti P2P tinklus. Tai leidžia įrenginiams tiesiogiai bendrauti tarpusavyje be centrinio šakotuvo ar maršrutizatoriaus. ESP-NOW kaip ir ESP-BLE-MESH tinka kurti DI tinklams su riboto diapazono ir mažo duomenų perdavimo spartos reikalavimais.

Dar vienas sprendimas, kuris pateikiamas Steph Rudd ir Hamish Cunningham moksliniame darbe, yra trys projektavimo koncepcijos [6]: paskirstyta infrastruktūra (angl. *Distributed infrastructure*), atrankinis privatumas (angl. *Selective privacy*) ir DI tinkami šifrai (angl. *IoT-appropriate ciphers*). Darbe pateikiamas sprendimas keisti konfigūraciją ir saugumo modelio topologiją taip, kad konfidencialumo aspektas būtų sumažintas kur įmanoma ir CA (angl. *Certificate Authority*), VA (angl. *Verification Authority*) ir RA (angl. *Registration Authority*) subjektų pašalinimas taip atsisakant centrinio serverio (arba trečiosios šalies) įdarbinimo modelio. Pašalinus TLS reikalavimą serveriui ir šifrų rinkiniams, tokius protokolus kaip BLE, LoRaWAN ir MQTT galima apsaugoti naudojant mažiau energijos suvartojančius atributus pasitelkiant apskaičiavimo spartinimą aparatine įranga ESP32 mikrovaldikliuose.

Naudojantis trijomis projektavimo koncepcijomis būtų galima pagerinti ESP32 DI tinklo saugumo aspektą netgi naudojant ESP-BLE-MESH protokolą, tačiau tai reikalauja papildomų išteklių prieš diegiant tokios konfigūracijos tinklą. Reikia atlikti poveikio duomenų apsaugai vertinimą (DPIA) – apskaičiuoti kiekvieno duomenų tipo riziką sistemoje, kad būtų galima sutaupyti energijos su tais duomenimis, kuriems galbūt nereikia konfidencialumo. Be to, suprastėtų tinklo plečiamumo aspektas, nes pridedant daugiau įrenginių prie tinklo reiktų papildomos konfigūracijos. Visgi darbe buvo ištirta, kad CBC (angl. *Cipher Block Chaining*) šifras palyginti su kitais tirtais šifrais yra 68% efektyvesnis energijos vartojimo požiūriu didesnių duomenų mainuose, kur tiktų pilnas AEAD (angl. *Authenticated Encryption with Additional Data*) taikymas.

Pagal duomenų perdavimo protokolų analizės išvadas 1.2.2.5 skyrelyje toliau nagrinėsime tik ESP-BLE-MESH protokolą iš trijų apžvelgtų *Espressif* technologijų, kadangi jis yra grįstas BLE 5 ryšio protokolu, kuris turėtų būti mažiausiai imlus energijos sunaudojimui ir užtikrinti pakankamą tinklo saugumą.

1.3.4. ESP-BLE-MESH protokolas

ESP-BLE-MESH yra BLE protokolų rinkinys sukurtas *Espressif Systems*, skirtas kurti belaidžius tinklelio tinklus. Jis pagrįstas *Bluetooth Mesh* specifikacija ir leidžia kurti didelio masto, mažos galios BLE įrenginių tinklus.

Vienas iš pagrindinių ESP-BLE-MESH privalumų yra energijos vartojimo efektyvumas, kuris leidžia tinklo įrenginių baterijos veikimo laiką prailginti. ESP-BLE-MESH naudoja mažos galios, mažos duomenų perdavimo spartos ryšį, kuris puikiai tinka baterijomis maitinamiems įrenginiams.

Be to, protokolų rinkinyje yra daug energiją taupančių funkcijų, tokių kaip miego režimai ir veikimo ciklas, kurios gali dar labiau padidinti tinklo įrenginių baterijos veikimo laiką.

Kalbant apie saugumą, ESP-BLE-MESH suteikia keletą funkcijų, užtikrinančių saugų tinklo veikimą. Protokolo rinkinys apima saugią sąranka, kuri saugiai prijungia įrenginius prie tinklo. Kita funkcija kurią palaiko šis rinkinys yra prieigos kontrolė, kuri apriboja prieigą prie tam tikrų tinklo dalių tik įgaliotiems įrenginiams.

1.3.4.1. ESP-BLE-MESH topologija

ESP-BLE-MESH topologija yra kelių šuolių, savaime išgijantis tinklelio tinklas. Kiekvienas tinklo įrenginys vadinamas mazgu. Kiekvienas mazgas gali veikti kaip šakninis mazgas (angl. *root node*), perdavimo mazgas (angl. *relay node*), draugo mazgas (angl. *friend node*) ir mažos galios mazgas (angl. *low power node*), kuris skirtas naudoti mažai energijos ir prailginti akumuliatoriaus veikimo laiką. Kiekvienas mazgas palaiko maršruto parinkimo lentelę, kurioje pateikiama informacija apie gretimus mazgus ir jų ryšio būseną. Funkcija vadinama „sleepy end-device“, leidžia įrenginiams pereiti prie mažos galios miego režimo, kai jie aktyviai nesiunčia arba negauna duomenų ir tapti mieguistu mazgu (angl. *sleepy node*).

Draugo mazgas yra mazgo tipas, naudojamas išplėsti tinklo diapazoną ir pagerinti ryšio patikimumą. Šis mazgas veikia kaip tarpinis serveris kitiems tinklo mazgams ir gali palaikyti ryšį su kitais tinklo įrenginiais jų vardu, net kai jie veikia mažos galios miego režimu. Tai gali padėti užtikrinti, kad duomenys būtų pristatyti, net jei vienas iš kelių nepasiekiamas.

Šakninis mazgas yra įrenginys, kuris yra tiesiogiai prijungtas prie tinklo sąsajos ir gali palaikyti ryšį su kitais tinklo įrenginiais ir tinklo sąsaja. Šakniniai mazgai veikia kaip pagrindinis duomenų įvedimo ir išėjimo į tinklą ir iš jo taškas.

Perdavimo mazgas yra įrenginys, kuris nėra tiesiogiai prijungtas prie tinklo sąsajos ir gali susisiekti tik su kitais tinklo įrenginiais. Perdavimo mazgai veikia kaip duomenų perdavimo tinkle tarpininkai, perduodantys duomenis iš vieno mazgo į kitą.

1.3.4.2. ESP-BLE-MESH saugumas

Papildomai prie jau minėtų saugumo funkcijų ESP-BLE-MESH taip pat naudoja pranešimų šifravimą, kuris užtikrina, kad visi tinklu perduodami duomenys būtų užšifruoti ir apsaugoti nuo klastojimo. Protokolų rinkinio viršutiniame transporto ir tinklo sluoksniuose naudojamas AES-CCM AEAD algoritmas.

Kalbant apie įrenginio autentifikavimą, ESP-BLE-MESH naudoja elipsinės kreivės skaitmeninio parašo algoritmą ECDSA (angl. *Elliptic Curve Digital Signature Algorithm*), kad autentifikuotų įrenginius tinkle. Taip užtikrinama, kad prie tinklo gali prisijungti tik leistini įrenginiai, o šių įrenginių perduodamais duomenimis galima pasitikėti.

ESP-BLE-MESH protokolų rinkinyje taip pat yra keletas kitų saugos funkcijų, tokių kaip apsauga nuo pakartojimo atakos, saugus raktų keitimas ir saugus programinės įrangos atnaujinimas. Šios funkcijos padeda užtikrinti bendrą tinklo saugumą ir apsisaugoti nuo įvairių tipų atakų.

Nepaisant ESP-BLE-MESH teikiamų saugos funkcijų, svarbu pažymėti, kad tinklas vis tiek gali būti pažeidžiamas tam tikrų tipų atakų, pvz., MITM atakų. Norint apsisaugoti nuo tokio tipo atakų, svarbu naudoti geriausią praktiką ir įdiegti papildomas saugos priemones.

1.3.4.3. ESP-BLE-MESH energijos vartojimo efektyvumas

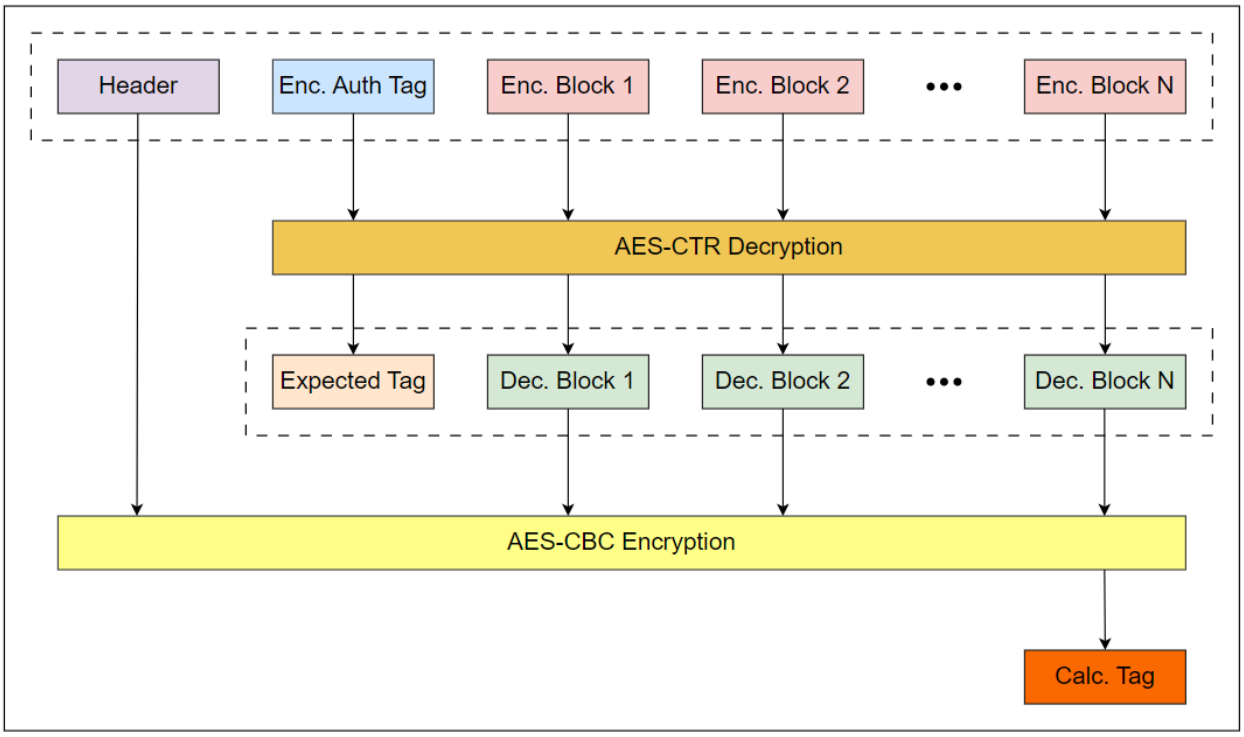
Viena iš pagrindinių ESP-BLE-MESH energiją taupančių funkcijų yra miego režimo funkcija. Ši funkcija leidžia tinklo įrenginiams pereiti į mažos galios būseną, kai jie aktyviai nesiunčia arba nepriima duomenų. Tai padeda sumažinti tinklo įrenginių energijos sąnaudas, nes jie sunaudoja mažiau energijos momentais tarp duomenų perdavimo ir priėmimo.

Kita ESP-BLE-MESH energiją tausojanti funkcija yra jo veikimo ciklo funkcija. Veikimo ciklas naudojamas siekiant sumažinti laiką kuomet įrenginys perduoda arba klausosi duomenų. Tai galima padaryti suplanavus, kad įrenginys tik perduotų arba klausytųsi duomenų tam tikru laiku, arba naudojant atsitiktinį tvarkaraštį, kad būtų sumažinta susidūrimų su kitais įrenginiais tikimybė. Tai taip pat padeda sumažinti tinklo įrenginių energijos suvartojimą.

ESP-BLE-MESH sumažinto energijos suvartojimo (angl. *reduced power consumption*) funkcija leidžia tinklo įrenginiams reguliuoti radijo bangų perdavimo galią ir darbo ciklą, kad būtų sumažintas energijos suvartojimas ir pailgėtų tinkle esančių įrenginių baterijos veikimo laikas. Perdavimo galia reguliuojama priklausomai nuo atstumo tarp įrenginių, darbo ciklo grafikas reguliuojamas pagal tinklo srautą. Abi funkcijos taip pat reguliuojamos pagal likusį įrenginio baterijos lygį.

1.4. Duomenų autentifikavimo ir šifravimo algoritmai

AES-CCM yra duomenų autentifikavimo ir šfiravimo algoritmas **1 pav**. AES-CCM (angl. *Advanced Encryption Standard – Counter with CBC-MAC*) yra plačiai naudojamas autentifikuoto šifravimo režimas, užtikrinantis duomenų konfidencialumą ir autentiškumą. AES yra naudojamas kaip pagrindinis bloko šifras ir sujungia skaitiklio (CTR) režimo šifravimą su CBC-MAC (angl. *Cipher Block Chaining Message Authentication Code*) autentifikavimu. AES-CCM gali būti brangus energijos vartojimo atžvilgiu dėl atliekamų didelio kiekio skaičiavimų, ypač ribotų išteklių įrenginiuose dėl naudojamo AES bloko šifro.



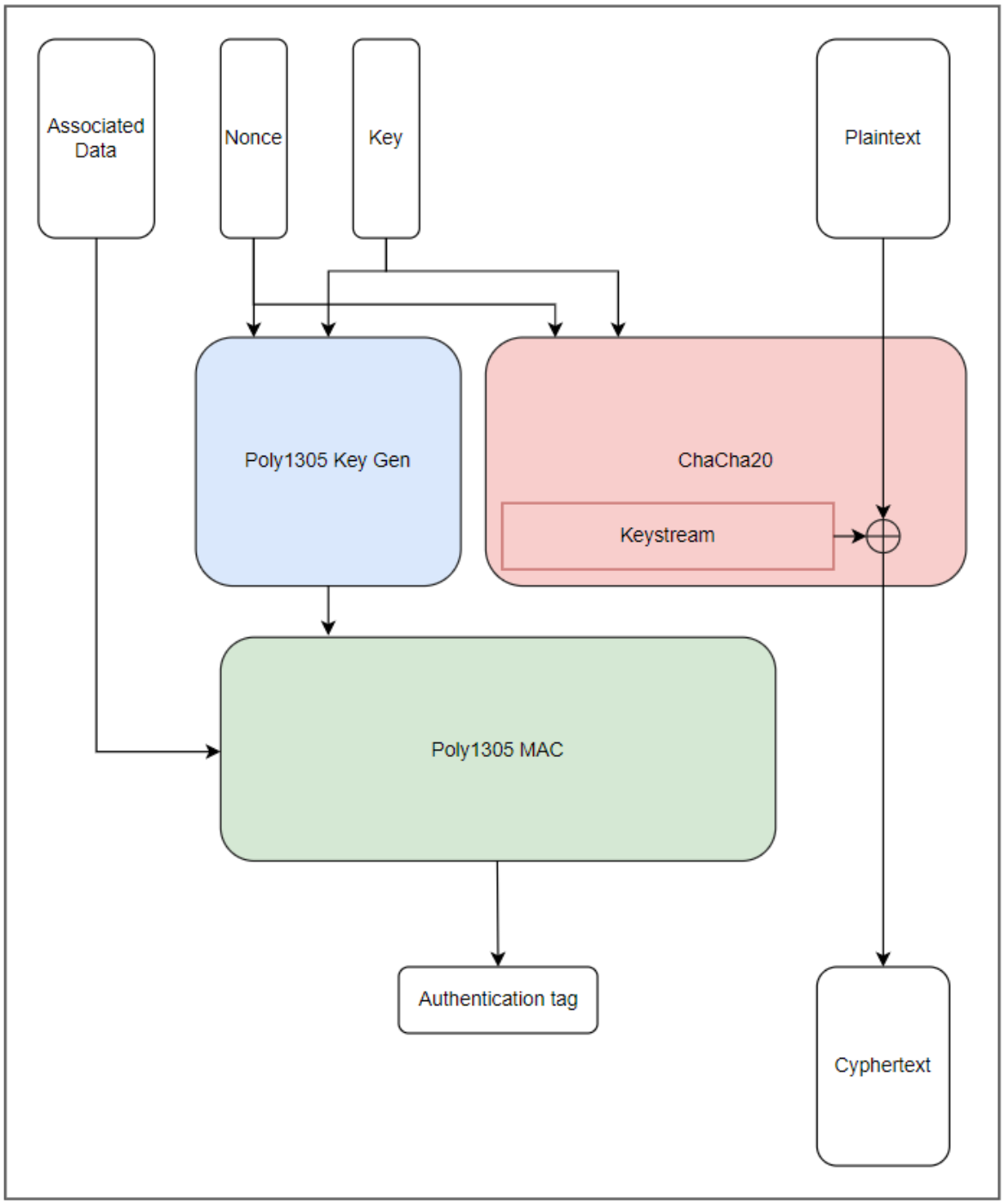
1 pav. AES-CCM blokų šifro veikimo schema

Blokų šifras veikia fiksuoto dydžio duomenų blokuose, paprastai 64 arba 128 bitų dydžio (ESP BLE MESH atveju). Įvesties pranešimas yra padalinamas į blokus, o kiekvienas blokas užšifruojamas arba iššifruojamas atskirai. Kiekvienas blokas reikalauja kelių raundų sudėtingų matematinių operacijų, tokių kaip pakeitimas, permutacija ir raktų maišymas. Šių operacijų atlikimas su dideliais duomenų blokais gali sunaudoti didelius kiekius įrenginio išteklių.

Blokų šifrai dažnai turi priklausomybių tarp užšifruojamų arba iššifruojamų blokų. Kiekvieno bloko šifravimas arba iššifravimas paprastai priklauso nuo ankstesnių blokų rezultatų, o tai įveda nuoseklų apdorojimą ir riboja lygiagretinimo galimybes. Dėl nuoseklumo gali būti lėtesnis bendras apdorojimas, palyginti su srauto šifrais, kurie gali veikti su duomenimis nuolat ir lygiagrečiai.

Norint sutaupyti įrenginio resursus bus naudojamas *ChaCha20-Poly1305* autentifikuoto šifravimo algoritmas, užtikrinantis duomenų perdavimo konfidencialumą, vientisumą ir autentiškumą. Tai populiarus saugios komunikacijos pasirinkimas dėl savo greičio, saugumo ir efektyvumo, ypač ribotų išteklių aplinkoje [2], [26].

ChaCha20-Poly1305 algoritmas šifruoja duomenis nenutrūkstamu bitų srautu **2 pav.** . Veikia su atskirais paprastojo ir šifruoto teksto bitais arba baitais. Dažniausiai naudoja raktų srauto (angl. *Keystream*) generatorių, kad sukurtų pseudoatsitiktinį bitų srautą, šifruoja naudojant XOR loginę operaciją sujungiant paprastą tekstą su raktų srautu. Palaiko srauto sinchronizavimo mechanizmus, pvz., inicijavimo vektorius (IV) arba nonce, kad būtų užtikrintas saugus šifravimas. Paprastai veikia greičiau nei blokiniai šifrai dideliems duomenų kiekiams užšifruoti.



2 pav. *ChaCha20-Poly1305* srauto šifro veikimo schema

ChaCha20 srauto šifras ir *Poly1305* maišos šeima yra sukurti taip, kad būtų lengvi ir efektyvūs skaičiavimo požiūriu, todėl tinka įrenginiams su ribota apdorojimo galia, atmintimi ar trumpu baterijos veikimo laiku. Taip pat *ChaCha20* šifras sukurtas taip, kad jį būtų galima lygiagrečiai suderinti, o tai leidžia efektyviai išnaudoti šiuolaikinius procesoriuos, įskaitant tuos, kurių ištekliai yra riboti. *ChaCha20-Poly1305* paprastai reikalauja mažiau atminties nei AES-CCM. Tai naudinga aplinkoje, kur labai svarbu sumažinti atminties naudojimą.

1.5. Išvados

Atlikus DI duomenų protokolų analizę galima teigti, kad DI tinkluose, kuriuose svarbu energijos vartojimo efektyvumas, saugumas, kuriems nėra keliami didelio diapozono reikalavimai ir svarbi maža įrenginių kaina, BLE gali būti geras pasirinkimas tinkamai sukonfigūravus tinklo įrenginius.

ESP32 mikrovaldikliai yra puiki opcija DI tinkluose, kadangi yra lyginamai nebrangūs, maži ir turi daugybę integruotų saugos funkcijų, apsaugančių nuo įvairių tipų kibernetinių atakų bei energijos vartojimo efektyvumą užtikrinančių savybių.

Apžvelgus DI tinklų ESP-BLE-MESH protokolą skirtą ESP mikrovaldikliams, atrodo, kad tai gali būti puikus pasirinkimas norint pasiekti pakankamą duomenų perdavimo apsaugą ir tinklo saugą išlaikant energijos sąnaudas minimalias.

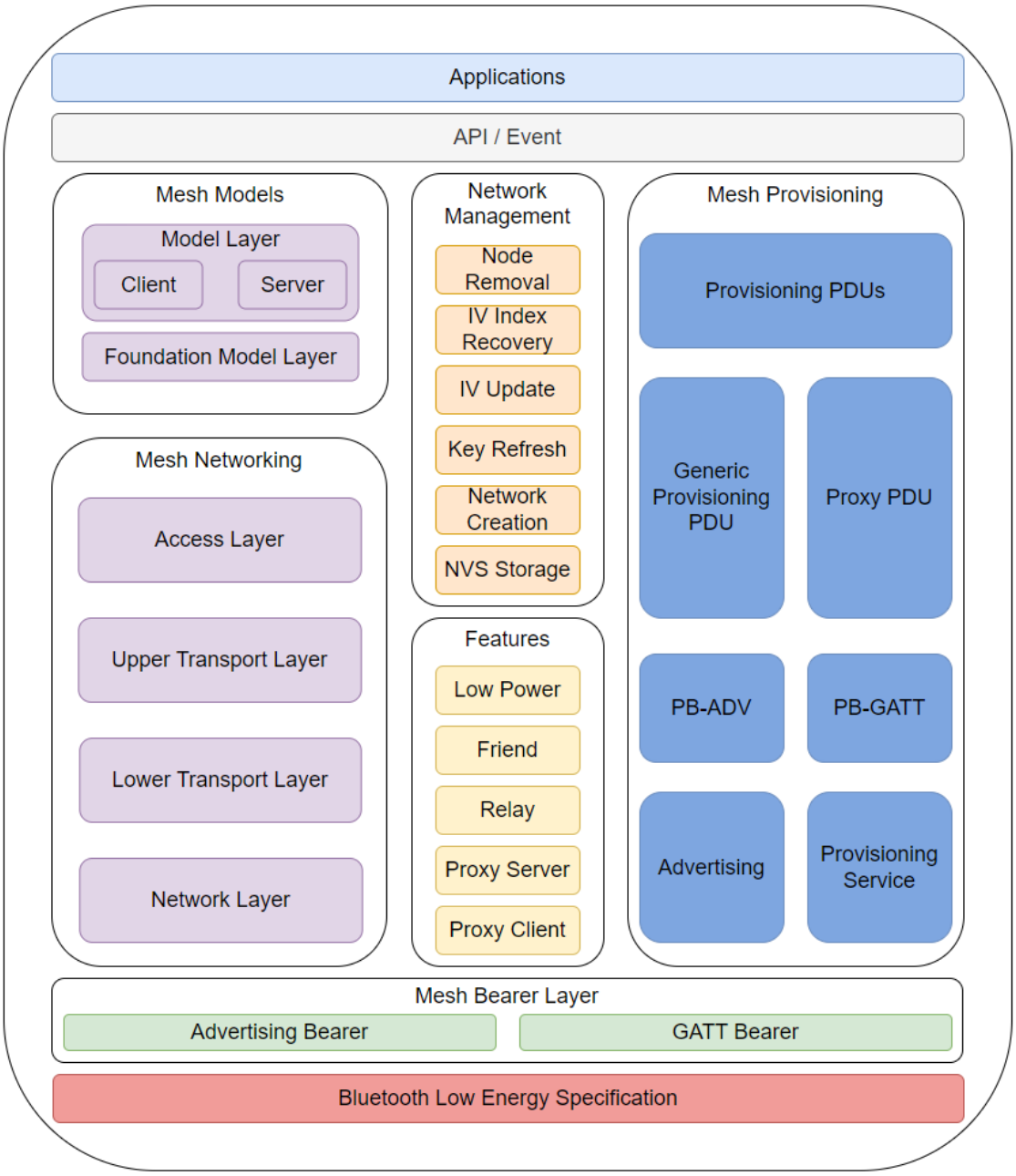
2. Daiktų interneto įrenginių ESP32 saugaus duomenų perdavimo metodas

Analizės dalyje apžvelgtas ESP-BLE-MESH protokolų rinkinys yra pakankamai geras pasirinkimas norint pagerinti duomenų perdavimo apsaugą ir tinklo saugą taupant energiją todėl jis bus naudojamas kaip pagrindas kuriant DI įrenginių tinklą. Visgi atlikus kelias modifikacijas bus siekiama sukurti mažiau energijos sunaudojantį DI įrenginių tinkle naudojamą duomenų perdavimo metodą, kuris būtų pakankamai saugus. ESP-BLE-MESH protokolų rinkinys bus modifikuojamas keičiant AEAD duomenų autentifikavimo ir šifravimo algoritmą.

2.1. ESP-BLE-MESH protokolų rinkinio tinklo architektūra

Šiuo metu ESP-BLE-MESH turi daugumą Bluetooth tinklelio profilio funkcijų ir visus kliento modelius, apibrėžtus tinklelio modelio specifikacijoje. Kitos trūkstamos funkcijos, modeliai yra kuriami ir bus išleidžiami su atnaujinimais. ESP-BLE-MESH architektūra **3 pav.** susideda iš penkių pagrindinių dalių:

- Tinklelio protokolų rinkinys:
  - Tinklelio tinklo dalis (angl. *Mesh Networking*) yra atsakinga už ESP-BLE-MESH mazgų pranešimų apdorojimą.
  - Tinklelio aprūpinimo dalis (angl. *Mesh Provisioning*) yra atsakinga už DI įrenginių priėmimą į ESP-BLE-MESH tinklą.
  - Tinklelio modeliai (angl. *Mesh Models*) yra atsakingi už Bluetooth SIG apibrėžtų modelių įgyvendinimą.
- Tinklo valdymo (angl. *Network Management*) įgyvendina keletą tinklo valdymo procedūrų, įskaitant mazgų pašalinimo procedūrą, IV indekso atkūrimo procedūrą ir kt.
- Funkcijos (angl. *Features*) įtraukia ESP-BLE-MESH funkcijas, pvz. mažos galios funkcija, draugo funkcija, perdavimo funkcija ir kt.
- Tinklelio nešiklio sluoksnis (angl. *Mesh Bearer Layer*) apima skelbimo nešiklį ir GATT nešiklį. Nešiklio sluoksnis yra labai svarbus ESP-BLE-MESH protokolų rinkiniui, nes leidžia duomenis perduoti BLE kanalais.
- Programos (angl. *Applications*):
  - Remiasi ESP-BLE-MESH protokolų rinkiniu ir tinklelio modeliais.
  - Naudodamos aplikacijų programavimo sąsaja ir tvarkydamos įvykius (angl. *Events*), sąveikauja su tinklelio tinklo, tinklelio aprūpinimo dalimis ir tinklelio modeliais.



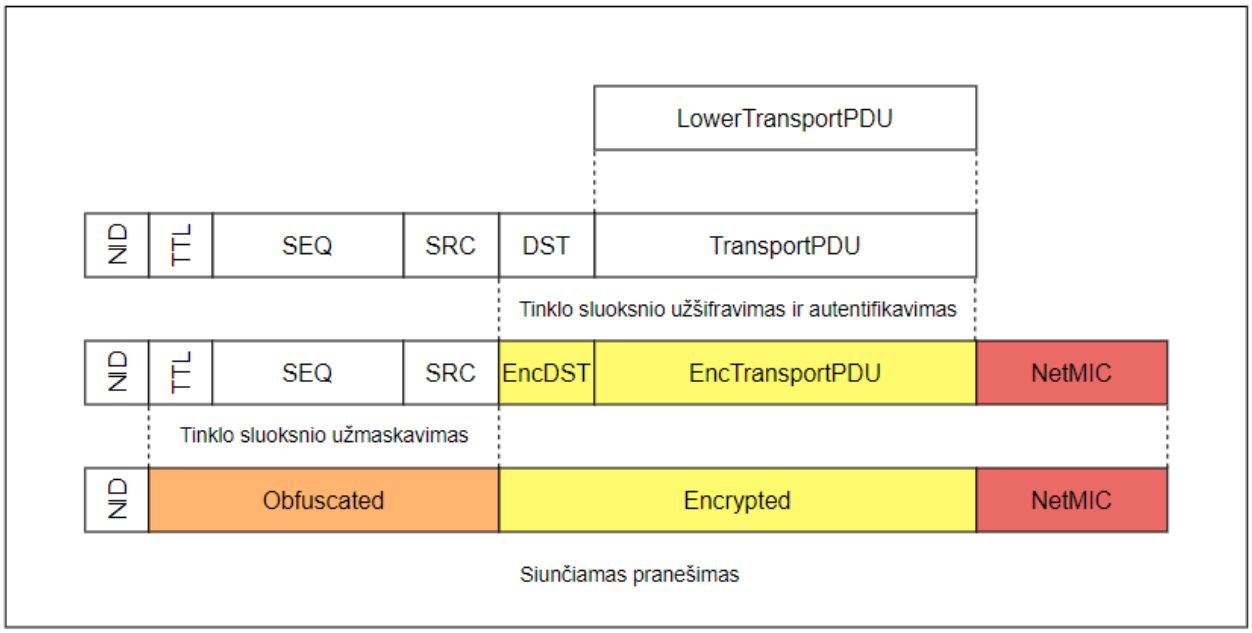
3 pav. ESP BLE MESH tinklo architektūra

Pagrindinės dalys, kuriose bus atliekami pakeitimai yra tinklelio tinklo dalis. Tinklo dalis užtikrina ryšį tarp mazgų, vykdo pranešimų šifravimą ir iššifravimą, segmentavimą ir surinkimą, tinklo išteklių valdymą tinklelio tinkle.

2.2. ESP BLE MESH duomenų autentifikavimo ir šifravimo algoritmas

Duomenys tinkle yra apsaugoti naudojant autentifikuoto šifravimo su susijusiais duomenimis algoritmą AES-CCM dviejuose lygmenyse. Duomenys yra užšifruojami ir autentifikuojami tinklo (angl. *network layer*) ir viršutiniame transporto (angl. *upper transport layer*) sluoksniuose. Kiekvienas pranešimas taip pat yra užmaskuotas, kad būtų paslėpta galima identifikavimo

informacija iš paketų. Apatiniame transporto ir tinklo sluoksnyje vykstančio proceso duomenų vienetų bendras vaizdas pateiktas žemiau **4 pav.** .



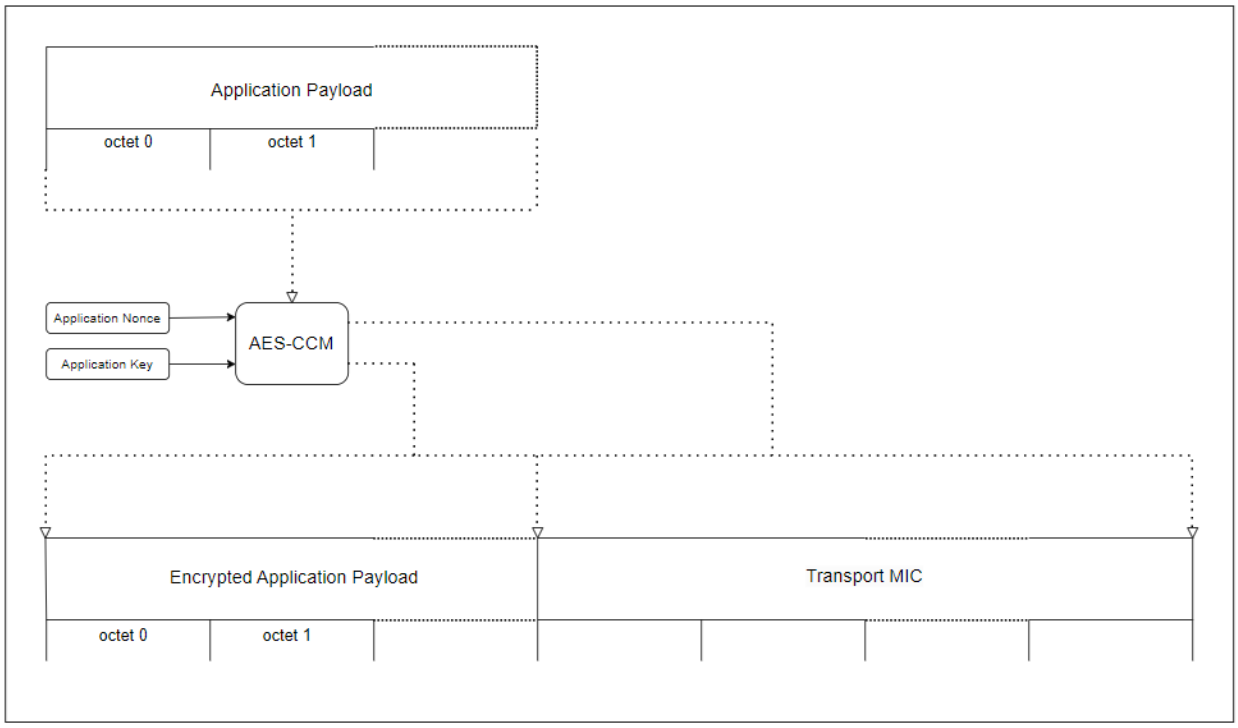
4 pav. Pranešimo duomenų užšifravimas, autentifikavimas ir užmaskavimas ESP BLE MESH

Kiekvienas pranešimas turi mažiausiai 64 bitus su juo susietos autentifikavimo informacijos. Autentifikavimo informacija gali būti padalinta tarp tinklo sluoksnio ir viršutinio transporto sluoksnio.

Kai kurie pranešimai, vadinami valdymo pranešimais, nėra autentifikuojami viršutiniame transportavimo lygmenyje ir todėl turi 64 bitų *NetMIC*. Prieigos pranešimai autentifikuojami viršutiniame transportavimo lygmenyje ir todėl turi 32 bitų *NetMIC*. Prieigos pranešimai, kurie siunčiami vienu nesegmentuotu pranešimu turi 32 bitų *TransMIC*. Prieigos pranešimai, suskirstyti į kelis tinklo PDU, gali turėti 32 bitų arba 64 bitų *TransMIC*. Tai leidžia aukštesniam sluoksniui nustatyti autentifikavimo lygį reikalingą saugiai pristatyti prieigos pranešimą ir taip pritaikyti tinkamą *TransMIC* dydį.

2.2.1. Viršutinio transporto sluoksnių duomenų autentifikavimas ir šifravimas

Pranešimų duomenų autentifikavimą ir šifravimą atlieka viršutinis transporto sluoksnis. Prieigos naudingoji apkrova yra užšifruojama ir autentifikuojama naudojant AES-CCM algoritmą. Tai yra identiškas būdas *Bluetooth* mažos energijos šifravimo ir autentifikavimo veikimui. Viršutinio transportavimo sluoksnio iliustracija pateikta žemiau **5 pav.** .

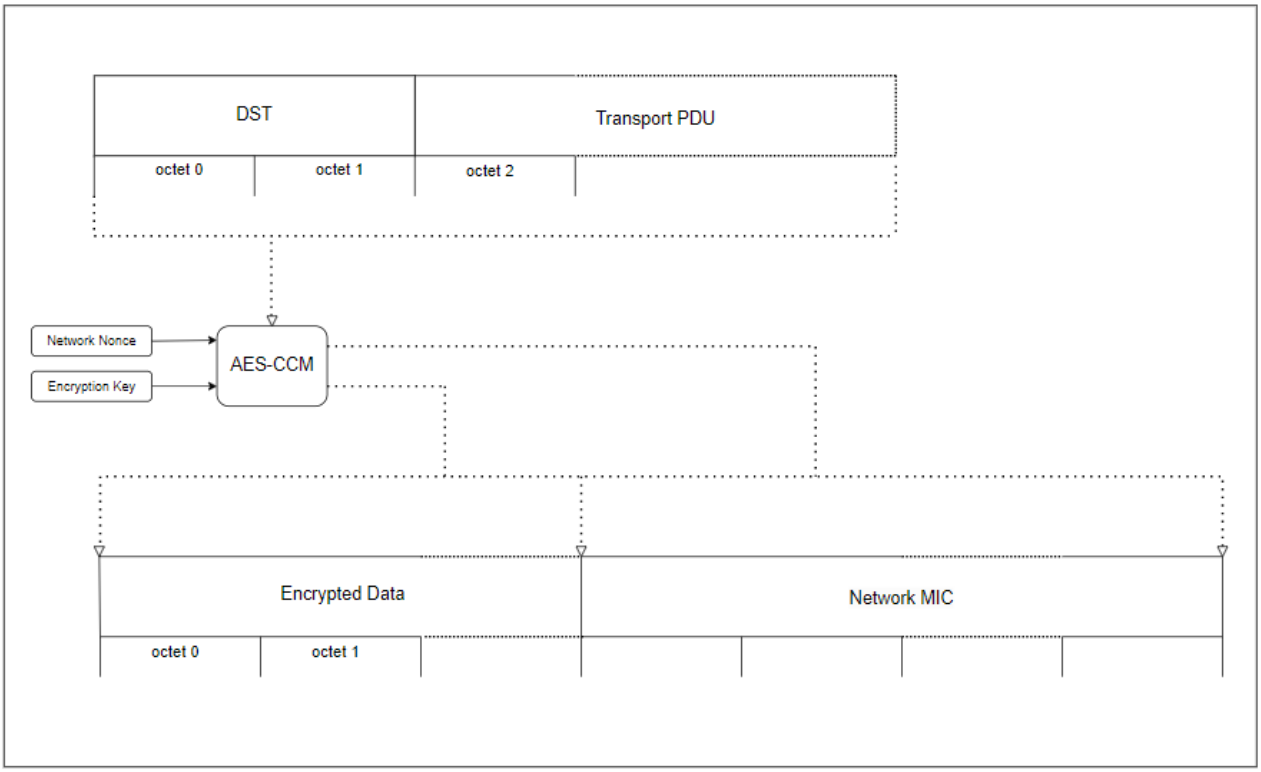


5 pav. Viršutinio transporto sluoksnio PDU šifravimas ir autentifikavimas

Jei duomenys apsaugoti naudojant programos raktą, tada prieigos naudingoji apkrova užšifruojama naudojant programų sluoksnio nonce ir programos raktą. Jei duomenys apsaugoti naudojant įrenginio raktą, tada prieigos naudingoji apkrova užšifruojama naudojant įrenginio nonce ir įrenginio raktą.

Nonce sudaro eilės numeris ir šaltinio adresas, užtikrinant, kad du skirtingi mazgai negalėtų naudoti to paties savavališko skaičiaus. IV indeksas naudojamas norint pateikti daug daugiau nonce reikšmių nei sekos skaičius gali suteikti tam tikram mazgui.

Tinklo sluoksnyje taip pat naudojamas AES-CCM paskirties adreso ir transporto protokolo duomenų vienetų užšifravimui ir autentifikavimui 6 pav. . Visi tinklo sluoksnio protokolo duomenų vienetai yra užšifruoti naudojant šifravimo raktą, kuris gaunamas iš tinklo rakto.

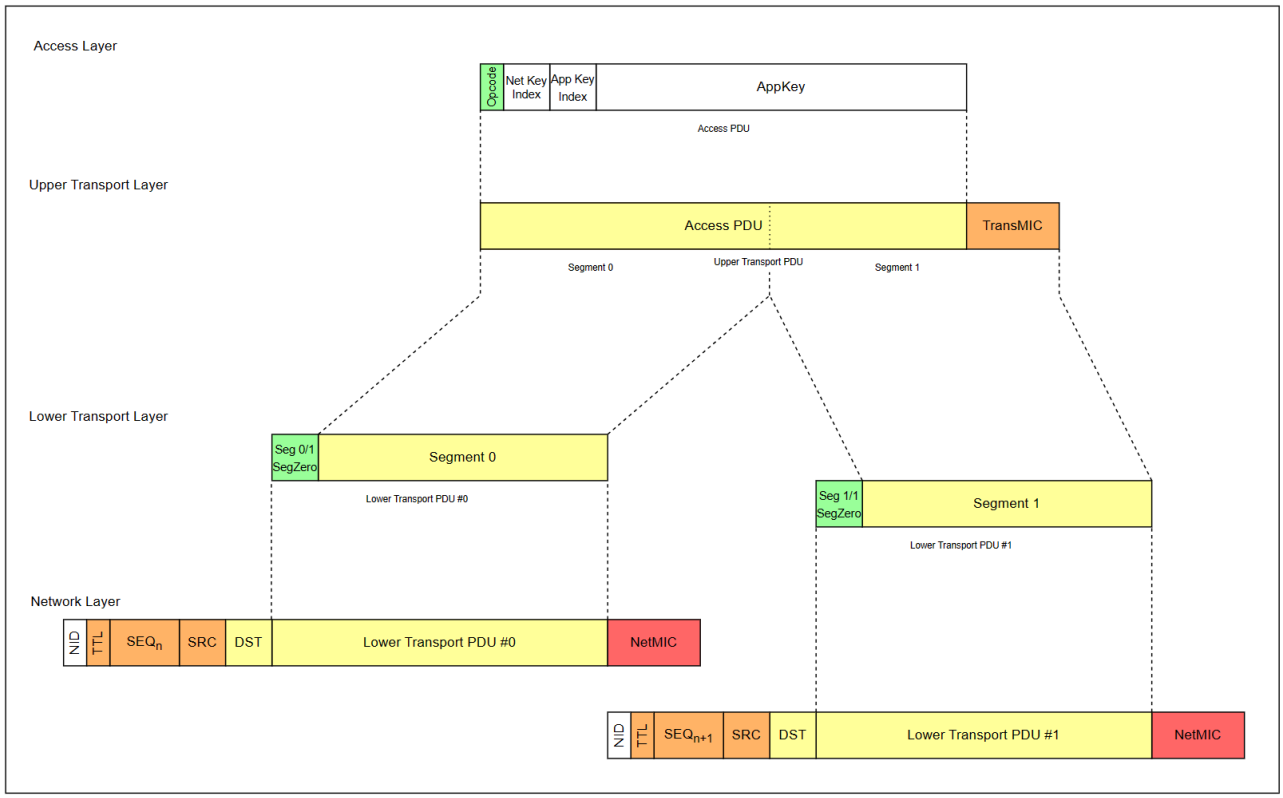


6 pav. Tinklo sluoksnio PDU šifravimas ir autentifikavimas

2.2.2. Autentifikuoto šifravimo su susijusiais duomenimis algoritmo pasirinkimas

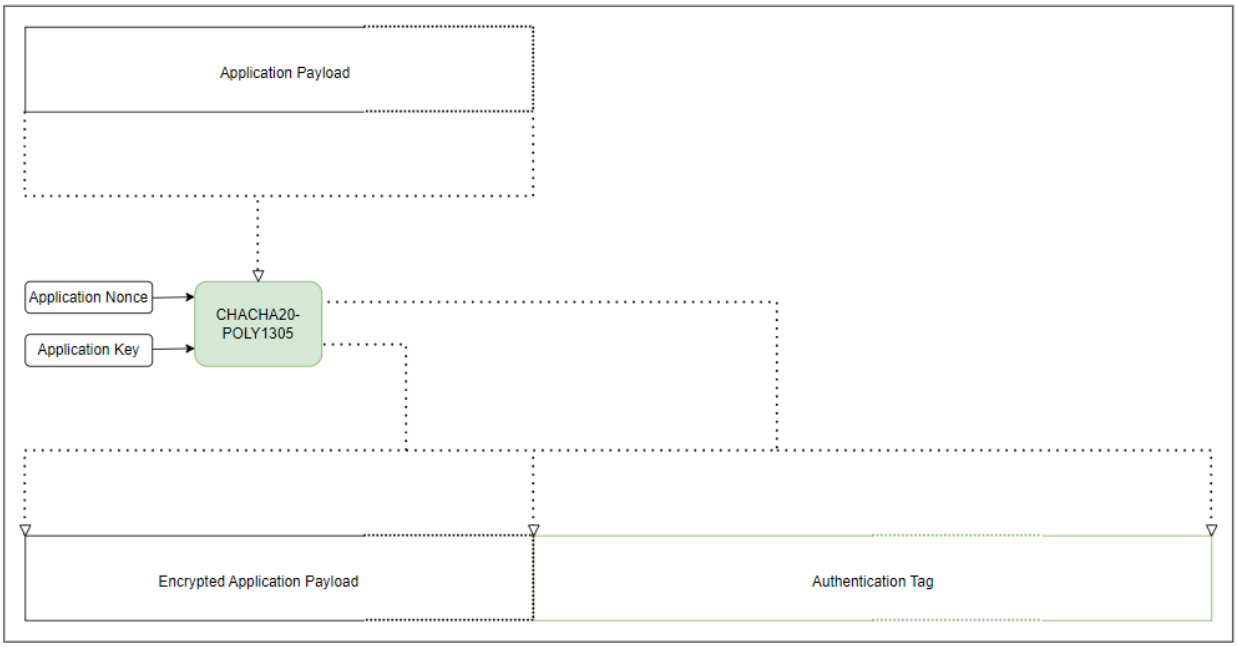
ChaCha20-Poly1305 bus panaudotas tik viršutiniame ESP BLE MESH transporto sluoksnyje 8 pav. . Iš diagramos matyti, kad įvestys išliks dvi. Išvestyse autentifikavimo žyma (angl. *Authentication tag*) bus naudojama vietoje *TransMIC* užtikrinti pranešimų vientisumui. Autentifikavimo žyma bus 128 bitų arba 16 oktetų ilgio, vietoje 4 arba 8 oktetų ilgio *TransMIC* žymos, todėl visi pranešimai bus skaidomi į mažesnius apatiniajame transporto sluoksnyje 7 pav..

Dėl AES-CCM ir *ChaCha20-Poly1305* algoritmų autentifikavimo žymių ilgių skirtumo *ChaCha20-Poly1305* nebus galima panaudoti ESP-BLE-MESH tinklo sluoksnyje. Didžiausias leistinas tinklo sluoksnio PDU dydis yra 31 baitas. Privalomos kontekstinės informacijos dydis yra 9 baitai. Žemesnio transporto sluoksnio PDU liktų 6 baitai. Tik kontekstinei informarcijai žemesnio transporto sluoksnio PDU reikia 4 baitų. Aplikacijos lygio duomenims liktų 2 baitai per vieną protokolų rinkinio tinklo žinutę. Tai padidintų siunčiamų žinučių tinklelio tinkle kiekį ir taip eikvotų įrenginių energiją.



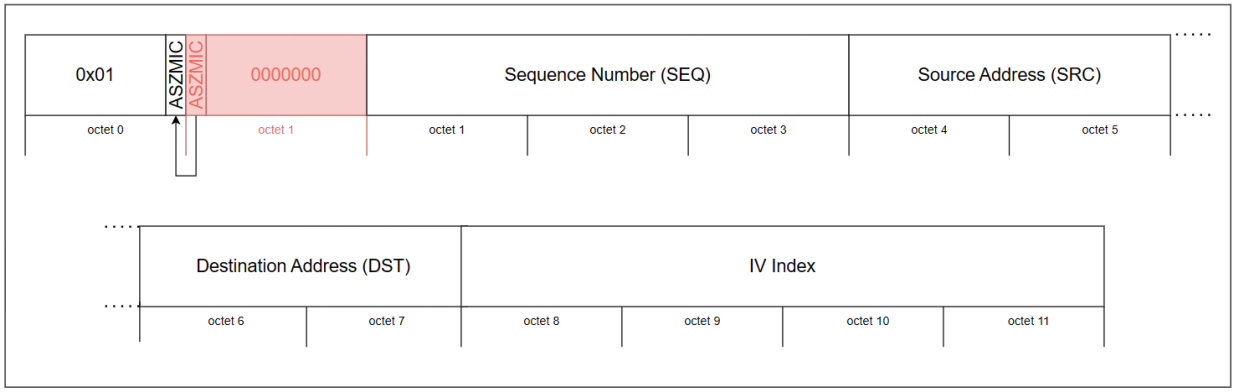
7 pav. Dviejų segmentų PDU segmentavimas ir surinkimas

Įvestys išliks panašios, tačiau jų struktūra keisis. Nonce įvestis bus 12 oktetų arba 96 bitų ilgio. Duomenų šifravimui naudojami raktai bus ne 128, o 256 bitų ilgio. Raktų apsikeitimo mechanizmas nesikeis, raktus kiekviena šalis generuos savarankiškai naudojant Diffie – Hellman elipsinės kreivės protokolu susitarimui ir bendrai paslapčiai. Sugeneruoti raktai išliks 128 bitų ilgio, tačiau prieš juos panaudojant autentifikuotam duomenų šifravimui raktas bus sujungtas su savimi ir taip išgaunamas 256 bitų ilgio raktas. Svarbu paminėti, kad duomenų šifravimo algoritmo saugumas labiausiai priklauso nuo nonce naudojimo, o ne nuo rakto. Svarbiau yra užtikrinti, kad nonce nebūtų perpanaudojamas perduodant duomenis tinkle, negu raktų transformacijos, jeigu raktų apsikeitimo mechanizmas yra pakankamai saugus.



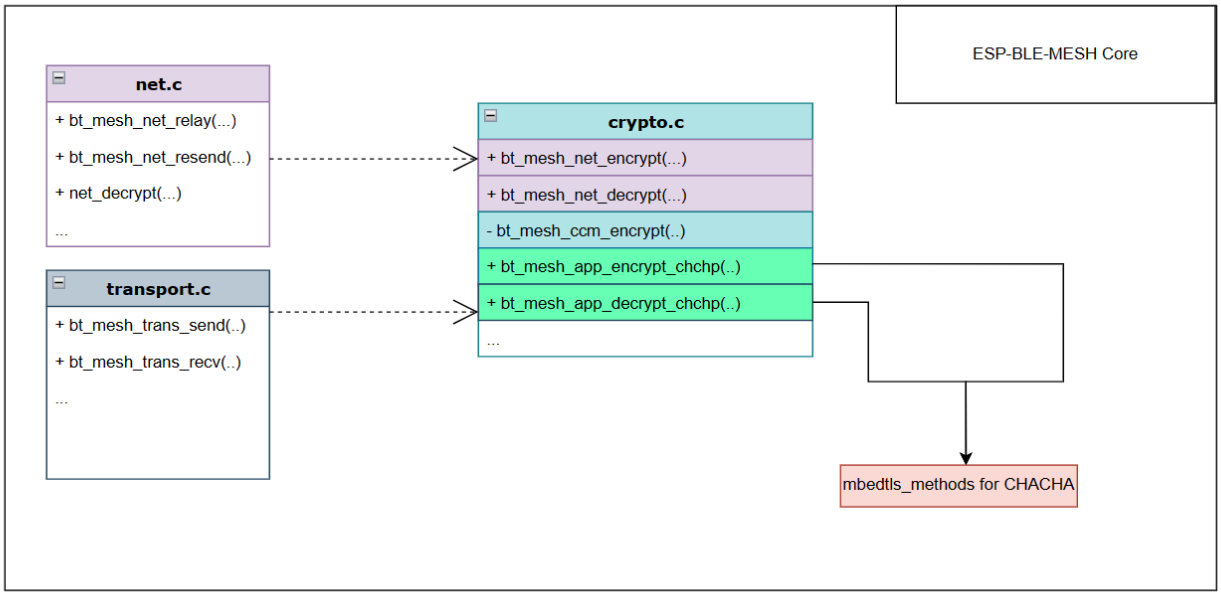
8 pav. Viršutinio transporto sluoksnio PDU šifravimas ir autentifikavimas naudojant *ChaCha20-Poly1305* algoritmą

Viršutiniame transporto sluoksnyje naudojamo programų sluoksnio nonce 2-asis oktetas bus pašalintas, o jame esantis ASZMIC bitas bus perkeltas į 1-ąjį oktetą. Pirmasis oktetas naudojamas tik nonce tipui nusakyti, o galimos reikšmės yra 0x01 arba 0x02, todėl paskutinis bitas jame visada bus nenaudojamas. Atliekami pakeitimai programų nonce matomi 9 pav. .



9 pav. Programų sluoksnio nonce struktūros pakeitimai

Norint implementuoti *ChaCha20-Poly1305* algoritmo panaudojimą ESP-BLE-MESH reiks pridėti naujus pagalbinius autentifikuoto šifravimo metodus kriptografinių funkcijų klasėje crypto.c. Pagalbine klase naudojasi tiek net.c (tinklo sluoksnio kontrolės) klasėje, tiek transport.c (viršutinio transporto sluoksnio kontrolės) klasėje naudojami metodai saugiam pranešimų siuntimui užtikrinti 10 pav.. Metodas bus įgyvendintas naudojantis mbedtls biblioteka esančia *Espressif* DI programavimo sistemoje (angl. *Espressif's official IoT Development Framework* (ESP-IDF)) su aparatinės įrangos spartinimo palaikymu.



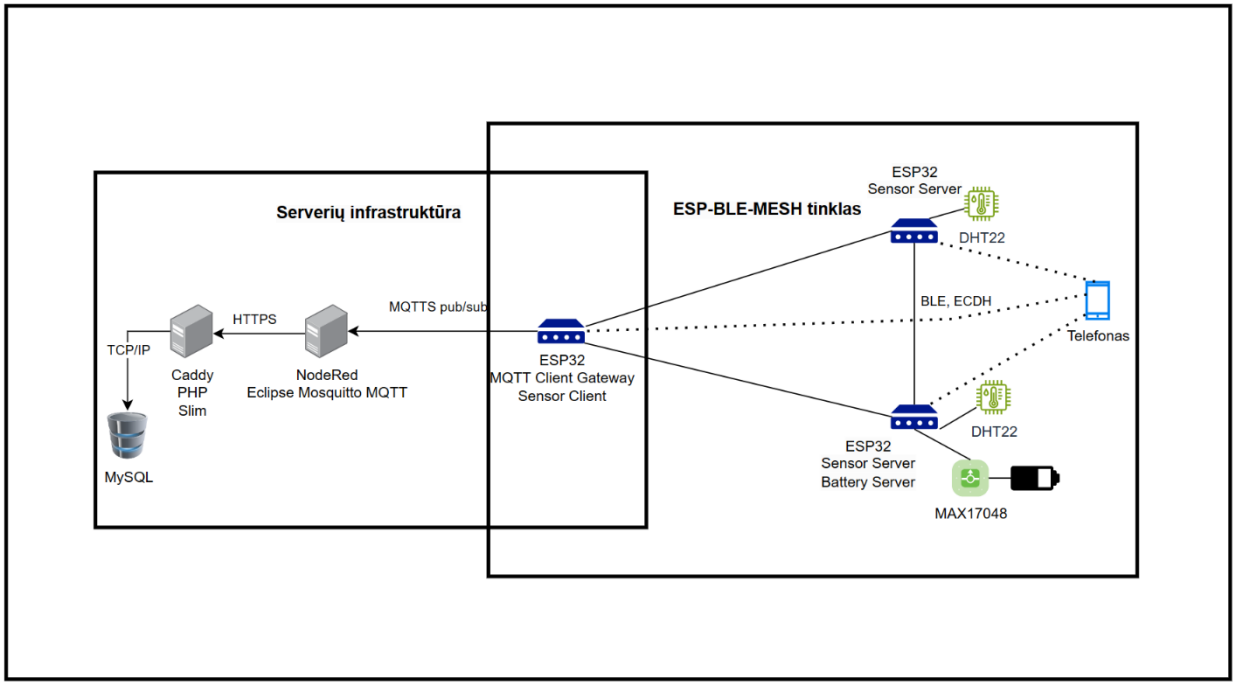
10 pav. Atliekamų pakeitimų diagrama ESP BLE MESH architektūroje

2.3. Išvados

ESP-BLE-MESH naudojamas AES-CCM autentifikuoto šifravimo algoritmas tinklo sluoksnyje negali būti pakeistas *ChaCha20-Poly1305* algoritmu dėl per ilgos autentifikacijos žymės, kai įrenginiai palaiko tik seną BLE technologiją. BLE reklaminio paketo naudingosios apkrovos maksimalus ilgis yra 31 baitas, autentifikacijos žymė užima 16 baitų, ESP-BLE-MESH kontekstinė informacija būtina užtikrinti protokolo veikimą užima 13 baitų, aplikacijos lygio duomenims liktų tik 2 baitai. Perduoti 10 baitų aplikacijos lygio duomenų reiktų išsiųsti ne 3 žinutes, o 13.

3. Daiktų interneto įrenginių tinklo realizacija

Realizuotą DI įrenginių tinklą sudaro dvi pagrindinės dalys: ESP-BLE-MESH potinklis ir serverių infrastruktūra. ESP-BLE-MESH potinklis susideda iš ESP32 mikrovaldiklių, kurie veikia jau apžvelgto ESP-BLE-MESH protokolų rinkinio pagrindu. Serverių infrastruktūra sudaro Node-RED tarpinis serveris, kuriame taip pat yra įdiegtas Mosquitto MQTT brokeris, žiniatinklio serveris ir duomenų bazės serveris. DI įrenginių tinklo prototipo architektūros schema pateikta 11 pav..

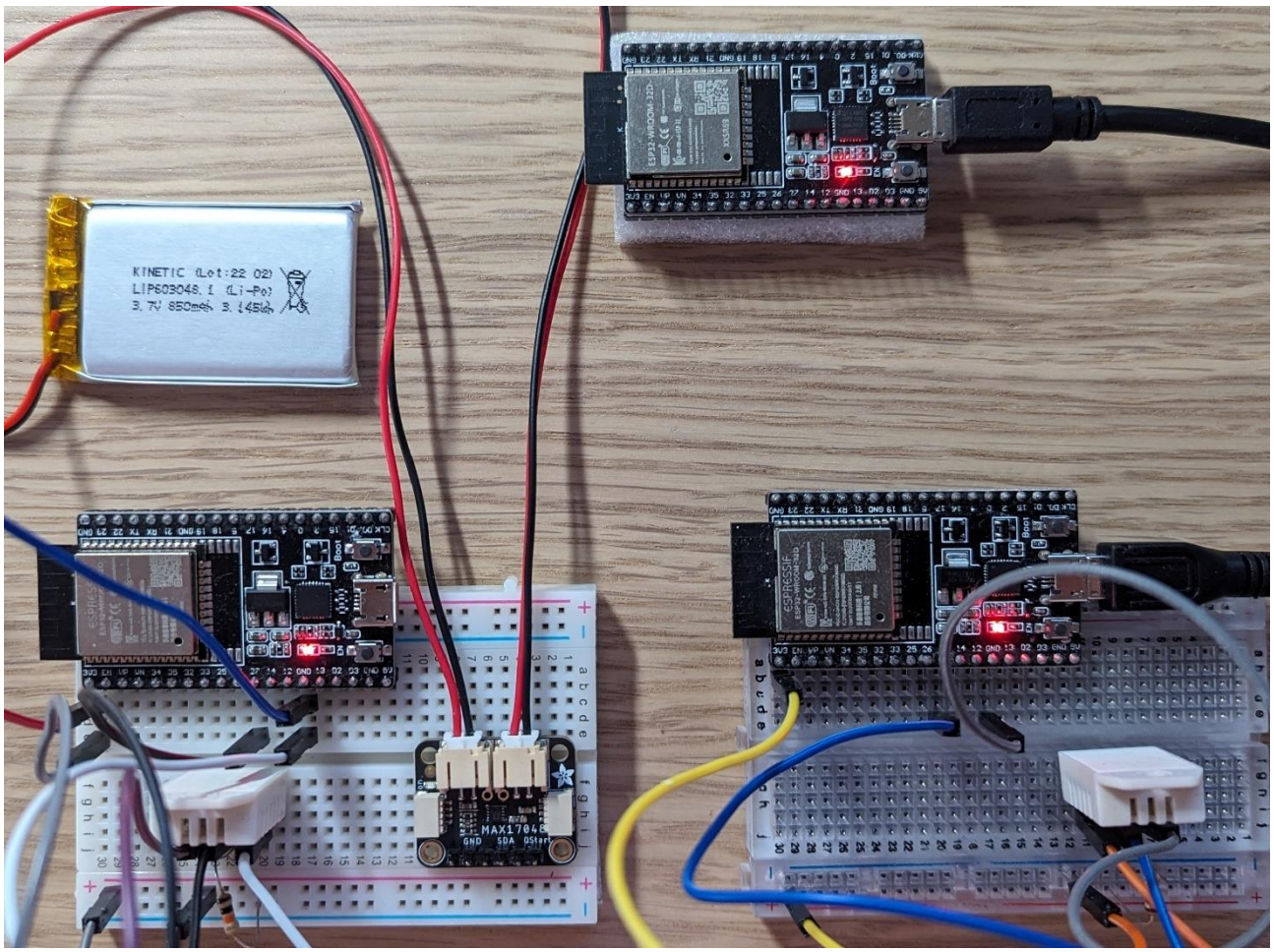


11 pav. Daiktų interneto įrenginių tinklo prototipo architektūros schema

3.1. ESP-BLE-MESH potinklis

Potinklis sudarytas iš 3 ESP32 mikrovaldiklių, kiekvienas iš jų naudoja ESP-BLE-MESH konfigūracijos serverio modelio sąsaja, kad galėtų būti konfigūruojamas. 2 mazgų architektūra paremta sensoriaus serverio modelio sąsaja. Visi mazgai potinklyje veikia kaip šakniniai mazgai pagal ESP-BLE-MESH topologiją (1.3.4.1). Serverio modelio sąsajos naudojamos įvairių sensorių duomenų nuskaitymui arba nustatymui, perdavimui į tinklą. Prototipinis DI įrenginių tinklo ESP-BLE-MESH potinklis matomas **12 pav.** .

Šliuzo mazgo architektūra paremta sensoriaus kliento modelio sąsaja, kad mazgas galėtų priimti sensorių duomenis siunčiamus tinkle mazgų, naudojančių serverio modelio sąsajas. Perduoti duomenis MQTT brokeriui pagal nustatytą filtrą galutinis mazgas naudoja MQTT klientą. Pagal tai į kokį kanalą ateina žinutė tarpinis serveris parenka atitinkamą veiksmų seką.



12 pav. Daiktų interneto įrenginių tinklo ESP-BLE-MESH potinklio prototipas

Kuriamas potinklis nuo įprasto ESP-BLE-MESH protokolų rinkiniu pagrįsto DI tinklo skiriasi antrame skyriuje pateiktu sprendimu – viršutiniame transporto sluoksnyje AEAD duomenų autentifikavimo ir šifravimo algoritmui naudojamas *ChaCha20-Poly1305* algoritmas. Tai reiškia, kad kiekviena žinutė perduodama tarp ESP32 mikrovaldiklių šiame tinkle yra šifruojama ir autentifikuojama pasitelkiant ne tik AES-CCM, bet ir *ChaCha20-Poly1305* algoritmą.

3.1.1. Tinklo konfigūracija

Tinklas konfigūruojamas naudojantis Nordic Seiconductor ASA sukurta mobiliąja aplikacija *nRF Mesh* [22], skirta lengvai prijungti ir konfigūruoti naujus mazgus tinkle, kurie palaiko *Bluetooth Mesh* specifikaciją. Programėlė supaprastina *Bluetooth Mesh* tinklo sudarymo ir konfigūravimo procesus, todėl galima grečiau sukurti prototipą. Kūrimo ciklo metu intuityvi sąsaja leidžia greitai konfigūruoti prie tinklo prijungtus mazgus.

3.1.1.1. Naujo įrenginio pridėjimas į tinklą

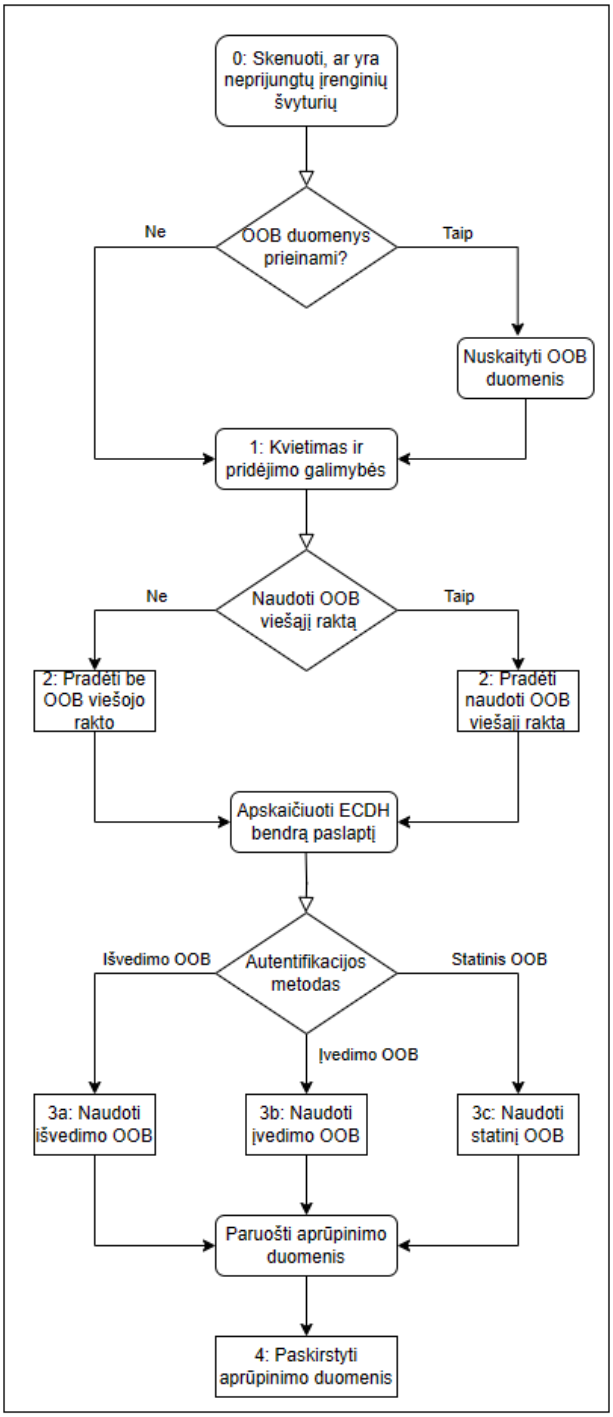
Neautentifikuoto mazgo pridėjimas į tinklą atliekamas naudojant penkių etapų procesą: signalizavimas, kvietimas, viešųjų raktų apsikeitimas, autentifikavimas ir aprūpinimo duomenų paskirstymas, kaip parodyta 13 pav. .

Įrenginys, kuris palaiko PB-ADV, ir nebuvo pridėtas prie tinklo ir šiuo metu nėra pridedamas, turi reklamuoti nepridėto įrenginio signalą. Šis signalizavimas gali rodyti OOB (angl. *Out-of-Band*)

duomenų prieinamumą, leidžiantį aplikacijai paraginti vartotoją įvesti šiuos OOB duomenis prieš kitą etapą.

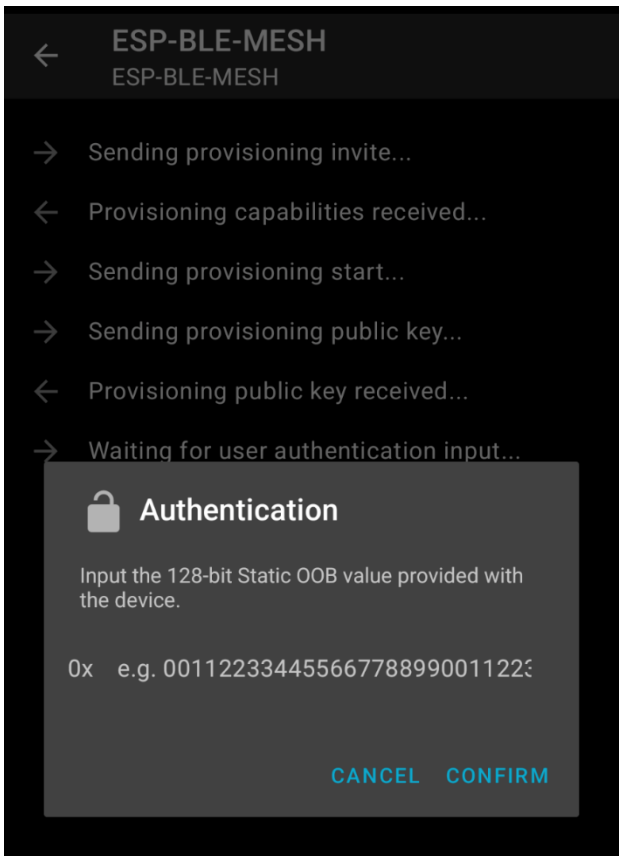
Kvietimo etape aplikacija siunčia kvietimo pridėti protokolo duomenų vienetą (angl. *Provisioning Invite PDU*), o įrenginys atsako atsiųsdamas pridėjimo galimybių protokolo duomenų vienetu (angl. *Provisioning Capabilities PDU*). Pridėjimo galimybių protokolo duomenų vienete įrenginys perduoda informacija apie tai kiek *Bluetooth Mesh* elementų palaiko, rinkinį palaikomų saugos algoritmų, viešojo rakto prieinamumą naudojant OOB technologiją, šio įrenginio galimybę išvesti vertę vartotojui, šio įrenginio galimybę leisti vartotojui įvesti vertę ir ar įrenginys turi OOB duomenų bloką, kurį galima naudoti autentifikavimui.

Viešųjų raktų apsikeitimo etape yra dvi galimybės, atsižvelgiant į tai, ar aplikacijai yra prieinamas įrenginio viešasis raktas. Kartu su trimis autentifikavimo žingsnio galimybėmis yra šeši galimi autentifikavimo keliai. Prototipe įrenginio viešasis raktas nėra prieinamas, todėl viešieji raktai yra sugeneruojami ir apsikeičiami tarp mobiliojo įrenginio ir mikrovaldiklio.



13 pav. Naujo mazgo pridėjimo į tinklą diagrama

Prototipe naudojamas statinis OOB metodas autentifikavimui. Metodas paremtas tuom, kad įrenginio programiniame kode yra įrašyti OOB duomenys, bandant pridėti įrenginį prie tinklo, reikia įvesti 128 bitų įrenginio OOB duomenyse nurodytą kodą 14 pav. . Toks OOB metodas autentifikacijai buvo pasirinktas dėl ribotos aparatinės įrangos, diegiant sistemą produkcinėje aplinkoje reikėtų pasirūpinti kažkurio iš įvesties arba išvesties OOB metodų palaikymu.



14 pav. *nRF Mesh* mobiliosios aplikacijos autentifikacijos kodo įvedimo forma bandant pridėti naują įrenginį prie ESP-BLE-MESH tinklo

Kai įrenginys yra autentifikuotas, programėlė ir įrenginys naudoja nustatytą Diffie-Hellman bendrą paslaptį (angl. *ECDHSecret*), kad apskaičiuoti įrenginio raktą (angl. *DevKey*). Įrenginio raktas naudojamas aprūpinimo duomenims užšifruoti ir autentifikuoti prieš siunčiant duomenis į mazgą. Įrenginys gavęs aprūpinimo duomenis, iššifruoja ir patvirtina gautus duomenis. Sėkmingai autentifikavus aprūpinimo duomenis, mikrovaldiklis išsaugo tinklo raktą (angl. *NetKey*) (identifikuojamą rakto indeksu), IV indeksą, IV atnaujinimo procedūros būseną pagal IV atnaujinimo vėliavėlę, rakto atnaujinimo fazę pagal raktą.

Tiesa, mazgo pridėjimo į tinklą procesą galima automatizuoti, įdiegus aprūpinimo mazgą, kuris naudotųsi konfigūracijos kliento modelio sąsaja ir neautentifikuotus mazgus pridėtų prie tinklo savaime. Visgi, tokia tinklo konfigūracija sumažina lankstumo galimybes prototipo stadijoje ir apsunkina tinklo kūrimo ir konfigūracijos procesą, ypač jeigu nėra praktinės patirties konfigūruojant ESP-BLE-MESH protokolų rinkinį naudojančio tinklo.

3.1.1.2. Duomenų perdavimas tinkle

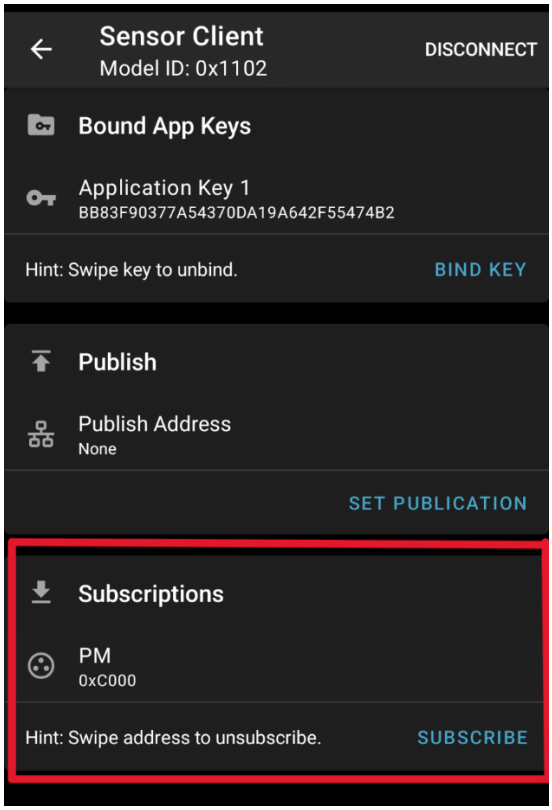
Duomenų perdavimas apima mazgų konfigūravimą, modelių nustatymą ir įvykių tvarkymą naudojant modelių API. Procesas prasideda nuo mazgų konfigūravimo ir inicijavimo ESP-BLE-MESH tinkle. Aplikacijos kode yra nustatomi konkretūs parametrai, kad pritaikyti mazgo elgseną. Inicijavimas apima bazinių mazgo komponentų inicializavimą (pastovios saugyklos, *Bluetooth*, lokalaus bevielio tinklo, *ble-mesh* bibliotekos).

Toliau mazgams reikia nustatyti modelius pagal norimo funkcionalumo palaikymą. Tai apima modelių inicijavimą ir registravimą publikavimo ar prenumeratos tikslais. Modelių registravimui

reikia numatyti kokie modelio įvykiai turi iššaukti atitinkamus veiksmus ir juos implementuoti naudojantis modelių taikomųjų programų programavimo sąsajomis.

Svarbus proceso aspektas yra įvykių tvarkymas, kai mazgai įgyvendina funkcijas, skirtas reaguoti į pokyčius, gauti duomenis arba suaktyvinti veiksmus, pagrįstus konkrečiais įvykiais. Pavyzdžiui, prototipe naudojamas sensoriaus serverio modelis turi įvykių tvarkymo funkciją, kuri reaguoja į publikavimo įvykį ir jo metu nuskaito sensoriaus duomenis prieš atlikdamas publikaciją. Įvykių tvarkymo užtikrinimo žingsnis leidžia mazgams dinamiškai prisitaikyti prie besikeičiančių sąlygų tinkle.

Aprūpinti atitinkamais modeliais mazgai gali dalyvauti prenumeratos ir publikavimo veikloje. Prenumeruodami konkrečias grupes mazgai išreiškia savo susidomėjimą gauti duomenis, o duomenų perdavimas apima duomenų publikavimą interesų grupėms. Tiesa, mazge užregistruoti modeliai turi būti sukonfigūruoti per *nRF Mesh* mobiliąją aplikaciją arba konfigūracija turi būti nustatyta aprūpinimo mazgo, nurodant kokiai adresų grupei publikuoti duomenis arba kokios adresų grupės duomenis mazgas norėtų gauti 15 pav. . Prototipe aprašytas sensoriaus kliento modelis turi įvykių tvarkymo funkciją, kuri reaguoja į publikavimo įvykį ir gautus duomenis perduoda MQTT brokeriui.



15 pav. Mazgo prijungto prie tinklo sensoriaus kliento modelio konfigūracija. Raudonai pažymėtas elementas nurodo, kad klientas prenumeruoja „PM“ kanalo žinutes

3.2. Serverių infrastruktūra

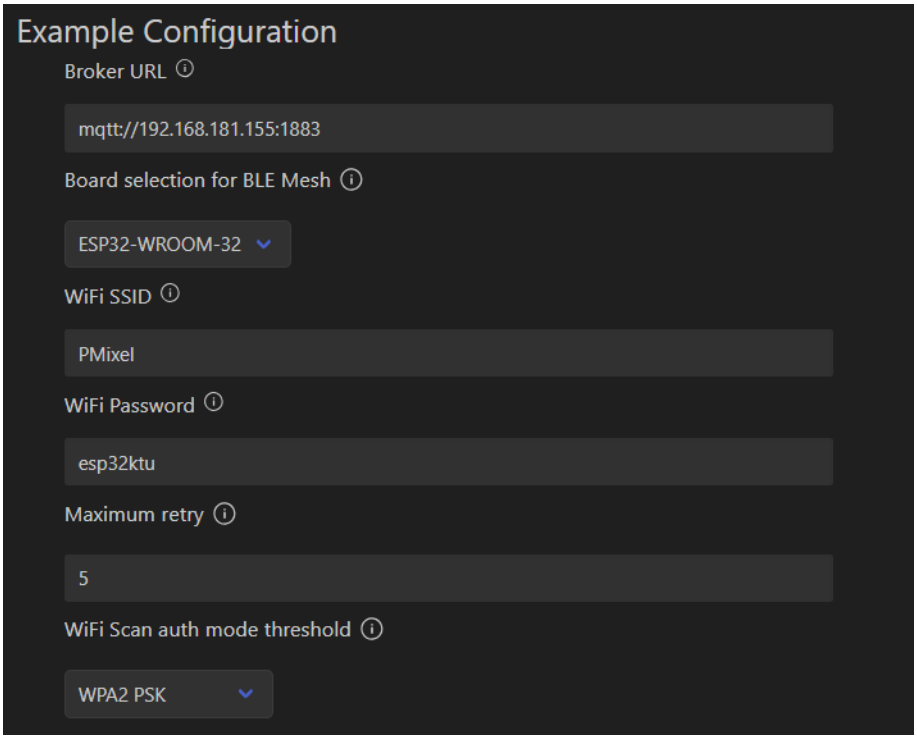
Duomenys iš ESP-BLE-MESH potinklio yra perduodami į Node-RED tarpinį serverį, kuriame yra įdiegtas Mosquitto MQTT brokeris, po duomenų apdorojimo, jie yra siunčiami į žiniatinklio serverį, kuris turi REST programinės įrangos architektūra paremtą sąsają. Perduoti duomenys yra validuojami, apdorojami ir išsaugomi duomenų bazėje.

3.2.1. Duomenų perdavimas tarp ESP-BLE-MESH potinklio ir serverių infrastruktūros

Duomenims tarp pagrindinių sistemos dalių perduoti yra naudojamas MQTTS protokolas [20]. MQTT (angl. *Message Queuing Telemetry Transport*), yra lengvas ir atviras pranešimų siuntimo protokolas, skirtas mažiems jutikliams ir mobiliesiems įrenginiams komunikuoti mažo pralaidumo, didelės delsos arba nepatikimuose tinkluose. Sukurtam prototipe MQTT naudoja TLS/SSL, kad būtų užtikrintas saugus ryšys.

MQTT protokolas veikia TCP protokolo pagrindu, todėl šliuzo mazgas, naudojantis MQTT klientą taip pat, turi būti prijungtas prie interneto. Sklandžiam mazgo veikimui naudojant tiek BLE, tiek vietinio belaidžio tinklo bevielius protokolus reikia naudoti skirsnių lentelę (angl. *partition table*). Skirsnių lentelė yra esminis komponentas ESP-BLE-MESH mazguose, kurie naudoja keletą skirtingų bevielių protokolų valdant išteklius, radijo dažnių kalibravimo duomenis ir konfigūracijos parametrus. Tai užtikrina, kad ESP32 mikrovaldiklis galėtų sklandžiai patenkinti abiejų ryšio protokolų poreikius be trukdžių, todėl gali patikimai ir efektyviai veikti įvairiose DI programose. Prototipe šliuzo mazge naudojama skirsnių lentelė pateikta 1 priede.

MQTT klientui mikrovaldiklyje naudojamas ESP-MQTT klientas, kurį galima panaudoti įtraukus *mqtt_client.h* biblioteką aplikacijos kode. Klientą reikia sukonfigūruoti, užregistruoti ir startuoti. Kliento konfigūracijai reikalingų duomenų įvedimui sukurta papildoma skiltis SDK konfigūracijos redaktoriuje 16 pav. , kuris supaprastina konfigūracijos valdymą suteikdamas intuityvesnę sąsają ir sumažindamas su konfigūracija susijusių klaidų tikimybę kuriant programinę įrangą. Tai tampa pagrindiniu tiesos šaltiniu, kuris taip pat gali sklandžiai dirbti su versijų valdymo sistemomis ir taip suteikia aiškesnę konfigūracijos pakeitimų apžvalgą. Tiesa, reiktų nepamiršti, kad produkcinės aplinkos autentifikavimo duomenų versijuoti nederėtų.



16 pav. Papildoma SDK konfigūracijos redaktoriaus skiltis MQTT kliento ir vietinio bevielio tinklo konfigūracijoms

ESP-BLE-MESH potinklyje perduodami duomenys yra jutiklių arba kita informacija gaunama iš tinklo įrenginių. Duomenys tinkle yra perduodami dvejetainiu formatu, o jų struktūrą apibrėžia *Bluetooth Mesh* specifikacija. Duomenų struktūrą gali sudaryti duomenų formatai, ilgiai ir savybių identifikaciniai numeriai. Perduodamų duomenų pavyzdys pateiktas **17 pav.** .

```
I (197165) Sensor Data: a4 0e a6 09 7c c4 0e d8 0e 00
I (197171) SENSOR CLIENT: Sensor Status, opcode 0x3ffbb008
I (197177) SENSOR CLIENT: Format A, length 0x02, Sensor Property ID 0x0075
I (197185) Sensor Data: a6 09 7c
I (197191) SENSOR CLIENT: Format A, length 0x02, Sensor Property ID 0x0076
I (197196) Sensor Data: d8 0e 00
I (197253) SENSOR CLIENT: Sensor client, event 2, addr 0x0019
I (197254) SENSOR CLIENT: ESP_BLE_MESH_SENSOR_CLIENT_PUBLISH_EVT
I (197256) SENSOR CLIENT: Sensor Status, opcode 0x0052
I (197262) SENSOR CLIENT: Sensor Status, data length: 10
I (197268) Sensor Data: a4 0e f6 09 fd c4 0e e2 13 00
I (197273) SENSOR CLIENT: Sensor Status, opcode 0x3ffbb008
I (197280) SENSOR CLIENT: Format A, length 0x02, Sensor Property ID 0x0075
I (197287) Sensor Data: f6 09 fd
I (197292) SENSOR CLIENT: Format A, length 0x02, Sensor Property ID 0x0076
I (197299) Sensor Data: e2 13 00
```

17 pav. Sensoriaus kliento ESP-BLE-MESH tinkle gauti duomenis atvaizduojami šešioliktainiu formatu

Prieš paskelbiant dvejetainius jutiklio duomenys į MQTT kanalą, duomenys yra konvertuojami į eilutę, kad būtų lengviau tvarkyti MQTT pranešimus. Nors pats MQTT neįpareigoja naudoti naudingosios apkrovos formato, jutiklio duomenų konvertavimas į eilutę atitinka įprastą praktiką, kai MQTT pranešimuose naudojami tekstiniai duomenys.

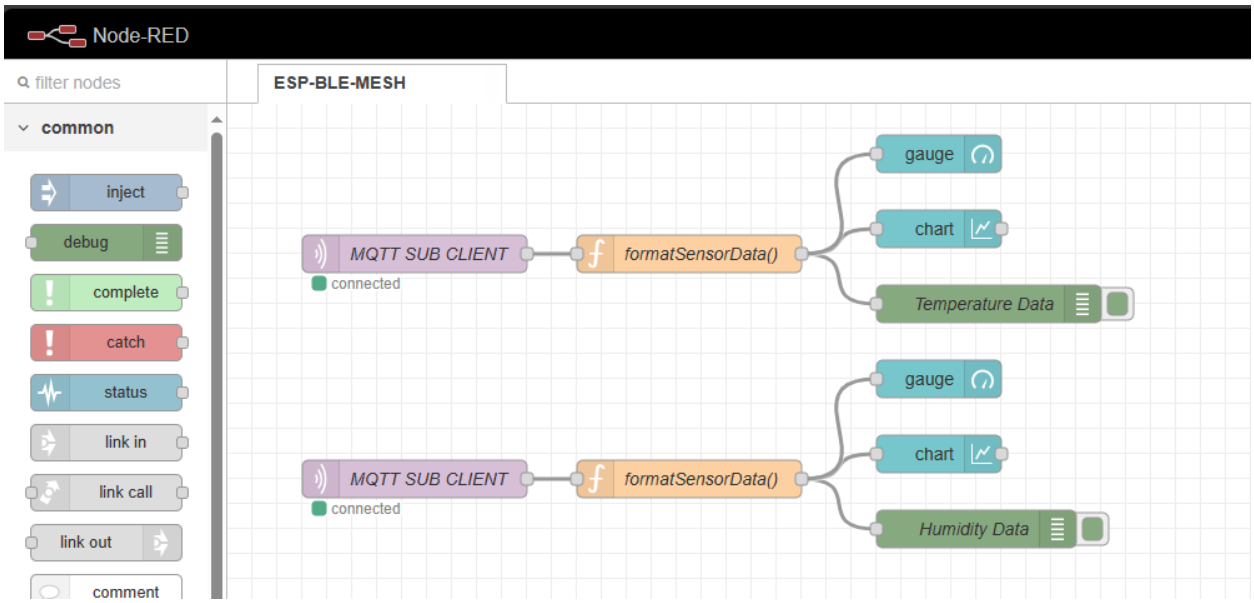
Jutiklio duomenų sėkmingam konvertavimui būtina išgauti duomenų formatą. Šis formatas naudojamas jutiklio duomenų struktūrai interpretuoti. Sukurtoje sistemoje vienas mazgas gali turėti daugiau negu vieną sensorių, o jų duomenys gali būti siunčiami supakuoti į vieną žinutę. Atitinkamai reikia žinoti tokią informaciją kaip duomenų ilgį, savybės identifikatorių ir tinklelio profilio identifikatorių (MPID), kuri išgaunama pagal jutiklio duomenų formatą. Atsižvelgiant į savybės identifikatorių jutiklio duomenys konvertuojami į eilutę ir paskelbiami skirtingose MQTT temose naudojant ESP-MQTT klientą.

3.2.2. Node-RED serverio funkcijos

Node-RED yra atviro kodo kūrimo priemonė, skirta vizualiai programuoti įvykiais pagrįstas programas. Serveris suteikia srauto (angl. *flow*) rengyklę naršyklėje, leidžiančią vartotojams vizualiai sujungti įrenginius, taikomųjų programų programavimo sąsajas ir internetines paslaugas. Prototipe šis serveris naudojamas sukūrti greitas sąsajas tarp skirtingų komponentų. Dėl savybės lengvai integruoti skirtingus klientus, paslaugas ir kurti žiniatinklio prietaisų skydelius duomenims vizualizuoti Node-RED serveris yra naudojamas kaip tarpinis serveris. Po kiekvieno atlikto veiksmo, sistemos būsena pasikeičia, tačiau pakeitimai matomi tik atlikus sistemos įdiegimą (angl. *deploy*).

Tarpinis Node-RED serveris diegiamas naudojant *Docker* platformą. Šiame serveryje yra įdiegtas MQTT brokeris, kuris yra tarpininkas tarp publikuojamų ESP-BLE-MESH potinklio duomenų ir Node-RED serveryje temas (angl. *topics*) prenumeruojančio kliento. Sensorių duomenys, kaip

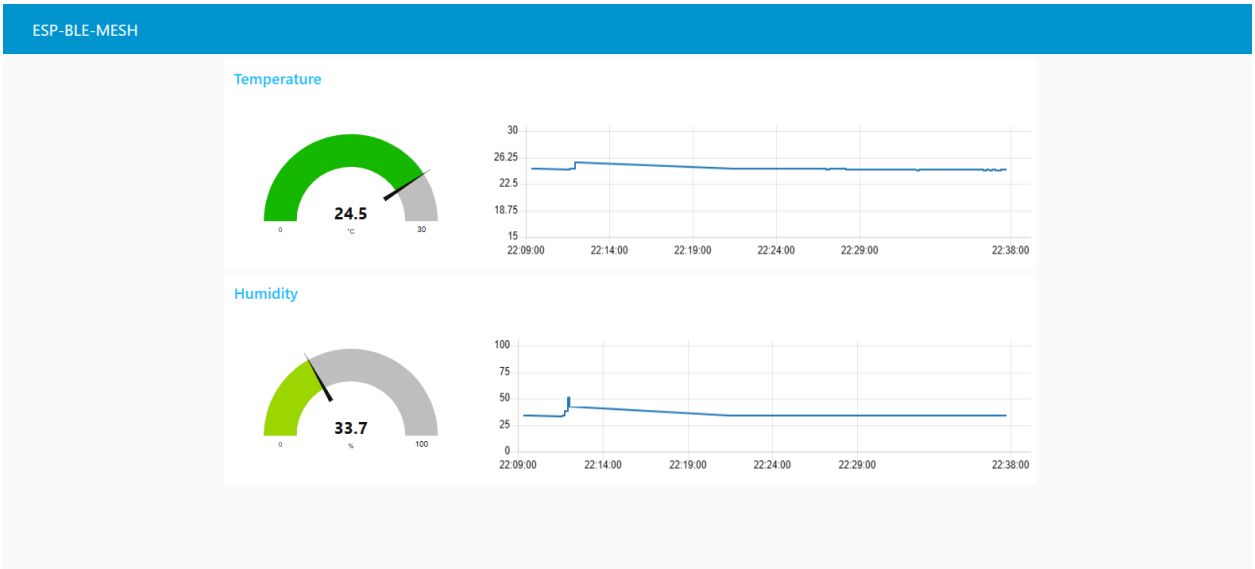
išdėstyta ankstesniame skyriuje, yra siunčiami naudojant MQTTS protokolą. Node-RED serverio srauto rengyklėje pirmiausią yra pasirenkamas MQTT kliento elementas pagal tai kokį veiksmą jis turės atlikti: publikuoti ar prenumeruoti. Prototipe yra pasirinktas kliento elementas, kuris prenumeruos temą, toliau jame nustatoma MQTT brokerio konfigūracija. Toliau grandinėles principu yra prijungiami elementai, kur kiekvienas iš jų yra sukonfigūruotas atlikti tam tikrus veiksmus. **18 pav.** pateiktas prototipo Node-RED serverio srautas.



18 pav. Node-RED serverio srauto rengyklė naršyklėje

Node-RED serveryje gauti duomenys yra konvertuojami ir išspausdinami į derinimo konsolę 19 pav. , tuo pačiu atnaujinami žiniatinklio puslapio elementai, kurie vaizduoja sensorių duomenis. Žiniatinklio puslapis pateiktas 20 pav. .

19 pav. Gautų MQTT kliento duomenų konvertavimas Node-RED serveryje



20 pav. Node-RED serverio sensoriaus duomenų atvaizdavimo žiniatinklio puslapis

3.2.3. Žiniatinklio serveris ir duomenų bazės serveris

Žiniatinklio serveris yra įdiegtas labiausiai orientuotas į REST taikomųjų programų programavimo sąsajos aptarnavimą. Serveryje naudojamas *Caddy* HTTP serveris kartu su PHP-FPM (angl. *FastCGI Process Manager*) PHP scenarijų tvarkymui. *Caddy* yra žinomas dėl savo paprastumo ir automatinio HTTPS palaikymo, supaprastina žiniatinklio serverio konfigūravimo ir apsaugos procesą. Jis puikiai tvarko atvirkštinio tarpinio serverio konfigūracijas, todėl puikiai tinka API užklausoms persiųsti į PHP-FPM procesą. PHP-FPM efektyviai valdo ir apdoroja PHP scenarijus, suteikdamas keičiamo dydžio ir didelio našumo vykdymo aplinką. *Caddy* konfigūraciją su PHP-FPM galimybėmis, suteikia galimybę sukurti saugią ir našią žiniatinklio serverio infrastruktūrą užtikrinant sklandų HTTP užklausų ir atsakymų tvarkymą.

Įtraukus *Slim* PHP karkasą į žiniatinklio serverio sąranką, kurioje *Caddy* naudojamas kaip HTTP serveris ir PHP-FPM PHP scenarijų tvarkymui, gaunama galinga ir supaprastinta aplinka, skirta kurti REST principais paremtą API. „Slim“ yra minimalistinio dizaino ir pabrėžto paprastumo PHP karkasas. *Slim* suteikia struktūrizuotą ir lengvą sistemą, skirtą maršrutams apibrėžti, tarpinei programinei įrangai tvarkyti ir API galutiniams taškams struktūrizuoti, taip supaprastinant REST paslaugų kūrimą. *Caddy* su automatiniu HTTPS palaikymu ir atvirkštinio tarpinio serverio galimybėmis efektyviai tarnauja kaip priekinis serveris, be vargo valdantis SSL/TLS sertifikatus. Tuo tarpu PHP-FPM efektyviai apdoroja PHP scenarijus vidinėje programoje. Kartu *Caddy*, PHP-FPM ir *Slim* derinys sukuria darnią žiniatinklio serverio infrastruktūrą, kuri puikiai tvarko API užklausas ir užtikrina saugią, našią aplinką kuriant REST principais paremtą API.

Naudojant žiniatinklio serverio REST API bus saugomi, skaitomi ir atnaujinami duomenys duomenų bazės serveryje. Duomenų bazės serveryje naudojama MySQL reliacinių duomenų bazių valdymo sistema, dirbanti SQL kalbos pagrindu.

Abu žiniatinklio ir duomenų bazės serveriai yra diegiami naudojant *Docker* platformą. Serveriai yra įdiegti skirtinguose konteineriuose, kurie yra prieinami iš išorės tik per atitinkamus prievadus. Duomenų bazės serveris yra pasiekiamas tik žiniatinklio serveriui, kadangi jie yra susieti. Detali *Docker* konfigūracija pateikiama **21 pav.**.

Duomenų bazės autentifikacijos duomenys yra saugomi atskirame faile, kuris nėra versijuojamas. Paleidžiant konteinerius autentifikavimo duomenys yra nuskaitomi iš failo ir užšifruojami, aplikacijoje paslaptys yra pasiekiamos „/run/secrets/{paslapties-pavadinimas}“. Prototipo konfigūracijoje paslaptis yra iš karto priskiriama aplinkos kintamajam.

21 pav. Žiniatinklio ir duomenų bazės serverio „Docker“ konfigūracija

3.3. Daiktų interneto tinklo kūrimui naudota programinė įranga

Kiekvienos sistemos dalies programinio kodo rašymui, konfigūravimui naudotas *Microsoft Visual Studio Code* kodo redaktorius su *Espressif IDF*, *CMake Tools*, *PHP Intelephense*, *CaddyFile Support* įskiepiais.

Sistemos programinio kodo versijų kontrolei naudota atviro kodo paskirstyto versijų valdymo sistema *Git*. Programinis kodas saugomas *GitHub* kūrėjų platformos saugyklose. Programiniam kodui talpinti yra sukurtos keturios saugyklos:

- Serverių infrastruktūros potinklio žiniatinklio ir duomenų bazės serverių programinis kodas yra talpinamas *SliMCaddy* projekto saugykloje. Saugykla pasiekiama adresu *https://github.com/MalakasP/SliMCaddy*.
- Tarpinio Node-RED serverio ir MQTT brokerio programinis kodas yra saugomas *https://github.com/MalakasP/NodeRed* saugykloje.

– ESP-BLE-MESH potinklio programos sluoksnio programinis kodas yra talpinamas *https://github.com/MalakasP/ESP_BLE_MESH_implementation* saugykloje.
– ESP-BLE-MESH protokolų rinkinio, kuriame atliktos modifikacijos, programinis kodas yra saugomas adresu *https://github.com/MalakasP/ESP-BLE-MESH-Core*.

3.4. Išvados

ESP-BLE-MESH protokolų rinkinys turi reikšminga mokymosi kreivę, dėl daug skirtingų koncepcijų, pradedant įvykiais pagrįstu programavimu, baigiant ESP mikrovaldikliais, ESP-IDF ir BLE. Protokolų rinkinio panaudojimas tinklelio tinklo kūrimui savaime užtikrina daug saugumo funkcijų, tačiau dėl sudėtingumo šis protokolų rinkinys neturėtų būti naudojamas smulkiems projektams.

Node-RED serveris dėl vizualinio požiūrio, plataus bibliotekų pasirinkimo ir integravimo galimybių, aktyvios bendruomenės palaikymo yra ypatingai patogus DI sistemos kūrimo įrankis, kurį lengvą įsidiegti „Docker“ konteineryje. Node-RED kartu su MQTT brokeriu padeda greitai sukonfigūruoti komunikaciją tarp DI įrenginių ir serverio ir tuo pačiu metu galima lengvai atvaizduoti duomenis žiniatinklio puslapyje naudojant skydelių prietaisų biblioteką.

4. Daiktų interneto duomenų perdavimo metodo įvertinimas ir eksperimentai

Sukurtas DI ESP-BLE-MESH potinklis su modifikuotu duomenų autentifikuoto šifravimo algoritmu bus lyginimas su potinkliu, kuris paremtas nemodifikuotu ESP-BLE-MESH protokolų rinkiniu. Tyrimą sudaro kokybinis tinklo palyginimas ir eksperimentinis tyrimas įrenginių naudojamos energijos našumui palyginti.

4.1. Kokybinis daiktų interneto tinklo palyginimas

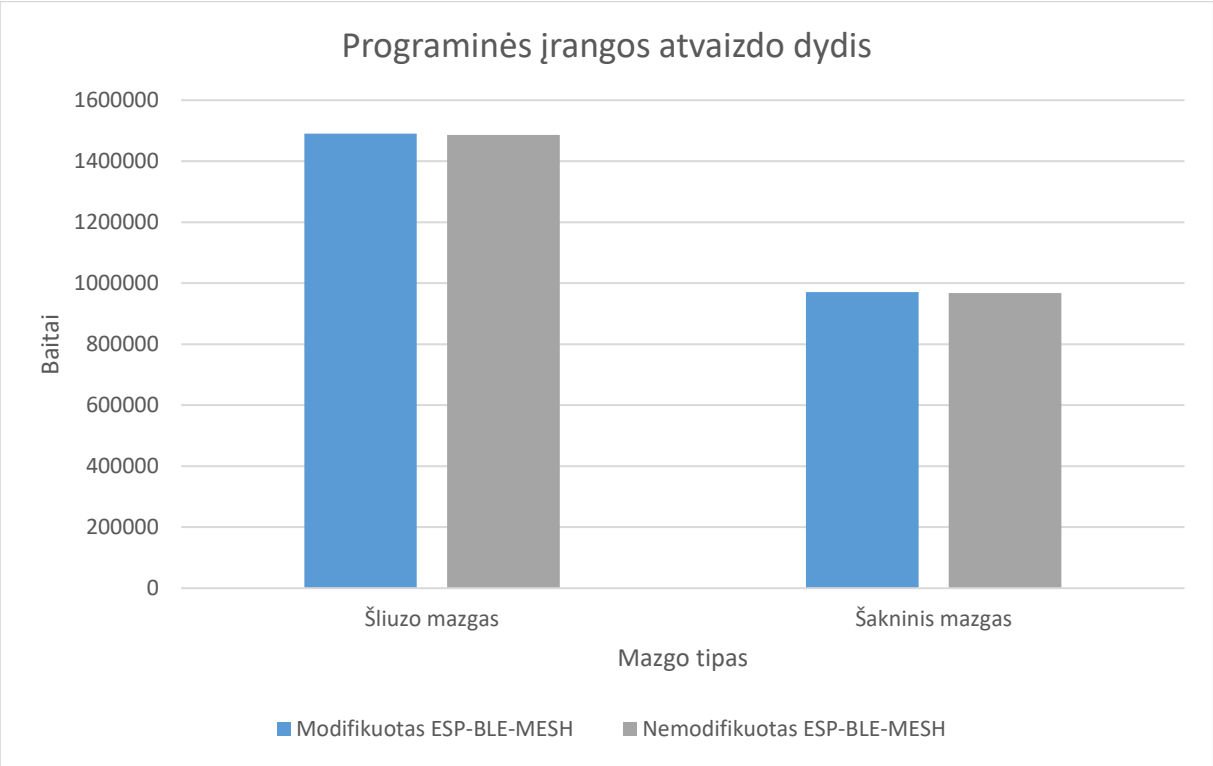
Pagrindis skirtumas tarp lyginamų protokolų rinkinų yra AES-CCM algoritmo pakeitimas į *ChaCha20-Poly1305* viršutiniame ESP-BLE-MESH tinklo transportavimo lygmenyje. Modifikacija suteikia keletą teorinių privalumų.

Nors AES šifravimo algoritmas yra plačiai laikomas saugiu, technologinė pažanga, nauji kriptoanalizės metodai ir atsitiktinumas gali susilpninti jo saugumą ateityje. *ChaCha20-Poly1305* siūlo kitokį kriptografinį pagrindą, kuris gali suteikti papildomo atsparumo būsimoms grėsmėms, kurios iškiltų, jeigu AES-CCM algoritmas būtų sukompromituotas. ESP-BLE-MESH DI tinkle, kuriame būtų naudojamas *ChaCha20-Poly1305* algoritmas viršutiniame transportavimo sluoksnyje, aplikacijos lygio duomenys išliktų konfidencialūs. Stabilaus tinklo funkcionalumo nebūtų galima užtikrinti, jeigu AES-CCM atsirastų pažeidžiamumas, tačiau svarbiausi duomenys liktų neatskleisti.

Mažiau svarbus aspektas yra sumažintas infrastruktūros sudėtingumas. *ChaCha20-Poly1305* paprastai laikomas mažiau sudėtingu nei AES-CCM dėl jo sudedamųjų algoritmų paprastumo, supaprastinto dizaino, trumpesnio kodo ilgio ir programinės įrangos diegimo efektyvumo. Kriptografinių algoritmų supaprastinimas gali sumažinti diegimo klaidų ir galimų pažeidžiamumų tikimybę. Todėl bendra tinklo saugumo padėtis gali būti pagerinta naudojant paprastesnį kriptografinį primityvą.

Visgi naudojant *ChaCha20-Poly1305* išauga žinučių ilgis dėl skirtingo dydžio autentifikacijos žymės. Dėl šio skirtumo naudojant AES-CCM algoritmą tinklo įrenginiai naudoja mažiau atminties šifruodami ir perduodami šifruotus duomenis tinkle. Skirtumas tarp autentifikacijos žymių yra vos 8-12 baitų, tačiau svarbu atsižvelgti į resursų naudojimą stengiantis optimizuoti ribotų resursų tinklų įrenginius.

Lyginant programinės įrangos atvaizdų dydžius pagal tinklo mazgo tipą prototipo mazgų atvaizdai užima nežymiai daugiau atminties **22 pav.** . Prototipo šliuzo mazgo programinės įrangos atvaizdo dydis yra 1490801 baitai. Nemodifikuoto ESP-BLE-MESH tinklo šliuzo mazgo programinės įrangos atvaizdo dydis yra 1486377 baitai. Detalesnė programinės įrangos atvaizdų dydžių informacija pateikia 2 ir 3 prieduose.



22 pav. Programinės įrangos atvaizdo dydžio palyginimas pagal mazgų tipus

Programinės įrangos atvaizdų dydžių skirtumas siekia 4,424 KB. Santykinis skirtumas procentaliai 1490,801 KB dydžio failo sudaro maždaug 0,3% viso failo dydžio. Šakninio mazgo atvaizdų dydžių santykinis skirtumas siekia 0,38% viso failo dydžio, kadangi programinės įrangos atvaizdo dydis yra mažesnis, užima 967565 baitus, daugiau informacijos 5 priede.

MbedTLS bibliotekos *Chacha20-Poly1305* kodo failų atvaizdų dydis siekia 3,395 KB (6 priedas). Galima teigti, kad programinės įrangos atvaizdų santykinis skirtumas yra nereikšmingas ir nesiekia 0.4% viso atvaizdo dydžio.

4.2. Eksperimentiniai daiktų interneto tinklo duomenų perdavimo metodo rezultatai

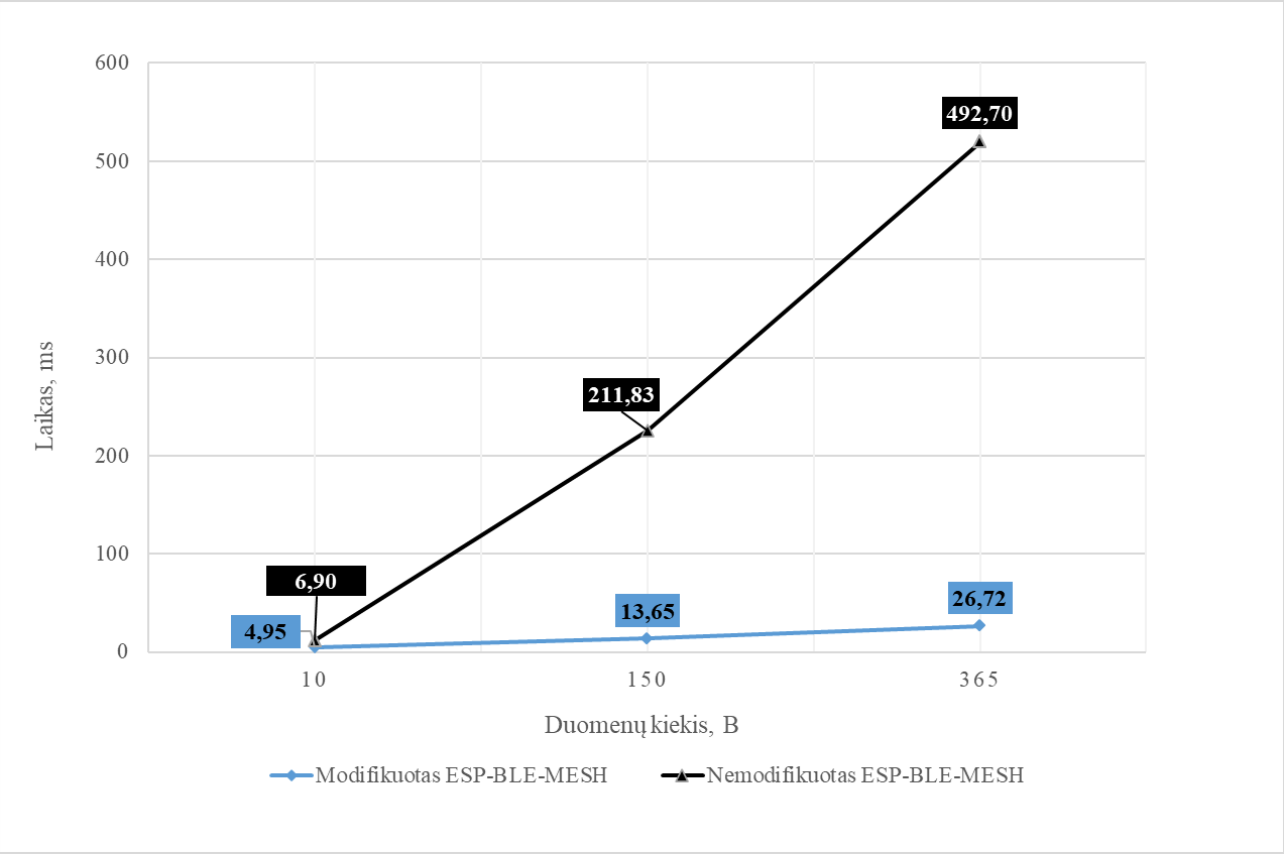
Buvo įgyvendinti du skirtingi bandymai, kad būtų galima įvertinti DI duomenų perdavimo metodo našumą ir energijos suvartojimą. Atliekant eksperimentus buvo lyginami DI ESP-BLE-MESH tinklo įrenginiai naudojantys modifikuotą ir nemodifikuotą ESP-BLE-MESH protokolų rinkinius. Eksperimentinių duomenų surinkimui buvo naudojami programos žurnalai, pildomi naudojant USB-UART tiltą esantį kūrimo plokštėje.

4.2.1. Duomenų publikavimo laiko eksperimento rezultatai

Pirmuoju eksperimentu buvo siekiama išmatuoti laiką, per kurį mazgas su jutiklio serverio modeliu perduoda duomenis į ESP-BLE-MESH tinklą. Naudodami programos lygio sekimą ir žurnalus, duomenų perdavimo laikai buvo kruopščiai įrašyti ir perduoti į pagrindinį kompiuterį prie kurio buvo prijungtas įrenginys. 23 pav. pateikiami laiko, reikalingo mazgams perduoti duomenis į tinklą naudojant modifikuotą ir nemodifikuotą ESP-BLE-MESH protokolų rinkinius su aparatinės įrangos spartinimo funkcija AES algoritmams, vidurkiai. 24 pav. lyginami tokie patys protokolų rinkiniai, tačiau be aparatinės įrangos spartinimo funkcijos AES šifravimo algoritmams. Eksperimente siunčiamų žinučių ilgiai buvo 10, 150 ir 365 baitų dydžio. 10 baitų yra standartinis 2 sensorių

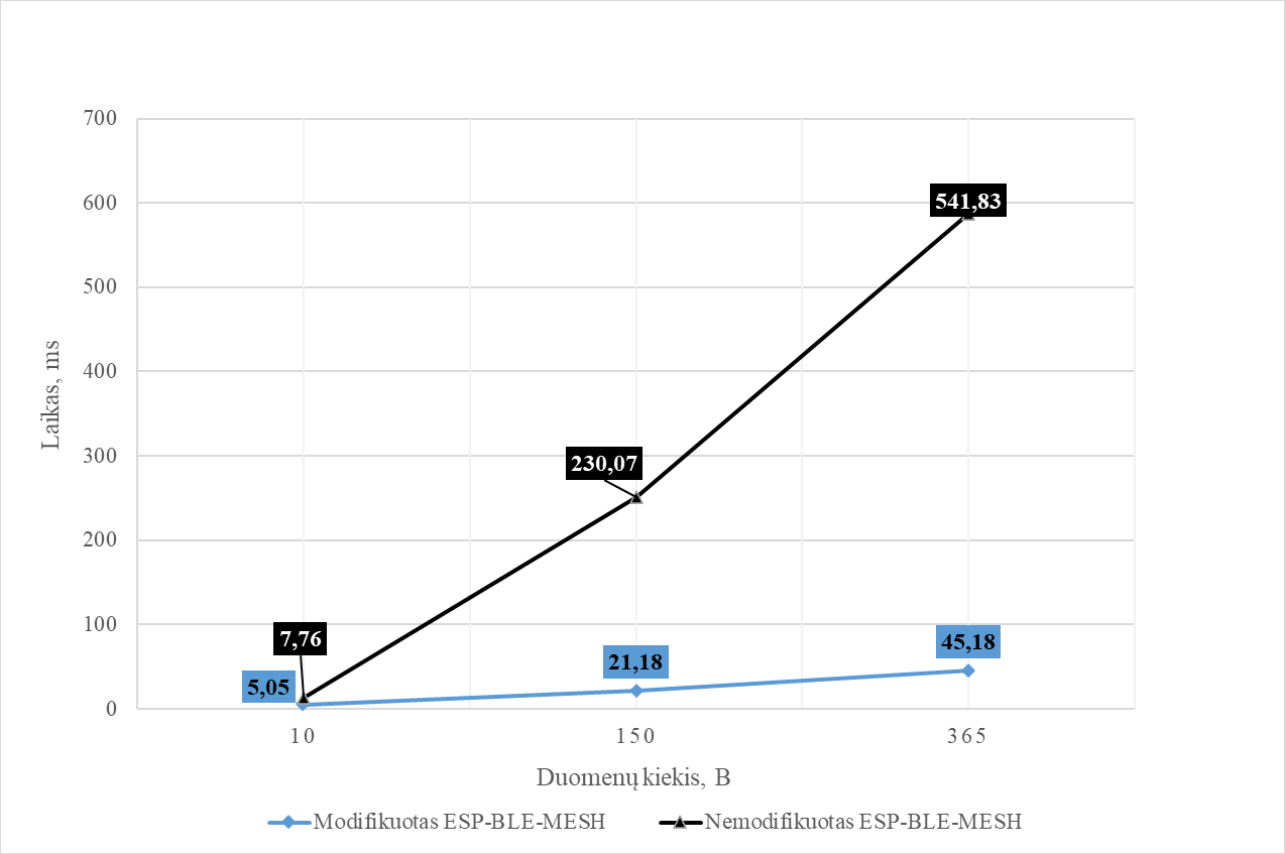
duomenų dydis pagal *Bluetooth Mesh* specifikaciją. 365 baitai yra maksimalus naudingosios apkrovos dydis vienai žinutei modifikuotame ESP-BLE-MESH protokolų rinkinyje. Galiausiai, paskutinis dydis buvo pasirinktas tarp 10 ir 365 baitų, maždaug per vidurį.

Eksperimentui atlikti buvo sudaryti 6 testavimo variantai, protokolų rinkinio ir žinutės duomenų kiekio poros. Duomenys buvo surinkti publikuojant po 100 žinučių kiekvienu testavimo variantu. Eksperimento rezultatas yra vidutinis žinutės perdavimo laikas milisekundėmis.



23 pav. Tinklo įrenginių žinutės perdavimo laikas su aparatinės įrangos spartinimu

Iš gautų rezultatų matyti, kad naudojant modifikuotą protokolų rinkinį žinutės vidutinis perdavimo laikas yra mažesnis negu naudojant įprastą ESP-BLE-MESH protokolų rinkinį. Nors vidutinis perdavimo laikas siunčiant 10 baitų dydžio žinutę labai nesiskiria, didinant duomenų kiekius jis stipriai padidėja. Siunčiant 365 baitų dydžio žinutę skirtumas siekia 18 kartų.

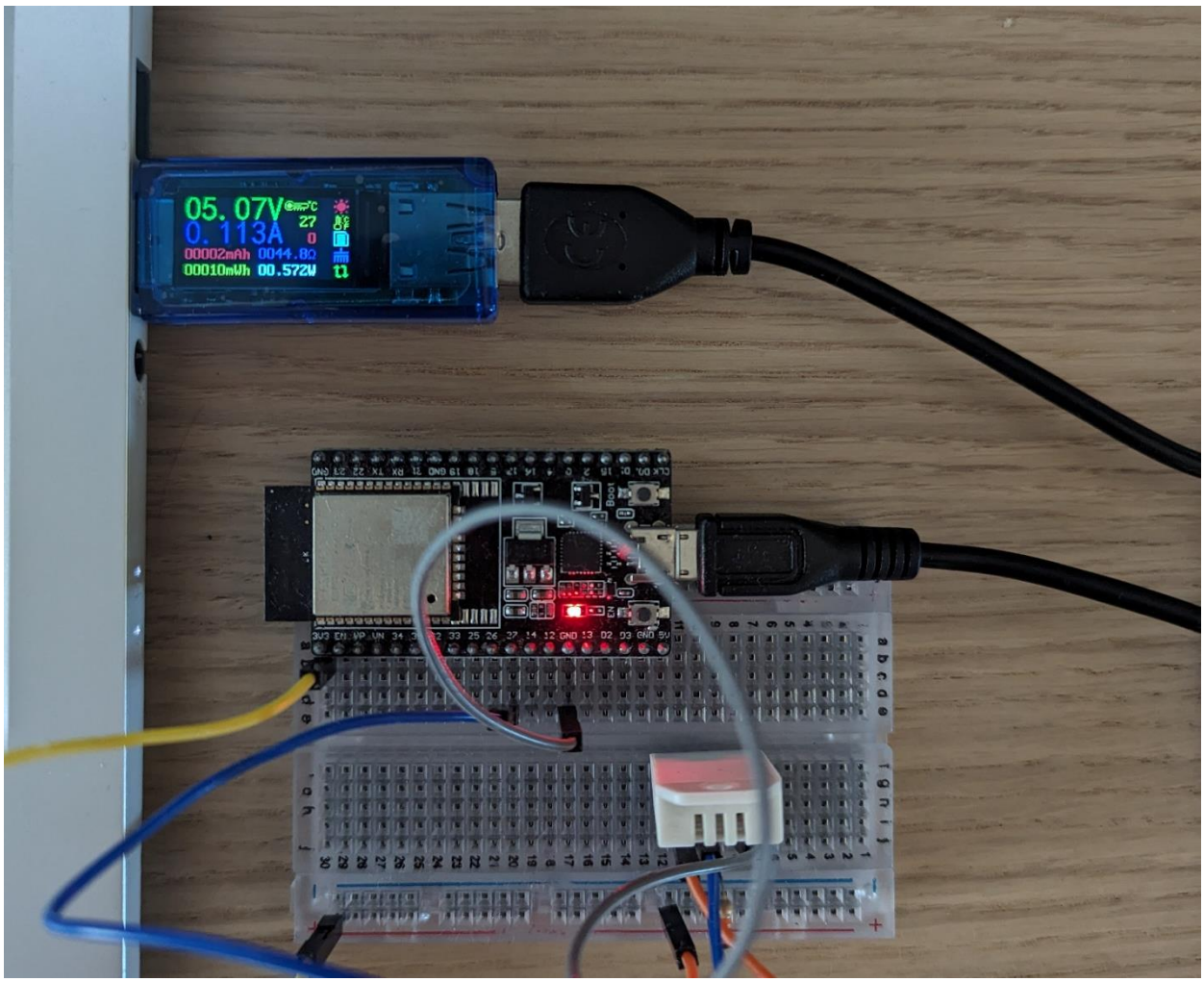


24 pav. Tinklo įrenginių žinutės perdavimo laikas be aparatinės įrangos spartinimo

Kaip ir tikėtasi, nenaudojant aparatinės įrangos spartinimo funkcijos AES šifravimo algoritmams žinutės perdavimo laikas išauga. Visgi taip pat kaip ir naudojant aparatinę įrangą spartinimui modifikuoto protokolų rinkinio vidutinis žinutės perdavimo laikas išlieka mažesnis. Skirtumas tarp vidutinių perdavimo laikų siunčiant žinutes išaugo, pvz. perduodant 365 baitų dydžio žinučių vidutinio perdavimo laiko skirtumas padidėjo apie 30 ms.

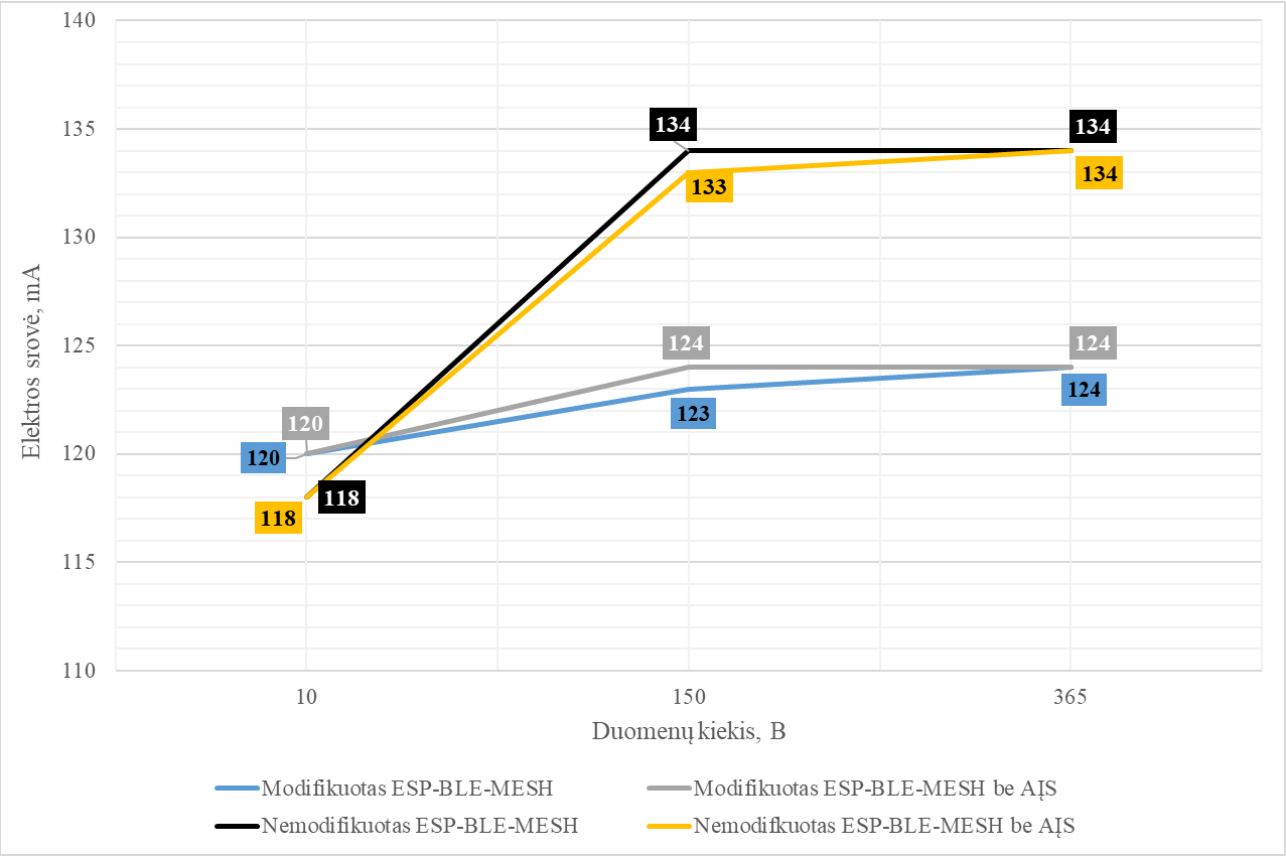
4.2.2. Elektros krūvio pokyčio eksperimentas

Antrasis eksperimentas buvo skirtas nustatyti maksimalią energijos srovę siunčiant žinutę. Duomenys buvo renkami iš jutiklio serverio mazgo, atsakingo už jutiklio duomenų rinkimą ir perdavimą į šliuzo mazgą. Naudojant JT-AT34 mažą, daugiafunkcinį USB multimetrą realiu laiku buvo renkami duomenys apie energijos suvartojimą, elektros srovę ir potencialą **25 pav.** .



25 pav. Energijos suvartojimo, elektros srovės ir potencialo matavimas atliekamas naudojant JT-AT34 USB multimetrą

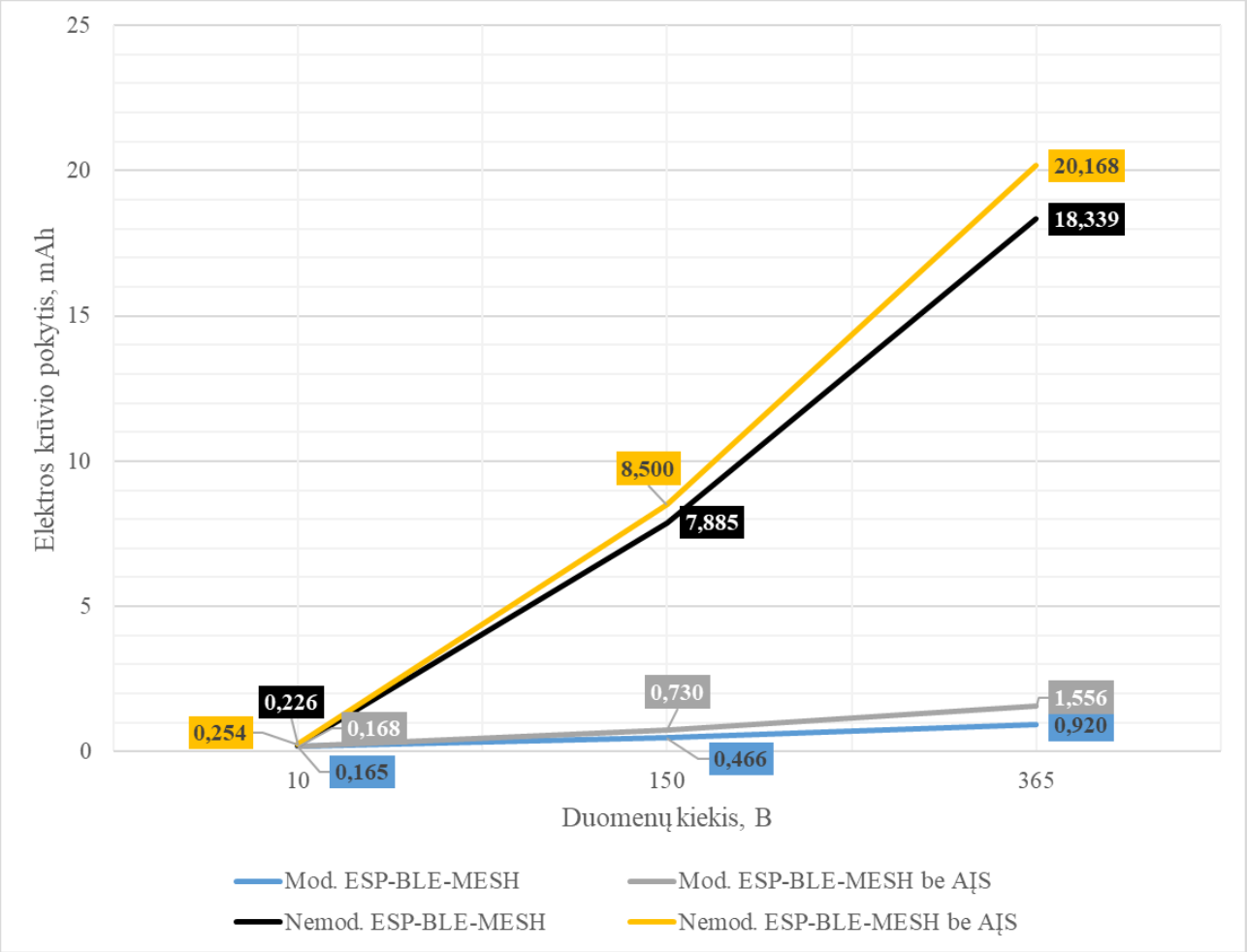
USB multimetro duomenų atnaujinimo dažnis yra 2 Hz. **26 pav.** matyti nustatyta maksimali srovė perduodant duomenis į tinklą iš jutiklio serverio mazgo naudojant pirmajame eksperimente tirtas protokolų rinkinio variacijas.



26 pav. Maksimali elektros srovė duomenų perdavimo metu jutiklio serverio mazge

Iš gautų rezultatų galime matyti, kad naudojant nemodifikuotą ESP-BLE-MESH protokolą įrenginio naudojama elektros srovė yra mažesnė siunčiant duomenis tik mažesniais kiekiais, pvz. 10 baitų aplikacijos duomenų. Siunčiant daugiau duomenų įreginio naudojama elektros srovė momentais viršija iki 10 mA lyginant su modifikuotą ESP-BLE-MESH protokolų rinkinį naudojančiu įrenginiu. Pagrindinė priežastis tokio skirtumo yra padidėjęs kiekis matematinių operacijų, reikalingų užšifruoti kiekvieną AES-CCM algoritme išskaidytą įvesties duomenų bloką.

Norint įvertinti siūlomo metodo energinį efektyvumą apskaičiavome koks elektros krūvio pokytis būtų pasiektas teoriškai jeigu žinutės publikavimo metu įrenginys naudotų maksimalią užfiksuotą elektros srovę. Elektros krūvio pokytis apskaičiuotas sudauginus žinutės perdavimo laiką (**23 pav.** , **24 pav.**) ir maksimalią srovę, užfiksuotą duomenų perdavimo į tinklą metu. Gautas elektros srovės kitimo greitis miliamperais per milisekundę konvertuotas į dažniau naudojamą vienetą miliampervalandes. Skaičiavimų rezultatai pateikti **27 pav.**.

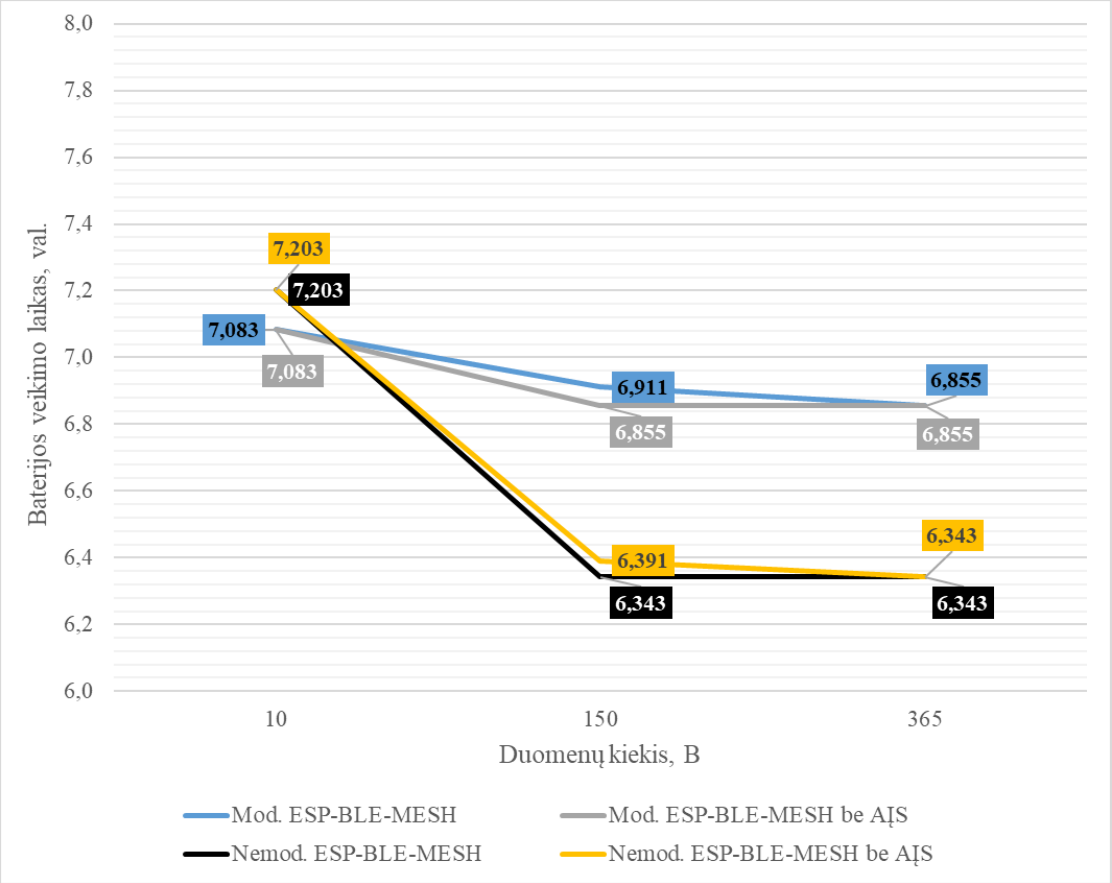


27 pav. Elektros krūvio pokytis perduodant duomenis į tinklą

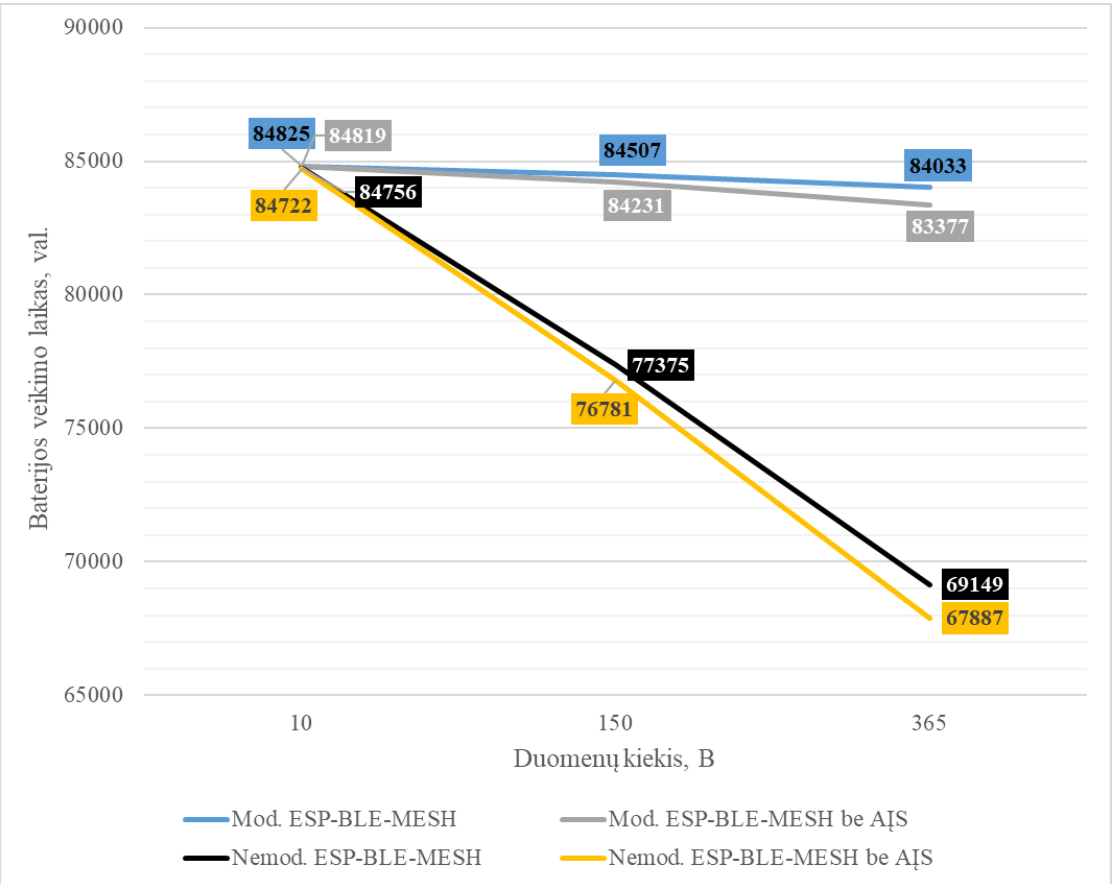
Iš rezultatų matyti, kad modifikuotas ESP-BLE-MESH protokolas sunaudoja mažiau energijos, nepriklausomai nuo to ar naudojamas aparatinės įrangos spartinimas. Siunčiant trumpesnes žinutes iki 10 baitų elektros krūvio pokyčio skirtumas nėra toks didelis lyginant su ilgesnėmis žinutėmis, tačiau reikšmingas. Duomenų kiekiui didėjant, nemodifikuoto ESP-BLE-MESH protokolo elektros krūvio pokyčio skirtumas tarp protokolų viršija daugiau nei 10 kartų.

Pagal didžiausią elektros srovės vertę perduodant duomenis į tinklą, apskaičiavome 850 mAh baterijos veikimo laiką idealiomis sąlygomis, kai duomenys į tinklą yra perduodami nuolat be pertraukų. Rezultatai gauti baterijos elektros krūvį dalinant iš maksimalios užfiksuotos elektros srovės duomenų perdavimo metu. Baterijos veikimo laikas 28 pav. pateiktas valandomis. Taip pat, buvo apskaičiuotas 850 mAh baterijos veikimo laikas idealiomis sąlygomis kai duomenis įrenginys publikuoja 3 kartus per parą, o likusiu metu yra gilaus miego režime. ESP32 gilaus miego režimu lustas sunaudoja nuo 0,15 mA (kai įjungtas ULP koprocesorius) iki 10 µA [15]. Skaičiavimuose buvo pasirinkta naudoti 10 µA elektros srovę. Rezultatai gauti naudojantis *IOT Battery Life Calculator* internetiniu įrankiu[1]. Baterijos veikimo laikas **29 pav.** pateiktas valandomis, o 2 lentelė baterijos veikimo laiko skirtumai lyginant modifikuotą ir nemodifikuotą protokolų rinkinius, su ir be aparatinės įrangos spartinimo.

[1] Įrankis pasiekiamas adresu *https://www.of-things.de/battery-life-calculator.php*.



28 pav. 850mAh baterijos veikimo laikas nuolat publikuojant duomenis



29 pav. 850mAh baterijos veikimo laikas duomenis publikuojant 3 kartus per parą

2 lentelė. 850mAh baterijos veikimo laiko skirtumai pagal duomenų kiekį duomenis publikuojant 3 kartus per parą

Protokolų rinkinys \ Duomenų kiekis	**10 B**	**150 B**	**365 B**
Mod. ESP-BLE-MESH	-	-	-
Mod. ESP-BLE-MESH be AĮS	6 val.	11d. 12 val. (276 val.)	27 d. 8 val. (656 val.)
Nemod. ESP-BLE-MESH	2 d. 21 val. (69 val.)	297 d. 4 val. (7132 val.)	620 d. 4 val. (14884 val.)
Nemod. ESP-BLE-MESH be AĮS	4 d. 7 val. (103 val.)	321 d. 22 val. (7726 val.)	672 d. 18 val. (16146 val.)

28 pav. matomi rezultatai neparodo didelio skirtumo baterijos veikimo laiko atžvilgiu, skirtumas tarp laikų nesiekia valandos. Visgi, vien tik iš šio grafiko ir skaičiavimų negalima nustatyti, kiek žinučių buvo išsiųsta naudojant tam tikrą protokolą. Išsiustų žinučių skaičių skirtumas siekia pusantro milijono.

Rezultatai gauti įrenginiui duomenis perduodant tam tikrais laiko momentais rodo, kad modifikuotas ESP-BLE-MESH protokolas su aparatinės įrangos spartinimu užtikrina ilgiausią baterijos veikimo laiką. Daiktų interneto įrenginių tinkle, kuriame mazgai dalinasi didesniais duomenų kiekiais negu 10 baitų, baterijos veikimo laiko skirtumas gali pasiekti 297 dienas (150 B) ir 620 dienų (365 B), jeigu įrenginiai naudoja aparatinės įrangos spartinimą, o duomenis įrenginys perduoda 3 kartus per parą. Matome, kad realiomis sąlygomis modifikuotas protokolų rinkinys yra pranašesnis energijos efektyvumo aspektu.

4.3. Daiktų interneto duomenų perdavimo metodo tyrimo išvados

Apibendrinant, galima daryti išvadą, kad pasiūlytas DI duomenų perdavimo metodas yra efektyvesnis energijos atžvilgiu ir teoriškai yra saugesnis negu standartinis ESP-BLE-MESH protokolų rinkinys, ypač perduodant didesnes negu 10 baitų žinutes. Pagrindinis pranašumas lemiantis tokius rezultatus yra duomenų publikavimo greitis, kuris yra 1,3, 15 ir 18 kartų greitesnis siunčiant 10, 150 ir 365 baitų duomenis atitinkamai. Tokie skirtumai gali prailginti 850mAh baterijos veikimo laiką virš mėnesio, jeigu įrenginys nuolat siųstų duomenis be pertraukų ir dar ilgiau, jeigu žinutės būtų siunčiamos momentais tarp įrenginio miego režimų.

Rezultatai ir išvados

1. Atlikus DI duomenų protokolų analizę galima teigti, kad DI tinkluose, kuriuose svarbu energijos vartojimo efektyvumas, saugumas, kuriems nėra keliami didelio diapozono reikalavimai ir svarbi maža įrenginių kaina, BLE gali būti geras pasirinkimas tinkamai sukonfigūravus tinklo įrenginius. BLE protokolas buvo pasirinktas, nes sunaudoja bent keletą kartų mažiau energijos nei MQTT, *ZigBee* ir LoRa, o įrenginių palaikančių protokolą kaina yra mažiausia kartu su MQTT.

2. ESP-BLE-MESH naudojamas AES-CCM autentifikuoto šifravimo algoritmas tinklo sluoksnyje negali būti pakeistas *ChaCha20-Poly1305* algoritmu dėl per ilgos autentifikacijos žymės, kai įrenginiai palaiko tik seną BLE technologiją. Perduoti 10 baitų aplikacijos lygio duomenų reikėtų išsiųsti 13 tinklo sluoksnio žinučių vietoje 3.

3. ESP-BLE-MESH protokolų rinkinys turi reikšminga mokymosi kreivę, dėl daug skirtingų koncepcijų ir technologijų, pradedant įvykiais pagrįstu programavimu, baigiant ESP mikrovaldikliais, ESP-IDF ir BLE. Protokolų rinkinio panaudojimas tinklelio tinklo kūrimui savaime užtikrina daug saugumo funkcijų, tačiau dėl sudėtingumo šis protokolų rinkinys neturėtų būti naudojamas smulkiems projektams.

4. Node-RED serveris dėl vizualinio požiūrio, plataus bibliotekų pasirinkimo ir integravimo galimybių, aktyvios bendruomenės palaikymo yra ypatingai patogus DI sistemos kūrimo įrankis, kurį lengva įsidiegti *Docker* konteineryje. Node-RED kartu su MQTT brokeriu leidžia greitai sukonfigūruoti komunikaciją tarp DI įrenginių ir serverio ir lengvai atvaizduoti duomenis žiniatinklio puslapyje naudojant skydelių prietaisų biblioteką.

5. Eksperimentiniai rezultatai rodo, kad naujas duomenų perdavimo metodas yra teoriškai saugesnė, greitesnė ir energijai našesnė ESP-BLE-MESH protokolo versija, ypač DI tinkluose, kuriuose žinutės ilgis yra didesnis negu 10 baitų. Siunčiant 365 baitų dydžio žinutę vidutinis siuntimo laikas yra mažesnis 18 kartų, elektros krūvio pokytis 19 kartų, o baterijos veikimo laikas ilgesnis 1,5 metų.

Literatūros šaltiniai

1. DANBATTA, S.J. - VAROL, A. Comparison of Zigbee, Z-Wave, Wi-Fi, and Bluetooth Wireless Technologies Used in Home Automation. In *2019 7th International Symposium on Digital Forensics and Security (ISDFS)* . 2019. p. 1–5. .

2. DE SANTIS, F. ir kt. ChaCha20-Poly1305 authenticated encryption for high-speed embedded IoT applications. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017* . 2017. p. 692–697. .

3. NIKOLOV, N. - NAKOV, O. Research of Secure Communication of Esp32 IoT Embedded System to.NET Core Cloud Structure using MQTTS SSL/TLS. In *2019 IEEE XXVIII International Scientific Conference Electronics (ET)* . 2019. p. 1–4. .

4. RAZA, U. ir kt. Low Power Wide Area Networks: An Overview. In *IEEE Communications Surveys & Tutorials* . 2017. Vol. 19, no. 2, p. 855–873. .

5. REZA PERMANA, I.M. ir kt. Comparative Analysis of Mesh and Star Topologies in Improving Smart Fire Alarms. In *2019 Fourth International Conference on Informatics and Computing (ICIC)* . 2019. p. 1–5. .

6. RUDD, S. - CUNNINGHAM, H. Selective privacy in IoT smart-farms for battery-powered device longevity. In *arXiv preprint arXiv:2108.02579* . 2021. .

7. SAMIE, F. ir kt. IoT technologies for embedded computing: A survey. In *2016 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ ISSS)* . [s.l.]: IEEE, 2016. p. 1–10. .

8. SANCHEZ-IBORRA, R. - CANO, M.-D. State of the Art in LP-WAN Solutions for Industrial IoT Services. In *Sensors* . 2016. Vol. 16, no. 5, p. 708. .

9. SINHA, R.S. ir kt. A survey on LPWA technology: LoRa and NB-IoT. In *ICT Express* . 2017. Vol. 3, no. 1, p. 14–21. .

10. TABISH, R. ir kt. A comparative analysis of BLE and 6LoWPAN for U-HealthCare applications. In *2013 7th IEEE GCC Conference and Exhibition (GCC)* . 2013. p. 286–291. .

11. Bluetooth 1.0 vs 2.0 vs 3.0 vs 4.0 vs 5.0 - How They Compare | Symmetry Blog. In *Symmetry Electronics* [interaktyvus]. [žiūrėta 2023-01-08]. Prieiga per internetą: <https://www.symmetryelectronics.com/blog/bluetooth-1-0-vs-2-0-vs-3-0-vs-4-0-vs-5-0-how-they-compare-symmetry-blog/>.

12. Cellular vs. WiFi: How Safe is Cellular Data? In *WilsonAmplifiers.com* [interaktyvus]. [žiūrėta 2023-01-07]. Prieiga per internetą: <https://www.wilsonamplifiers.com/blog/cellular-vs-wifi-how-safe-is-cellular-data/>.

13. Comparison of Internet of Things (IoT) Data Link Protocols. In [interaktyvus]. [žiūrėta 2023-01-14]. Prieiga per internetą: <https://www.cse.wustl.edu/~jain/cse570-15/ftp/iot_dlc/#sec2.2>.

14. ESP-IDF Programming Guide - ESP32-C3 - — ESP-IDF Programming Guide v4.3 documentation. In [interaktyvus]. [žiūrėta 2023-01-15]. Prieiga per internetą: <https://docs.espressif.com/projects/esp-idf/en/v4.3/esp32c3/index.html>.

15. Insight Into ESP32 Sleep Modes & Their Power Consumption. In *Last Minute Engineers* [interaktyvus]. 2018. [žiūrėta 2024-05-13]. Prieiga per internetą: <https://lastminuteengineers.com/esp32-sleep-modes-power-consumption/>.

16. Internet of Things Protocols and Standards. In [interaktyvus]. [žiūrėta 2023-01-07]. Prieiga per internetą: <https://www.cse.wustl.edu/~jain/cse570-15/ftp/iot_prot/>.

17. [Interaktyvus]. [žiūrėta 2023-01-10]. Prieiga per internetą: <https://hz1.37b.myftpupload.com/resource_hub/lorawan-is-secure-but-implementation-matters/>.

18. LTE-M - Implementing Low-Power Wide-Area Network (LPWAN) Solutions with AWS IoT. In [interaktyvus]. [žiūrėta 2023-01-11]. Prieiga per internetą: <https://docs.aws.amazon.com/whitepapers/latest/implementing-lpwan-solutions-with-aws/lte-m.html>.

19. LTE-M vs NB-IoT | EMnify Blog. In [interaktyvus]. [žiūrėta 2023-01-11]. Prieiga per internetą: <https://www.emnify.com/blog/lte-m-nb-iot>.

20. MQTT - The Standard for IoT Messaging. In [interaktyvus]. [žiūrėta 2024-01-14]. Prieiga per internetą: <https://mqtt.org/>.

21. NB-IoT - Implementing Low-Power Wide-Area Network (LPWAN) Solutions with AWS IoT. In [interaktyvus]. [žiūrėta 2023-01-10]. Prieiga per internetą: <https://docs.aws.amazon.com/whitepapers/latest/implementing-lpwan-solutions-with-aws/nb-iot.html>.

22. nRF Mesh - Apps on Google Play. In [interaktyvus]. [žiūrėta 2024-01-14]. Prieiga per internetą: <https://play.google.com/store/apps/details?id=no.nordicsemi.android.nrfmeshprovisioner&hl=en_US>.

23. [Interaktyvus]. . [žiūrėta 2023-01-10]. Prieiga per internetą: <https://www.gsma.com/iot/wp-content/uploads/2019/09/Security-Features-of-LTE-M-and-NB-IoT-Networks.pdf>.

24. [Interaktyvus]. 2022. [žiūrėta 2022-12-28]. Prieiga per internetą: <https://iot-analytics.com/number-connected-iot-devices/>.

25. Understanding Bluetooth Security. In *Decipher* [interaktyvus]. [žiūrėta 2023-01-14]. Prieiga per internetą: <https://duo.com/decipher/understanding-bluetooth-security>.

26. XChaCha20 Encryption vs AES-256: What's the Difference? In [interaktyvus]. [žiūrėta 2023-05-21]. Prieiga per internetą: <https://nordpass.com/blog/xchacha20-encryption-vs-aes-256/>.

27. Zigbee Security Vulnerabilities: Architecture and Security Issues. In *Payatu* [interaktyvus]. 2020. [žiūrėta 2023-01-08]. Prieiga per internetą: <https://payatu.com/blog/dattatray/zigbee-security-101>.

28. Z-Wave Security - Silicon Labs. In [interaktyvus]. [žiūrėta 2023-01-08]. Prieiga per internetą: <https://www.silabs.com/wireless/z-wave/specification/security>.

Priedai

1 priedas. Prototipo šliuzo mazge naudojama skirsnių lentelė.

```
# Name,   Type, SubType,  Offset,  Size, Flags
# Note: if you have increased the bootloader size, make sure to update the offsets to avoid overlap
nvs,      data, nvs,     0x9000,  16k
otadata,  data, ota,     0xd000,  8k
phy_init, data, phy,     0xf000,  4k
factory,  app,  factory, 0x10000, 2M
```

2 priedas. Modifikuoto ESP-BLE-MESH tinklo šliuzo mazgo programinės įrangos atvaizdo dydžio informacija.

```
Total sizes:
Used static DRAM:   79488 bytes (  45092 remain, 63.8% used)
      .data size:   24096 bytes
      .bss  size:   55392 bytes
Used static IRAM:  113662 bytes (  17410 remain, 86.7% used)
      .text size:  112635 bytes
   .vectors size:    1027 bytes
Used Flash size : 1353043 bytes
           .text: 1089851 bytes
         .rodata:  262936 bytes
Total image size: 1490801 bytes (.bin may be padded larger)
```

3 priedas. Nemodifikuoto ESP-BLE-MESH tinklo šliuzo mazgo programinės įrangos atvaizdo dydžio informacija.

```
Total sizes:
Used static DRAM:   79480 bytes (  45100 remain, 63.8% used)
      .data size:   24096 bytes
      .bss  size:   55384 bytes
Used static IRAM:  113662 bytes (  17410 remain, 86.7% used)
      .text size:  112635 bytes
   .vectors size:    1027 bytes
Used Flash size : 1348619 bytes
           .text: 1085787 bytes
         .rodata:  262576 bytes
Total image size: 1486377 bytes (.bin may be padded larger)
```

4 priedas. Modifikuoto ESP-BLE-MESH tinklo šakninio mazgo programinės įrangos atvaizdo dydžio informacija.

```
Total sizes:
Used static DRAM:   51004 bytes (  73576 remain, 40.9% used)
      .data size:   20148 bytes
      .bss  size:   30856 bytes
Used static IRAM:  104098 bytes (  26974 remain, 79.4% used)
      .text size:  103071 bytes
   .vectors size:    1027 bytes
Used Flash size :  846991 bytes
           .text:  662635 bytes
         .rodata:  184100 bytes
Total image size:  971237 bytes (.bin may be padded larger)
```

5 priedas. Nemodifikuoto ESP-BLE-MESH tinklo šakninio mazgo programinės įrangos atvaizdo dydžio informacija.

```
Total sizes:
Used static DRAM:   51004 bytes (  73576 remain, 40.9% used)
      .data size:   20148 bytes
      .bss  size:   30856 bytes
Used static IRAM:  104098 bytes (  26974 remain, 79.4% used)
      .text size:  103071 bytes
   .vectors size:    1027 bytes
Used Flash size :  843319 bytes
           .text:  658963 bytes
         .rodata:  184100 bytes
Total image size:  967565 bytes (.bin may be padded larger)
```

6 priedas. MbedTLS bibliotekos Chacha20-Poly1305 kodo failų atvaizdų dydžių informacija.

Object File	DRAM .data &	0.bss	IRAM0 .text &	0.vectors	ram_st_total	Flash .text &	.rodata &	.appdesc	flash_total
poly1305.c.obj	0	0	0	0	0	1608	0	0	1608
chacha20.c.obj	0	0	0	0	0	1030	0	0	1030
chachapoly.c.obj	0	0	0	0	0	757	0	0	757