УНИВЕРЗИТЕТ „ГОЦЕ ДЕЛЧЕВ“ - ШТИП
ФАКУЛТЕТ ЗА ИНФОРМАТИКА

Докторска дисертација

ПОДОБРУВАЊЕ НА БЕЗБЕДНОСТА НА IoT-КОМУНИКАЦИЈАТА СО АНАЛИЗА НА МРЕЖНИ СКРИЕНИ КАНАЛИ

м-р Александар Велинов

Штип, октомври 2021

Комисија за оценка и одбрана:

Интерен ментор: **проф. д-р Александра Милева,**
редовен професор на Факултет за информатика,
Универзитет „Гоце Делчев“ – Штип

Екстерен ментор: **проф. д-р Весна Димитрова,**
редовен професор на Факултет за информатички науки и компјутерско инженерство,
Универзитет „Св. Кирил и Методиј“ – Скопје

Членови на комисија за оценка и одбрана:

Претседател: **проф. д-р Зоран Здравев,**
редовен професор на Факултет за информатика,
Универзитет „Гоце Делчев“ - Штип

Член: **проф. д-р Александра Милева,**
редовен професор на Факултет за информатика,
Универзитет „Гоце Делчев“ - Штип

Член: **проф. д-р Весна Димитрова,**
редовен професор на Факултет за информатички науки и компјутерско инженерство,
Универзитет „Св. Кирил и Методиј“ - Скопје

Член: **проф. д-р Цвета Мартиновска Банде,**
редовен професор на Факултет за информатика,
Универзитет „Гоце Делчев“ - Штип

Член: **Проф. д-р Митко Богданоски,**
редовен професор на Воена академија „Генерал Михаило Апостолски“ - Скопје

Научно поле: **212 Компјутерска техника и информатика**
110 Информатика

Научна област: **21202 Информациони системи и мрежи**
11002 Информациони системи и програмирање

Датум на одбрана: 26.10.2021
Датум на промоција:

Посвета и благодарност

Би сакал најпрвин да изразам огромна благодарност до моето семејство за целосната помош и поддршка која ми ја даде во текот на овие докторски студии.

Би сакал да изразам огромна благодарност до мојата менторка, проф. д-р Александра Милева, за посветеноста, ангажираноста, поддршката и мотивацијата во целиот овој процес на истражувања, објавување трудови и подготовка на докторска дисертација. Вистински ментор!

Би сакал да изразам благодарност до проф. д-р Зоран Здравев, за поддршката и мотивацијата. Благодарност до проф. д-р Весна Димитрова затоа што прифати да биде екстерен ментор на мојата докторска дисертација. Благодарност до проф. д-р Цвета Мартиновска Банде, проф. д-р Доне Стојанов и до сите мои други колеги и соработници. Ви благодарам за поддршката, мотивацијата и разбирањето.

Би сакал да изразам и благодарност до двајцата професори од странство Prof. Steffen Wendzel, Prof. Wojciech Mazurczyk и до студентката на докторски студии Laura Hartmann.

Оваа докторска дисертација ја посветувам на мојот прерано починат татко, без кого никогаш не би бил овде. Оваа докторска дисертација ја посветувам на мојата мајка, затоа што секогаш беше до мене и ме поддржуваше. Оваа докторска дисертација ја посветувам на мојот брат, на внуката Ива и на сите останати членови од моето семејство, моите роднини, пријатели, колеги и соработници. Ви благодарам за сè!

Листа на рецензирани и објавени трудови произлезени од истражувањата:

[1] Mileva, Aleksandra and Velinov, Aleksandar and Stojanov, Done (2018) New Covert Channels in Internet of Things. In: The 12th International Conference on Emerging Security Information, Systems and Technologies - SECURWARE 2018, September 16-20, 2018, Venice, Italy.

[2] Velinov, Aleksandar and Mileva, Aleksandra and Stojanov, Done (2019) Power Consumption Analysis of the New Covert Channels in CoAP. International Journal On Advances in Security, 12 (1 & 2). pp. 42-52. ISSN 1942-2636.

[3] Velinov, Aleksandar and Mileva, Aleksandra and Wendzel, Steffen and Mazurczyk, Wojciech (2019) Covert Channels in the MQTT-based Internet of Things. IEEE Access, 7. pp. 161899-161915. ISSN 2169-3536, **IF(2019)=3.745**.

[4] Mileva, Aleksandra and Velinov, Aleksandar and Hartmann, Laura and Wendzel, Steffen and Mazurczyk, Wojciech (2021) Comprehensive analysis of MQTT 5.0 susceptibility to network covert channels. Computers & Security, 104 (102207). ISSN 0167-4048, **IF(2019)=3.579**.

[5] Velinov, Aleksandar and Stojanovic, Igor and Dimitrova, Vesna (2021) State-of-the-art Survey of Data Hiding in ECG signal. Balkan Journal of Applied Mathematics and Informatics, 4 (2), ISSN 2545-4803 (accepted)

ПОДОБРУВАЊЕ НА БЕЗБЕДНОСТА НА IoT-КОМУНИКАЦИЈАТА СО АНАЛИЗА НА МРЕЖНИ СКРИЕНИ КАНАЛИ

Краток извадок

Мрежното криење на информации е дисциплина која се занимава со пренесување на скриени податоци преку комуникациските мрежи и нивна детекција. Методите за криење преку легитимните податочни текови креираат скриени канали коишто овозможуваат тајно пренесување на податоци. Тие може да бидат применети за воена комуникација во непријателски средини, заобиколување на цензурирање, заштита на поткажувачи и новинари и сл. Од друга страна, скриените канали може да бидат злоупотребени за комуникација на терористи и криминалци, за индустриска шпионажа, за координирање DDoS-напади, за ширење злонамерен софтвер и сл.

Со цел за подобрување на безбедноста на комуникацијата кај интернет на нештата (IoT), во оваа докторска дисертација е прикажана анализа на скриени канали за протоколите на апликациско ниво кај IoT: CoAP, MQTT-верзија 3.1.1 и MQTT-верзија 5.0. За сите протоколи е направена соодветна категоризација на предложените канали со користење на шемите за мрежни скриени канали.

За протоколот CoAP откриени се вкупно 8 скриени канали од кои 6 се складирачки, а 2 временски. Дополнително, направена е експериментална евалуација на скриениот канал кој ги користи методите PUT и DELETE.

За протоколот MQTT-верзија 3.1.1 предложени се вкупно 13 скриени канали од кои 7 се директни додека 6 се индиректни. Направена е експериментална евалуација за скриениот канал кој користи подредување на теми и ажурирање со присуство или отсуство.

За протоколот MQTT-верзија 5.0 предложени се вкупно 23 скриени канали од кои 18 се директни, додека 5 се индиректни. Направена е експериментална евалуација на скриениот канал кој користи различни теми.

За евалуација на скриените канали кај MQTT-верзија 3.1.1 и верзија 5.0, користени беа три параметри: пропусен опсег, неоткривање и робусност.

Клучни зборови: интернет на нештата, скриени канали, шеми за скриени канали, CoAP, MQTT, MQTT-верзија 3.1.1, MQTT-верзија 5.0, објави-претплати.

SECURITY IMPROVEMENT OF THE IOT COMMUNICATION BY ANALYSIS OF NETWORK COVERT CHANNELS

Abstract

Network information hiding is a discipline that deals with the transmission of hidden data through communication networks and their detection. Hiding methods through legitimate data streams create covert channels that allow secret data transmission. They can be used for military communication in hostile environments, circumvention of censorship, protection of whistleblowers and journalists, etc. On the other hand, covert channels can be misused for communication between terrorists and criminals, for industrial espionage, for coordinating DDoS attacks, for spreading malware, and so on.

In order to improve the security of Internet of Things (IoT) communication, this doctoral dissertation presents an analysis of covert channels for IoT application layer protocols: CoAP, MQTT version 3.1.1 and MQTT version 5.0. Appropriate categorization of the proposed covert channels has been made for all protocols using network covert channels patterns.

For the CoAP protocol, a total of 8 covert channels were detected, 6 of which are storage and 2 are timing. Additionally, an experimental evaluation of the covert channel that use PUT and DELETE methods was performed.

For the MQTT version 3.1.1, a total of 13 covert channels have been proposed, of which 7 are direct while 6 are indirect. An experimental evaluation was performed for the covert channel that use topics ordering and updating with presence or absence.

For the MQTT version 5.0 protocol, a total of 23 hidden channels have been proposed, of which 18 are direct while 5 are indirect. An experimental evaluation of the covert channel that use different topics was performed.

For the evaluation of covert channels in MQTT version 3.1.1 and version 5.0, three parameters were used: bandwidth, undetectability and robustness.

Keywords: Internet of Things, covert channels, patterns for covert channels, CoAP, MQTT, MQTT version 3.1.1, MQTT version 5.0, publish-subscribe.

СОДРЖИНА

Листа на слики

List of Figures

Листа на табели

List of Tables

1. ВОВЕД

Мрежното криење на информации е дисциплина која се занимава со пренесување на скриени податоци преку комуникациските мрежи и нивна детекција. Методите за криење преку легитимните податочни текови креираат таканаречени скриени канали коишто овозможуваат тајно пренесување на податоци. Скриените канали биле воведени од Lampson во 1973 година како техники за комуникација со прекршување на безбедносните политики, нешто што не било предвидено од страна на дизајнерите на системите (Lampson, 1973). Тие може да бидат применети за воена комуникација во непријателски средини (Murdoch, 2007), заобиколување на цензурирање (Feamster et al., 2002), заштита на поткажувачи и новинари (Wendzel & Keller, 2014), за безбеден менаџмент на мрежна комуникација и сл. (Zander et al., 2007; Lucena et al., 2005; Forte et al., 2005), за заобиколување на ограничувањето за користење интернет во некои земји (пример: Infranet (Feamster et al., 2002)), обезбедување на квалитет на сервис за VoIP (Voice over Internet Protocol) сообраќај (Mazurczyk & Kotulski, 2006), вметнување воден жиг во мрежните потоци (пример: RAINBOW (Houmansadr et al., 2009)), следење напад на шифриран сообраќај или следење анонимни VoIP-повици (Wang & Reeves, 2003; Wang et al., 2005). Од друга страна, скриените канали може да бидат злоупотребени за комуникација на терористи и криминалци (Patel et al., 2007), индустриска шпионажа (Ulz et al., 2019), координирање DDoS-напади (Mehic et al., 2016), ширење злонамерен софтвер (Caviglione & Migliardi, 2018; Cabaj et al., 2018), споделување илегален материјал и сл. Jeremiah Denton, затвореник во Виетнамската војна, користел скриен канал за комуникација без другите засегнати лица да знаат за тоа (CEH: Certified Ethical Hacker, 2012). Тој бил интервјуиран за Јапонска телевизија и во видеото било забележано дека тој трепкал на невообичаен начин. Подоцна било откриено дека трепкањето во основа значело претставување на букви од Морзеовата азбука. На овој начин тој успеал да пренесе скриена порака и ова е реално сценарио за тоа како скриен канал може да се искористи за ова. Друг пример за пренесување на информации преку скриен канал е со користење на карактеристиките на документите наместо пренос на информациите во документите. На пример, може да се искористи големината на документот за пренесување скриена порака. Испраќачот доставува документ со соодветна

големина (на пр. 16B). Примателот го прима документот, ја утврдува големината и скриената порака (бројот 16). Скриените канали станале особено важна тема во воен контекст каде процесите со висока безбедност коишто имаат сензитивни информации (HIGH-процес) мора да бидат заштитени од протекување на информации до процеси со ниска сигурност („LOW“ процес) преку скриени канали (Ogurtsov et al., 1996). На пример, на „LOW“ процес којшто е класифициран како „SECRET“ не треба да му биде овозможено да пристапи до „TOP SECRET“ податоци на „HIGH“ процес (Wendzel et al., 2015). Ова е многу значајно во комуникацијата за воени цели кога се разменуваат сензитивни информации.

Сите погоре наведени случаи на користење само ја потврдуваат важноста на детекцијата на скриените канали. Развивањето на соодветни противмерки за заштита исто така е доста важно, посебно во случаите кога скриените канали се употребуваат за злонамерни цели.

1.1 Цели на истражувањето

Главната цел на ова истражување беше откривање нови мрежни скриени канали кај протоколите: CoAP, MQTT-верзија 3.1.1 и MQTT-верзија 5.0, кои се користат кај интернет на нештата (IoT). Покрај ова други цели беа следните:

- Да се направи соодветна експериментална евалуација на некои од предложените скриени канали за да се потврди нивната изводливост;
- Да се испита пропусниот опсег кој се обезбедува со даден имплементиран скриен канал;
- Да се согледа дали скриениот канал може лесно да се детектира или не;
- Да се испита робусноста и да се види како таа влијае на толкувањето на пораките на приемната страна;
- Да се предложат соодветни противмерки за новите скриени канали.

1.2 Главни придобивки

Оваа докторска дисертација е плод на истражувачката работа во последните пет години. Во почетокот на нашите истражувања при одредување на насоките утврдивме дека сè уште не постојат научни трудови кои се однесуваат на скриени канали кај протоколите кои се користат кај IoT и тоа: MQTT и CoAP. Ова беше основниот предизвик и мотивација за нас да дадеме придонес во научната сфера која се занимава со оваа област.

Главните придобивки од оваа докторска дисертација за CoAP-протоколот се следните:

- Предложени се вкупно 8 скриени канали, од кое 6 се складирачки, додека 2 се временски.
- Направена е експериментална евалуација на скриениот канал кој ги користи методите PUT и DELETE.
- Со помош на Cooja симулаторот разгледана е потрошувачката на енергија за Zolertia Z1 модул со имплементиран и без имплементиран скриен канал. Според добиените податоци имплементацијата на скриениот канал со PUT и DELETE-методите не влијае многу на потрошувачката на енергија.
- Предложени се соодветни противмерки за детекција, лимитирање или отстранување на новите скриени канали.

Главните придобивки од истражувањата за MQTT-верзија 3.1.1 протоколот се следните:

- Предложени се 13 мрежни скриени канали, од кои 6 се директни, додека 7 индиректни.
- Направена е експериментална евалуација на скриениот канал кој користи подредување теми и ажурирања со присуство/отсуство.
- Разгледани се три особини за предложениот скриен канал и тоа: пропусен опсег, неоткривање и робусност.

- Пропусниот опсег за овој скриен канал зависи од бројот на темите кои се користат за испраќање пораки. Во експерименталното сценарио користени се најмногу 4 теми, според што максималниот пропусен опсег изнесува 4 bps.

- Направено е истражување за детекција на скриениот канал во случај кога имаме легитимен сообраќај, испраќање ASCII пораки и испраќање пораки кои се шифрирани со AES. Според добиените резултати детекцијата значително се намалува во случајот кога имаме шифриран сообраќај со AES.

- За робусноста е откриено дека со зголемување на времето на доцнење на пакетите се зголемува и превртувањето на битовите. Резултатите покажуваат дека за 10 ms доцнење, 2,5 % од битовите се примени со доцнење, за 50ms, 3,9 % се примени со грешка, и за 100ms, 8,6% од битовите се примени со грешка.

- Предложени се соодветни противмерки за детекција, лимитирање или отстранување на новите скриени канали.

Главните придобивки од истражувањата за скриени канали кај протоколот MQTT-верзија 5.0 се следните:

- Предложени се 18 директни и 5 индиректни скриени канали.

- Направена е експериментална евалуација на скриениот канал кој користи различни теми.

- Испитување на пропусниот опсег, неоткривањето и робусноста за предложениот скриен канал.

- Во однос на пропусниот опсег, експериментите се извршени за три различни пропусни опсези, 2, 4 и 6 бита/секунда, што одговара на објавување во 1, 2 и 3 теми, соодветно.

- За неоткривањето ги имаме истите сценарија како и кај верзијата 3.1.1 на MQTT-протоколот: испраќање пораки преку легитимен комуникациски канал, испраќање ASCII пораки и испраќање пораки кои се шифрирани со

AES. Според добиените резултати имплементираниот AES-скриен канал не може да се детектира.

- За робусноста, експерименталните резултати покажуваат дека за 50 милисекунди доцнење, 0,84 % од сите битови се примени со грешки, за 100 милисекунди се примени 1,66 %, за 200 милисекунди се примени 2,5 %, за 300 милисекунди се примени 5,84 %, за 400 милисекунди се примени 12,50 % и за 500 милисекунди доцнење се примени 18,34 % битови со грешки. Според ова може да се забележи дека со зголемување на доцнењето, се зголемува и процентот на битови со грешки.
- Предложени се соодветни противмерки за детекција, лимитирање или отстранување на новите скриени канали.

1.3 Преглед на содржината

Докторската дисертација се состои од 7 глави и два прилога.

Првата глава е вовед во мрежното криење на информации. Во оваа глава исто се претставени целите на истражувањето, придобивките и преглед на содржината.

Во втората глава опфатени се видовите на мрежни скриени канали и претставени се трудовите кои се објавени досега во оваа сфера. Во овој дел исто така прикажани се стандардизираните шеми за мрежни скриени канали кои подоцна се користат за категоризација на предложените скриени канали.

Во третата глава претставен е концептот на интернет на нештата, протоколите кои се користат, стекот на протоколи кај IoT како и предложената 3-слојна архитектура на протоколи кај IoT.

Четвртата глава се однесува на скриените канали кај CoAP-протоколот. Овде се претставени основите на CoAP-протоколот, предложените скриени канали, прикажани се резултатите од експерименталната евалуација на еден од каналите и предложени се противмерки за детекција и отстранување на скриените канали.

Во петтата глава се прикажани скриените канали кај протоколот MQTT-верзија 3.1.1. Овде најпрвин се објаснети основите на MQTT. Потоа се

прикажани скриените канали кај овој протокол. Претставена е експериментална евалуација на еден од предложените скриени канали и обезбедени се соодветни противмерки.

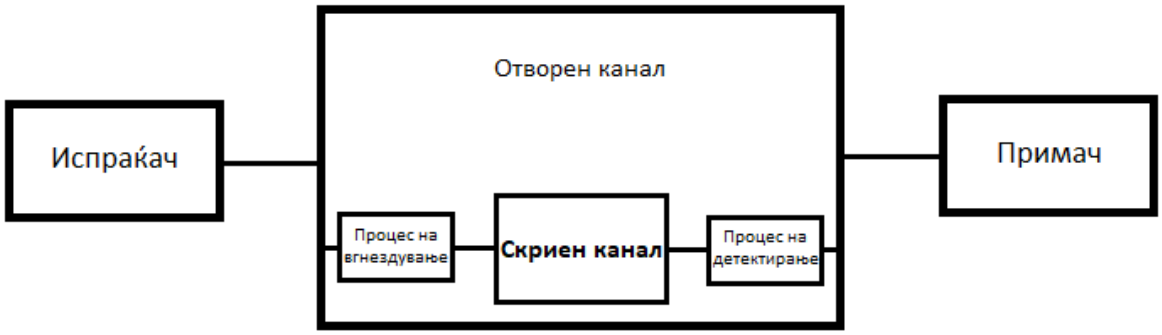
Шестата глава се однесува на скриени канали кај протоколот MQTT-верзија 5.0. Овде се претставени новите особини кои се додадени во оваа верзија на протоколот. Прикажана е применливоста на постоечките скриени канали од MQTT-верзија 3.1.1 на MQTT-верзија 5.0. Предложени се и нови скриени канали кои се креирани со користење на новододадените особини кај MQTT 5.0. Во оваа глава дадена е експериментална евалуација на еден од предложените скриени канали како и соодветни противмерки.

Седмата глава е заклучок од истражувањата и добиените резултати. Овде исто така се прикажани плановите за следни истражувања.

На крајот од докторската дисертација се прикажани два прилога. Во прилог 1 е прикажан кодот за испраќање легитимен сообраќај (без имплементиран скриен канал кај MQTT-верзија 3.1.1) и кодот за испраќање пораки во ASCII-формат. Во прилог 2 е прикажан кодот за објавувач без имплементиран скриен канал кај MQTT-верзија 5.0 и кодот за објавувач со имплементиран скриен канал за испраќање порака во ASCII-формат.

2. МРЕЖНИ СКРИЕНИ КАНАЛИ И НИВНИ ШЕМИ

На почетокот истражувачите се фокусирале на истражувања за локални скриени канали. Кај нив се креира комуникациска патека за пренесување скриени пораки помеѓу два процеси кои се извршуваат на ист уред на пр. мобилен уред (Urbanski et al., 2017). Со развитокот на компјутерските мрежи истражувањата биле насочени кон мрежни скриени канали. Тие кодираат скриени пораки во мрежните протоколи и уште се нарекуваат отворени канали (сл. 2.1).



Слика 2.1 Скриен канал за пренесување на пораки

Figure 2.1 Covert channel for transmitting messages

Мрежните скриени канали може да бидат складирачки, кои ги кодираат податоците во полињата на протоколите, и временски, кои ги сокриваат податоците со манипулирање на времињата на рамките, пакетите или пораките. Складирачките скриени канали се лесни за употреба, но повеќето од овие канали може лесно да се детектираат и да се елиминираат. Временските канали се потешки за користење, но исто така тие потешко може да се детектираат и елиминираат.

Повеќе скриени канали за комуникациските протоколи се проучувани во последните три декади (Mazurczyk et al., 2016; Zander et al., 2007; Mileva & Panajotov, 2014). Повеќето од каналите кои биле предложени биле складирачки. Girling предложил криење на информации во адресните полиња (Girling, 1987). Rowland предложил скриени канали во различни неискористени полиња во IPv4 заглавието и во TCP заглавието (Rowland, 1997). Handel и Sandford предложиле скриен канал со користење на неискористените битови на „Type of Service“ (TOS) полето во IP-заглавието или „Flags“ полето во TCP заглавието (Handel & Sanford,

1996). Hintz предлага пренесување скриени податоци во TCP Urgent Pointer (кое се користи за да означи податоци со висок приоритет) кое е неискористено ако URG-битот не е поставен (Hintz, 2002). Kundur и Ahsan предлагаат користење на „Don't Fragment" (DF) битот од IP-заглавието како скриен канал (Kundur & Ahsan, 2003).

Постојат и неколку студии кои прикажуваат индиректни мрежни скриени канали (Rowland, 1997; Danezis, 2008; Seclists.org, 2005; SecuriTeam, 2005). Една од студиите прикажува индиректен складирачки таен канал кој може да се користи во иста LAN-мрежа, за Address Resolution Protocol (ARP) и Simple Network Management Protocol (SNMP) (Schmidbauer et al., 2019). Основната идеја за овој таен канал е тоа што тајниот испраќач го искористува ARP-кешот на невина страница за складирање скриени пораки, додека тајниот примач го искористува SNMP за да ги прими пораките. Друг индиректен складирачки таен канал ги користи кешираните влезови на Pending Interest Table (PIT), одржувана од NDN (Named Data Networking) рутер, заедно со PIT-промашувања и PIT-погодоци (Ambrosin et al.).

Покрај складирачките биле предложени и временски скриени канали. Wolf предложил скриен канал којшто бил базиран на времетраењето на потврдите на пораките (Wolf, 1989). Во (Berk et al., 2005) предложен е временски канал кој ги користи празнините помеѓу последователните пакети (меѓупакетни празнини) за кодирање скриени битови. Cabuk предложил кодирање скриени податоци во ратите на пакетите кои вариpаат со тек на време (Cabuk, 2006).

Постојат неколку студии кои се објавени и кои обработуваат техники за криење податоци во некои протоколи кај интернет на нештата. За Building Automation and Control Networking Protocol (BACnet) протоколот се откриени два складирачки и еден временски скриен канал (Wendzel, 2012). За Extensible Messaging and Presence Protocol (XMPP) откриени се неколку складирачки скриени канали (Patuck & Hernandez-Castro, 2013). Во една од студиите прикажано е криење податоци во сајбер-физичките системи (како на пример паметните згради), со модификување на некоја од компонентите, како сензори, активатори, контролери и сл., како и со криење податоци во регистри кои не се користат (Wendzel, 2017).

Постојат многу малку истражувања кои се однесуваат на скриени канали кај протоколите кои се специјализирани за ограничени уреди кај интернет на нештата (Denney et al., 2016; Islam et al., 2017), како што се протоколите MQTT и CoAP. Тоа е причината поради која во оваа докторска дисертација прикажуваме една сеопфатна анализа на скриени канали кај овие протоколи.

Шемите за мрежни скриени канали се апстрактни описи за тоа како податоците може да бидат скриени при мрежна трансмисија (Wendzel et al., 2015). Во основа секоја шема претставува идеја за тоа како скриените податоци може да бидат претставени преку мрежен сообраќај. Сите овие шеми формираат таксономија. Целосен преглед на шемите може да се види на сл. 2.2. Притоа оригиналната таксономија била проширувана повеќе пати. Претпоследното проширување произлегува од ова истражување, според кое додадена е потшемата PS11.c - Value Influencing (Velinov et al., 2019; Network information hiding patterns, 2019). Последното проширување потекнува од друго истражување за скриени канали кај последната верзија на MQTT v.5.0 (Mileva et al., 2021), според кое е додадена потшемата PT.15 – Artificial Retransmissions. Како што може да се види на слика 3.1, шемите за криење се поделени во две главни категории и тоа: временски и складирачки. Кај временските шеми за криење податоци се врши модулирање на временското однесување на мрежниот сообраќај. Кај складирачките шеми се врши модулирање на складираните вредности во мрежниот сообраќај.

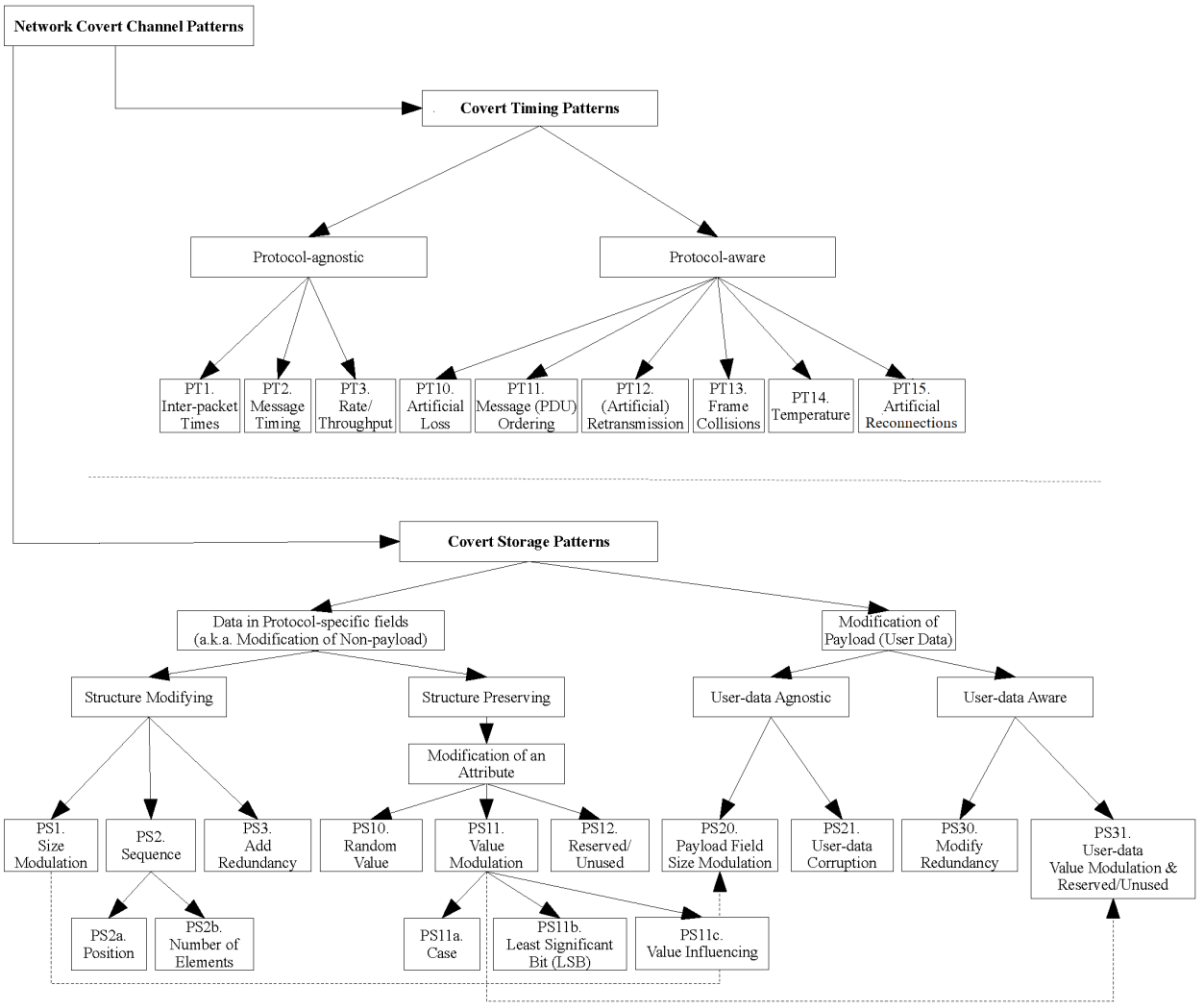
Пример за временски скриени канали може да биде модификување на времето помеѓу мрежните пакети со цел да се изврши кодирање на скриени податоци. Пример за складирачки скриени канали може да биде модификување на неискористените битови во заглавието на мрежните пакети.

Временските канали може да бидат „protocol-agnostic“ (нивното однесување не ја зема предвид интерпретацијата на податоците кај протоколот) или „protocol-aware“ (нивното однесување мора да ја земе предвид интерпретацијата на податоците кај протоколот). На пример, кога се врши модулирање на времињата помеѓу мрежните пакети (шема PT1. Inter-packet Times), не е важно кој тип на мрежен протокол се користи или каква е семантиката на сообраќајот. Доколку пак скриените податоци се сигнализираат

со повторни испраќања на селектирани рамки (шема PT12. Retransmission), тогаш структурата на овие рамки, на пример бројот на секвенца или полето за идентификатор мора да бидат утврдени за да се одреди кој пакет бил повторно пренесен.

Шемите за складирачките скриени канали од друга страна се поделени во две групи и тоа: шеми за канали кои вршат модификување на атрибутите на товарот и шеми за канали кои вршат модификување на атрибутите кои не се однесуваат на товарот, како што е заглавието на протоколот и „padding“ полињата. Шемите за криење коишто вршат модификување на полиња кои не се однесуваат на товарот ја менуваат структурата на пакетите кои се пренесуваат (на пр. со модулирање на големината на пакетите – шема PS1. Size Modulation) или тие ја зачувуваат структурата на пакетите кои се пренесуваат (на пр. со замена на некоја случајна вредност со некоја шифрирана тајна вредност – на пр. шема PS10. Random Value). Во однос на шемите за складирање кои вршат модификување на товарот, во таксономијата се вклучени само оние методи кои не вршат директна манипулација на содржината на товарот (на пр., слики и e-mail содржина), бидејќи сличните методи би биле дел од стеганографијата на дигитални медиуми. Постојат две категории за шеми за криење кои вршат модификување на товарот и тоа: шеми свесни за корисничките податоци (user-data aware) и шеми агностички за корисничките податоци (user-data agnostic). Шемите свесни за корисничките податоци бараат разбирање на податоците кои се пренесуваат (на пр., компресирање на VoIP-податоци со друг кодек, а не со оригиналниот, за да се создаде место за криење податоци во пакетот - шема PS30. Modify Redundancy). Шемите агностички за корисничките податоци вршат игнорирање на значењето на пренесената содржина (на пр., со зголемување на големината на товарот - шема PS20. Payload Field Size Modulation или со презапишување на оригиналната содржина - шема PS21. User-data Corruption).

Во табела 2.1 прикажан е преглед на сите досега познати шеми за мрежни скриени канали. Последната верзија од целата таксономија може секогаш да се види на следната веб-страница http://ih-patterns.blogspot.com/.



Слика 2.2 Класификација на шемите за мрежни скриени канали

Figure 2.2 Classification of covert channels patterns

Табела 2.1 Шеми за криење на информации

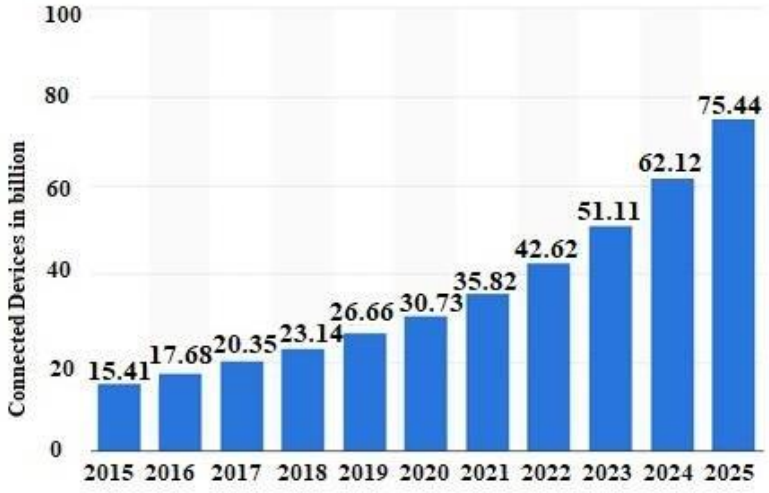
Table 2.1 Information hiding patterns

Име на шемата	Опис на шемата
Шеми за временски скриени канали	
Protocol-agnostic	
PT1. Inter-packet times	Овој скриен канал ги менува временските интервали помеѓу мрежните пораки во даден проток за да овозможи криење податоци.
PT2. Message timing	Скриените податоци се кодирани во времето на секвенците од пораки во даден проток, на пример земање предвид на секоја n-та примена порака или испраќање команди m пати.
PT3. Rate/throughput	Тајниот испраќач кај овој скриен канал ја менува податочната брзина на даден проток самостојно или од друго лице до примателот.
Protocol-aware	
PT10. Artificial loss	Овој таен канал сигнализира скриени информации преку вештачка загуба на пораки кои се пренесуваат во даден поток, на пример корупција на рамка или губење на порака.

PT11. Message Ordering	Овој скриен канал кодира податоци користејќи синтетички редослед на пораки во проток.
PT12. (Artificial) Retransmission	Со овој скриен канал се реемитуваат претходно испратени или примени пораки во даден проток.
PT13. Frame collisions	Испраќачот прави вештачки колизии на рамки за да сигнализира скриени информации.
PT14. Temperature	Испраќачот влијае на температурата на хардверот на даден јазол со користење сообраќај на проток. Мора да постои техника за тајниот примател да ја мери температурата (индиректно).
PT15. Artificial Reconnection	Овој таен канал користи вештачки (присилни) повторни конекции на клиентите (со соодветно ID) за пренесување тајни пораки.
Шеми за складирачки скриени канали	
Податоци во специфични полиња на протоколот (Модификација на не-товар)	
Модификување на структурата (Structure Modifying)	
PS1. Size Modulation	Овој скриен канал ја користи големината на метаподатоците на протокот (на пример, PDU големината или големината на елемент во заглавието) за кодирање скриени податоци.
PS2. Sequence modulation	Овој скриен канал ја менува секвенцата на метаподатоците во протокот за да кодира скриени информации. Оваа шема понатаму се дели на: P2.a Позиција и P2.b Број на елементи.
PS3. Add redundancy	Овој скриен канал вгнездува редундантни метаподатоци (на пример, со додавање на неискористена IP опција) во кои податоците се скриени во протокот. Забележете дека во споредба со PS1, податоците се скриени во редундантната присутност на метаподатоците, не во големината на PDU или елемент во заглавието.
Зачувување на структурата (Structure preserving)	
Модификација на атрибут (Modification of an Attribute)	
PS10. Random Value	Овој скриен канал вгнездува скриени податоци во метаподатоците на протокот коишто содржат псевдо – случајна вредност.
PS11. Value Modulation	Овој скриен канал селектира една од n вредности коишто елементот на метаподатокот во протокот може да ги содржи за да се изврши кодирање на скриена порака. Оваа шема понатаму се дели на: PS11.a Буква, PS11.b Најмалку значаен бит (LSB) и PS11.c Влијание на вредност (оваа потшема е додадена според ова истражување)
PS12. Reserved/Unused	Овој скриен канал кодира скриени податоци во резервираните и неискористените елементи на метаподатоците во протокот.
Модификација на товар (Кориснички податоци)	
Модификување на структурата (Structure Modifying)	
User-data Agnostic	
PS20. Payload field size modulation	Големината на товарот во протокот се користи за кодирање на скриени информации (ова е дериват на PS1, но за товарот бидејќи тој овозможува модификација на товарот на големината на полето товар на PDU).
PS21. User-data corruption	Овој скриен канал извршува (слепо) внесување на скриени податоци во товарот на протокот (слично на PT10).
User-data Aware	
PS30. Modify redundancy	Овој скриен канал го компресира товарот на протокот и резултирачкиот ослободен простор се користи за криење податоци.
PS31. User-data Value Modulation and Reserved/Unused	Овој скриен канал извршува модификација на товарот на протокот на начин на кој тоа не се рефлектира од PS30 и тоа не резултира во значителна модификувана интерпретација на податоците, на пр., со модификување на најмалку значајните битови кај дигиталните слики или криење податоци во неискористени/резервирани битови на товарот.

3. ИНТЕРНЕТ НА НЕШТАТА (IOT) И ПРОТОКОЛИ КАЈ IOT

Интернет на нештата претставува мрежа на уреди („објекти“ или „нешта“), кои имаат вградено сензори, софтвер или други слични технологии и кои имаат за цел да разменуваат податоци со други уреди или системи преку интернет. Бројот на уреди постојано се зголемува. Според одредени предвидувања бројот на уреди кај интернет на нештата до 2025 година ќе изнесува околу 75 милијарди уреди (сл. 3.1).



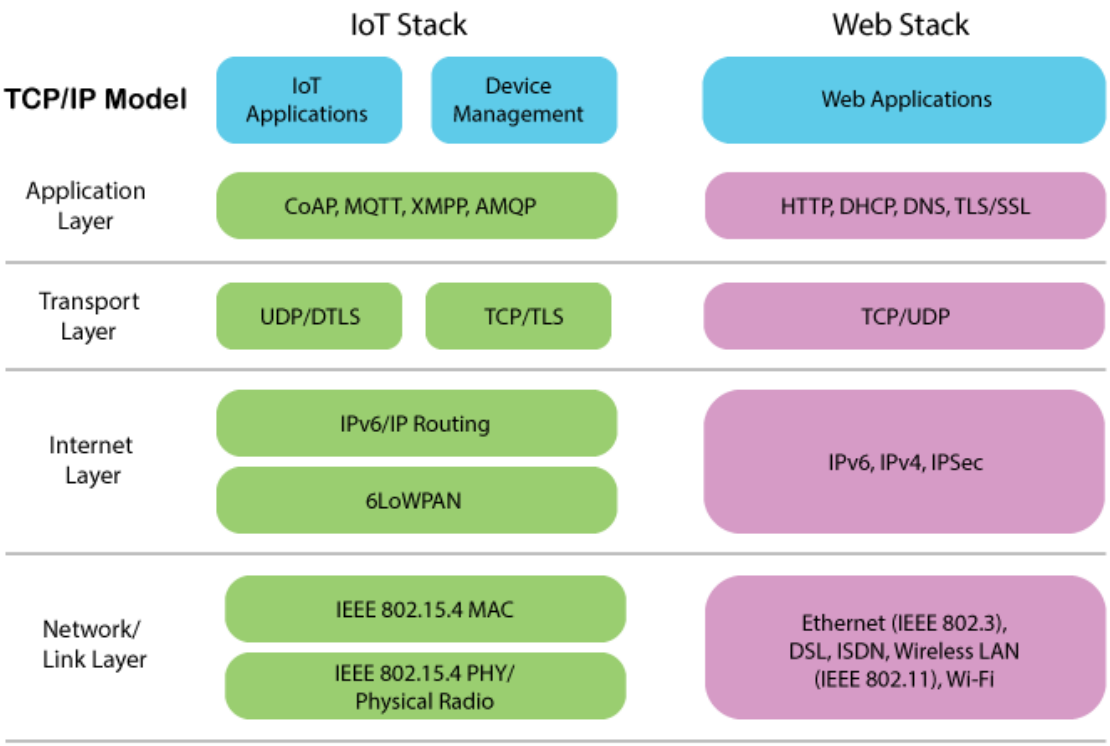
Слика 3.1 Број на уреди кај интернет на нештата (2015-2025) (Statista, 2016)

Figure 3.1 Number of Internet of Things devices (2015-2025) (Statista, 2016)

Во 2010 година се предвидуваше дека бројот на уреди до 2020 година ќе биде 50 милијарди. Ова за прв пат е споменато во 2010 година од страна на Hans Vestberg, поранешен извршен директор на Ericsson (Ericsson, 2010). Во 2011 година истото мислење го имал и Dave Evans, вработен во Cisco во тоа време (Cisco, 2011). Information Handling Services (IHS) во 2016 година предвидел дека бројот на IoT-уреди до 2020 година ќе биде 30,7 милијарди, додека до 2025 година овој број би се зголемил на 75,4 милијарди (IHS, 2016). Предвидувањата пак на Gartner биле дека во 2020 би имале 20 милијарди уреди (Gartner, 2017), додека International Data Corporation (IDC) предвидувал 28,1 милијарда уреди исклучувајќи ги паметните телефони, таблетите и компјутерите (IEEE Spectrum, 2016). Покрај овие предвидувања, бројот на уреди моментално изнесува околу 31 милијарда (Techjury, 2021). IDC предвидува дека до 2025 година ќе има 55.7 милијарди поврзани уреди, од кои 75 % ќе бидат поврзани на некоја IoT-

платформа. IDC исто така проценува дека до 2025 година би имале околу 73,1 ZB (зетабајти) податоци кои ќе бидат генерирани од поврзаните IoT-уреди, во споредба со 18,3 ZB во 2019 година (IDC, 2020). Најголемиот дел од овие податоци се претпоставува дека ќе потекнуваат од апликациите за безбедност и видео надзор, но се очекува исто така дека би имале голем број на генерирани податоци и од IoT-уредите кои се користат во индустријата (Industrial Internet of Things - IIoT).

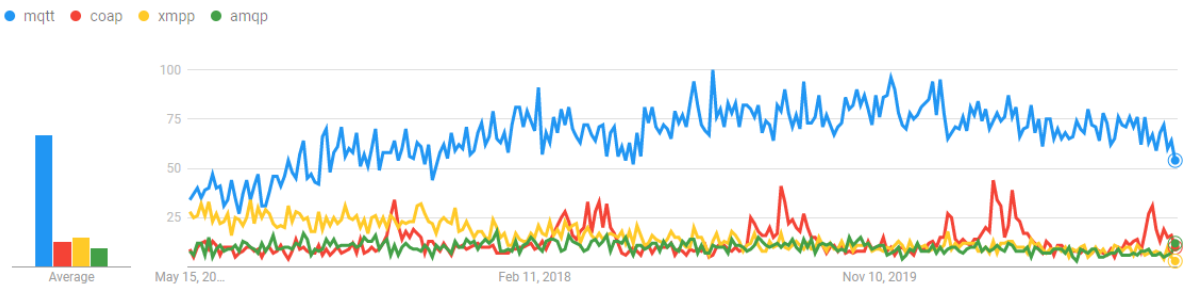
Кај IoT-комуникацијата посебно се важни протоколите кои се наоѓаат на апликациско ниво од стекот на протоколи кај IoT (сл. 3.2). На ова ниво најпознати протоколи се: CoAP, MQTT, XMPP и AMQP.



Слика 3.2 IoT-стек на протоколи

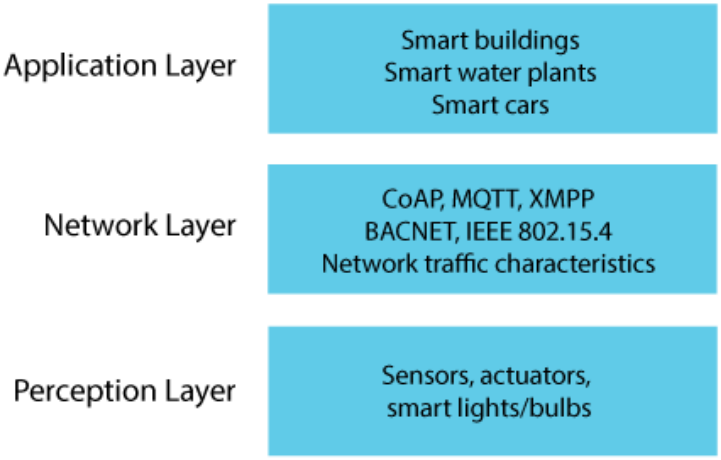
Figure 3.2 IoT protocol stack

Трендот во однос на пребарувањата за овие протоколи од 15.5.2016 до 15.5.2021 може да се погледне на слика 3.3. Како што може да се забележи најмногу се издвојува и најмногу пребаруван е MQTT-протоколот.



Слика 3.3 Интерес за протоколите на апликациско ниво кај IoT во однос на пребарувањето (Google Trends, 2021)

Figure 3.3 Interest for application layer protocols in IoT in terms of search (Google Trends, 2021)



Слика 3.4 IoT 3-слојна архитектура

Figure 3.4 IoT 3-layer architecture

Во однос на IoT предложена е 3-слојна архитектура која може да се види на слика 3.4 (Lin et al., 2017). Таа е составена од следните слоеви:

- Перцепциски слој: Тој уште е познат како сензорски слој. Истиот е имплементиран како најдолен слој во IoT-архитектурата. Овој слој е во интеракција со средината и со физичките уреди и компоненти преку паметни уреди (сензори, активатори и сл.). Неговите основни цели се да ги поврзе нештата во IoT-мрежа, и да мери, собира, процесира податоци кои се поврзани со тие нешта и да ги пренесе податоците на погорниот слој преку соодветни интерфејси.

- Мрежен слој: Тој уште е познат како слој за пренесување. Имплементиран е како среден слој во IoT-архитектурата. Овој слој се користи за да ги прими процесираните податоци кои се обезбедени од перцепциискиот слој и да ги одреди насоките за пренесување на овие податоци до IoT центар (hub), уреди или апликации преку интегрирани мрежи. Мрежниот слој е најважниот слој во IoT-архитектурата, бидејќи врши пренесување на податоците од различни уреди до апликацискиот слој. За ова се користат различни комуникациски технологии и протоколи како MQTT, CoAP, XMPP и сл.
- Апликациски слој: Овој слој уште е познат како бизнис слој и е имплементиран како најгорен во IoT-архитектурата. Тој ги прима податоците кои се испратени од мрежниот слој и ги користи нив за да обезбеди соодветни услуги или операции. На пример на овој слој може да се наоѓа сервис за креирање резервна копија со што податоците се складираат во бази. Дополнително овде може да се наоѓа соодветна апликација која овозможува анализа на податоците и извлекување соодветни заклучоци од нив. Ова е доста важно пред сè за индустриските процеси каде со анализа на податоците може соодветно да се менаџира и унапреди производството, да се донесуваат правилни одлуки и да се обезбеди раст на приходите и развој на компаниите. Денес сè повеќе компании вршат имплементација на IoT решенија и во иднина се очекува овој тренд на креирање паметни фабрики да расте (Frontiers in ICT, 2019). Според ова е креиран и концептот за индустриски интернет на нештата (IIoT).

Во периодот на нашите истражувања сè уште не постоеја студии кои ги истражуваат скриените канали на мрежниот слој од 3-слојната архитектура кај интернет на нештата освен за XMPP-протоколот. Затоа нашите истражувања ги насочивме кон откривање на скриени канали кај CoAP и MQTT (v.3.1.1 и v.5.0). Во однос на AMQP-протоколот, бидејќи истиот не се користи често кај IoT-комуникацијата, го оставивме за следни истражувања.

Големиот број на уреди коишто се користат кај интернет на нештата како и големата количина на генерирани податоци само ја потврдуваат важноста од

обезбедување соодветна безбедна IoT комуникација. Затоа во оваа докторска дисертација претставуваме посебен осврт на подобрувањето на безбедноста кај IoT со анализа на скриени канали кај протоколите CoAP и MQTT.

4. СКРИЕНИ КАНАЛИ КАЈ COAP-ПРОТОКОЛОТ

4.1 Основи на CoAP

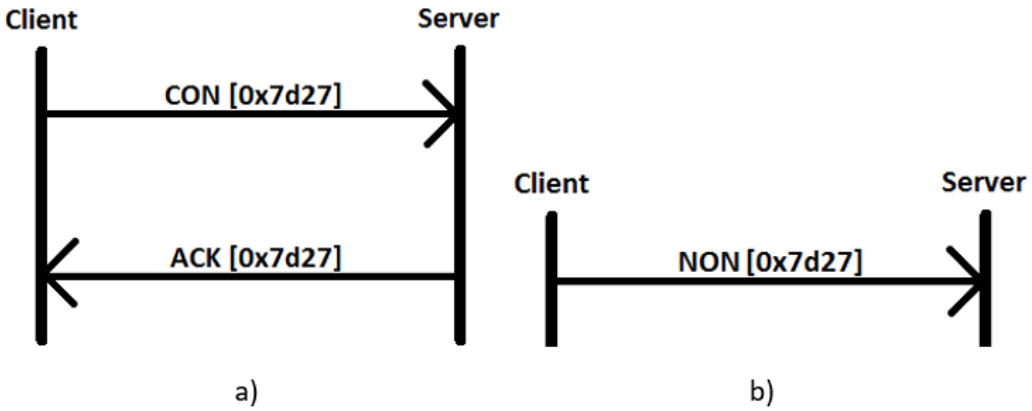
CoAP (Constrained Application Protocol) е специјализиран протокол за пренос на веб-содржини, кој се наоѓа на апликациско ниво кај IoT (Shelby et al., 2014). Тој може да се користи за ограничени уреди и ограничени мрежи кај IoT. Уредите се ограничени бидејќи имаат ограничена процесорска моќ (8 битни микро-контролери), ограничена RAM и ROM-меморија. Ограничените мрежи често имаат големи рати на грешки кај пакетите и мали податочни брзини (на пример, IPv6 over Low-Power Wireless Personal Area Networks – 6LoWPANs). CoAP е дизајниран за машина-до-машина комуникација (M2M) и неговата последна верзија била објавена во јуни 2014 во RFC 7252 (Shelby et al., 2014). Во основа тој е RESTful (Representational State Transfer) протокол со поддршка за мултикаст и набљудување.

Слично како HTTP протоколот, CoAP користи клиент/сервер модел со испраќање пораки како барање-одговор. Тој исто така поддржува вградено откривање на сервиси и ресурси, униформни идентификатори на ресурси (URIs) и интернет медиумски типови. CoAP испраќа порака за барање за извршување на некоја акција (со користење код за метод – Method Code) до ресурс којшто е идентификуван со URI и е хостиран на серверот. Серверот одговара на ова барање со користење порака за одговор, којашто содржи код за одговор (Response Code) и можеби некоја претстава на ресурс. CoAP дефинира четири типови на пораки и тоа:

- Потврдувачки (Confirmable - CON)
- Непотврдувачки (Non-Confirmable - NON)
- Верификувачки (Acknowledgement - ACK)
- Ресетирачки (Reset - RST).

Овие типови на пораки користат кодови за метод и кодови за одговор за да пренесуваат барања или одговори. Барањата може да се пренесат како CON или NON-типови на пораки, додека одговорите може да се пренесат преку CON, NON, ACK и „Piggybacked“ (пораки кај кои одговорот се враќа веднаш при испратено барање) типови на пораки.

CoAP користи чист UDP или DTLS на транспортно ниво за асинхрона размена на пораки помеѓу крајните комуникациски точки. Секоја порака содржи идентификатор на порака (Message ID) кој се користи за оптимална доверливост и детектирање дупликати (сл. 4.1).



Слика 4.1 a) Надежно пренесување на CoAP-порака b) Ненадежно пренесување на CoAP-порака

Figure 4.1 Reliable transmission of CoAP message b) Unreliable transmission of CoAP message

Пораката која бара надежно пренесување е означена како CON, додека онаа која не бара е означена со NON. CON-пораката се пренесува повторно, со користење на даден временски интервал и бинарен, експоненцијален алгоритам за зголемување на временскиот интервал помеѓу повторните пренесувања, сè додека примателот не испрати ACK-порака со ист идентификатор на порака (Message ID). Кога примателот воопшто не е во можност да процесира CON или NON-пораки, тој одговара со RST-порака.

CoAP-пораките се кодирани во едноставен бинарен формат. Структурата на пораката може да се види на слика 4.2.

VER	T	TKL	Code	Message ID
Token (if any, TKL bytes)				
Options (if any)				
11111111	Payload (if any)			

Слика 4.2 Формат на CoAP-порака

Figure 4.2 CoAP message format

Секоја порака започнува со 4-бајтно фиксно заглавие, проследено со полето „Token“, со големина од 0 до 8 бајти. Потоа доаѓа полето „Options“ кое е опционално и на крај полето „Payload“ кое исто така е опционално. Доколку полето „Payload“ е присутно во тој случај тоа е претходено со едно-бајтен „Payload Marker“ (0xFF).

Полињата кои се наоѓаат во заглавието на CoAP-пораката се следните:

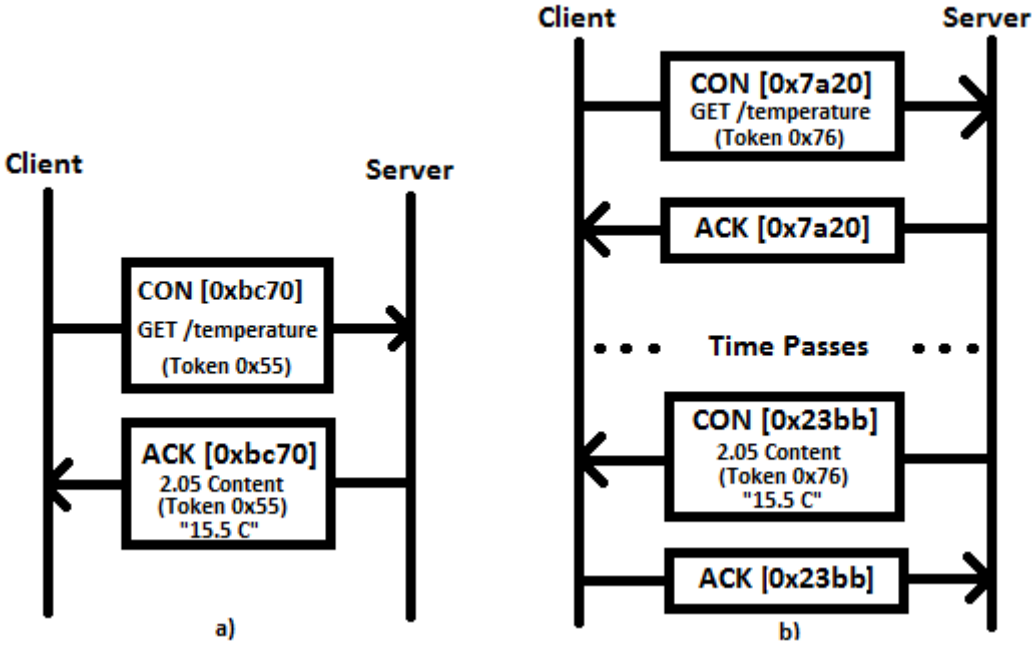
- Version (Ver) – 2-битен неозначен цел број (unsigned integer) кој ја идентификува верзијата на CoAP. Моментално тој мора да биде поставен на 01.
- Type (t) – 2-битен неозначен цел број кој го претставува типот на пораката: Confirmable (0), Non-Confirmable (1), Acknowledgement (2) и Reset (3).
- Token Length (TKL) – 4-битен неозначен цел број кој се однесува на големината на полето Token (0 - 64 бита). Големините 9 - 15 се резервирани и мора да бидат процесирани како грешка во форматот на пораката.
- Code – 8-битен неозначен цел број. Тој е поделен на два дела: 3-битна класа (најзначајните битови) и 5-битни детали (најмалку значајните битови). Форматот на кодот е следниот „c.dd“, каде „c“ е цифра од 0 до 7 и ја претставува класата, додека „dd“ се две цифри од 00 до 31. Според класата ние може да го одредиме типот на пораката, на пример: барање (0), успешен одговор (2), одговор со грешка кај клиентот (4), одговор со грешка кај серверот (5). CoAP има посебен коден регистар кој обезбедува опис за сите кодови (IANA, 2012).
- Message ID – 16-битен неозначен цел број кој се користи за детектирање пораки кои се дупликати и за поврзување на „Acknowledgement/Reset“ пораки со „Confirmable/Non-Confirmable“ пораки.

По заглавието на пораката следува полето „Token“ со променлива големина од 0 до 64 бита. Ова поле се користи за поврзување на барањата и одговорите. По полето „Token“ следува полето „Options“ кое е опционално и со него се дефинираат една или повеќе опции. CoAP дефинира единствен сет на опции кои се користат за барања и одговори. Тие се: Content-Format, Etag,

Location-Query, Location-Path, Proxy-Uri, Proxy-Scheme, Uri-Host, Uri-Port, Uri-Path, Uri-Query, If-Match, If-None-Match, Accept и Size1.

Товарот (Payload) на одговорите/барањата во основа ја содржи претставата на ресурсот или резултатот од бараната акција.

Постојат два типа на одговори и тоа: „piggybacked“ и посебен (separate) одговор (сл. 4.3). Ако барањето е пренесено преку CON или NON порака, и ако одговорот е достапен и пренесен преку ACK-порака, тогаш тоа е „piggybacked“ одговор. Доколку серверот не е во состојба да одговори веднаш на барањето, се испраќа празна порака (со код 0.00) која му кажува на клиентот да прекине со испраќање на барање. Ако серверот може подоцна да му одговори на клиентот, тој испраќа CON-порака која мора да биде потврдена од клиентот. Ова е наречено посебен (separate) одговор. Слично на HTTP, CoAP ги користи следните методи: GET (со код 0.01), POST (со код 0.02), PUT (со код 0.03) и DELETE (со код 0.04).



Слика 4.3 а) „Piggybacked“ одговор b) Посебен одговор

Figure 4.3 a) Piggybacked response b) Separate response

4.2 Скриени канали кај CoAP

Со користење на карактеристиките на CoAP-протоколот може да се креираат скриени канали. Кај CoAP постојат одредени механизми со кои се

овозможува заштита против мрежна стеганографија. Еден од овие механизми е воведувањето на соодветен редослед на различни особини во пораката. На овој начин оние стеганографски техники кои користат различен редослед на особини, не може да се применат.

Како што беше споменато претходно, CoAP најчесто се користи кај уреди кои имаат ограничени перформанси. Овој протокол може да се примени во повеќе области како: паметни мрежи, контрола на градба, интелигентна контрола на осветлување, индустриски контролни системи, следење средства, надгледување во средината итн. Скриените канали кај CoAP може да се користат за поддршка на автентикација на геолокација на уреди кај IoT, за тајна комуникација помеѓу преносливи уреди во непријателска средина, за потребите на војниците, или, помеѓу јазли во безжични сензорски мрежи.

За Constrained Application Protocol (CoAP) откриени се шест складирачки и два временски скриени канали (Mileva et al., 2018). Дополнително, претставена е експериментална евалуација на еден од скриените канали кај CoAP (Velinov et al., 2019).

4.2.1 Скриен канал со користење на полињата Token и/или Message ID

Полето „Message ID“ содржи случајна 16-битна вредност. Во случај на „piggybacked“ одговор за CON-порака, ова поле треба да биде исто како и во барањето. Во случај на посебен одговор, серверот генерира случаен и различен „Message ID“. При праќање на празна ACK-порака, вредноста на „Message ID“ се копира од барањето. Во комуникацијата помеѓу две исти крајни комуникациски точки, истата вредност на „Message ID“ во EXCHANGE\LIFETIME, кое е околу 247 секунди, не може да се користи повторно.

Полето „Token“ е друго случајно генерирано поле, со променлива големина до 64 бита. Тоа се користи како локален клиентски идентификатор за да се направи разлика помеѓу истовремени барања. Ако барањето резултира со одговор, вредноста за полето Token треба да се повтори и во одговорот. Ова исто така се случува и кога серверот испраќа одделен одговор.

Со овој скриен канал може да се креира еднонасочна или двонасочна комуникација помеѓу два хоста, со испраќање на 16 (од Message ID) и/или 64 (од Token) бита по порака (PRBR $\in$ {16, 64, 80})), каде PRBR е Packet Raw Bit Rate).

Шема за криење на информации:

PS10. Random Value Pattern

4.2.2 Скриен канал со користење на „Piggybacked“ и посебен одговор

Серверот има опција за испраќање на „piggybacked“ или посебен одговор, со што може да се креира 1-битен (по порака) еднонасочен или двонасочен скриен канал (PRBR=1), на следниот начин:

- „Piggybacked“ одговор да биде бинарно 1
- Посебен одговор да биде бинарно 0

При големо оптоварување, серверот можеби нема да може да одговори, па така овој скриен канал е ограничен на времето кога серверот има избор.

Шема за криење на информации:

PS11. Value Modulation

4.2.3 Скриен канал со користење на товарот (Payload) на пораката

Барањата и одговорите може да вклучат товар, зависно од методот (Method) или од кодот за одговор (Response Code). Неговиот формат е специфициран во интернет медиумскиот тип (Internet Media Type) и кодирањето на содржината обезбедено од опцијата “Content-Format”. Во основа товарот на барањето или одговорот е типична претстава на ресурсот или резултат од побараната акција. Доколку не е дадена опцијата Content-Format, товарот на одговорите којшто претставува клиентска или серверска грешка е дијагностички товар, со кратка дијагностичка порака кодирана со користење на UTF-8 во Net-Unicode формат.

CoAP-спецификацијата обезбедува само горна граница на големина на пораката за да биде прифатена во еден IP-датаграм (и во еден UDP товар). Максималната големина на еден IPv4-датаграм е 65.535 бајти, но ова не може да се примени кај ограничени уреди и мрежи. Според IPv4-спецификацијата во

RFC 791, сите хостови треба да бидат подготвени да прифатат датаграми до 576 бајти, додека IPv6 бара максималната единица за пренесување да биде најмалку 1280 бајти. Апсолутната минимална вредност на IP MTU за IPv4 е 68 бајти, кој треба да остави најмногу 35 бајти за CoAP-товар (најмалата големина на CoAP-заглавие со Payload Marker пред товарот е 5 бајти, по претпоставка 0 бајти за токенот и без опции). Кај ограничените мрежи постои и друго ограничување. На пример, за стандардот IEEE 802.15.4, стандардната големина на пакет е 127 бајти (со 25 бајти максимално надминување на рамка), што остава 102 бајти за погорните слоеви. Големините на влезните и излезните бафери кај ограничените уреди е друго ограничување на максималниот товар.

Со користење на товарот на пораката може да се креира еднонасочен или двонасочен скриен канал помеѓу два хоста, со праќање на дијагностички товар со максимална големина од 35 бајти по порака (PRBR=280).

Шема за криење на информации:

PS31. User-data value modulation & reserved/unused

4.2.4 Скриен канал со користење на „case-insensitive“ делови од униформните идентификатори на ресурси (URIs)

CoAP користи „coap“ или „coaps“ URI (Uniform Resource Identifier) шеми за идентификација на CoAP-ресурси и обезбедување на средства за лоцирање на ресурси. Овие идентификатори во барањето се пренесуваат преку неколку опции: URI-host, URI-Port, URI-Path и URI-Query. Тие се користат за да го специфицираат целниот ресурс на барањето до CoAP-серверот. URI-host и шемата не се чувствителни на букви, додека другите компоненти се чувствителни. Според ова може да се креира еднонасочен скриен канал помеѓу клиентот и серверот на следниот начин:

- Голема буква во опцијата URI-host да означува бинарно 1
- Мала буква во опцијата URI-host да означува бинарно 0

Имајќи предвид дека валидно DNS (Domain Name System) име има најмногу 255 бајти, тогаш PRBR на овој канал е до 255 бајти.

Шема за криење на информации:

PS11a. Case

CoAP-протоколот поддржува и прокси (посредник), каде што прокси е CoAP-крајна точка која може да биде зададена од CoAP-клиентите да извршува барања во нивно име. Посредниците може експлицитно да бидат селектирани од клиентите, со користење на опцијата Proxy-URI и оваа улога е „forward-proxy“. Тие може да бидат вметнати и на страната на серверот и во оваа улога се познати како „reverse-proxy“. Според ова може да се креира сличен скриен канал со користење на деловите шема и хост од опцијата Proxy-URI. Барањето кое ја содржи опцијата Proxy-URI не треба да ги вклучува опциите URI-host, URI-Path, URI-Port и URI-Query.

4.2.5 Скриен канал со користење на методите PUT и DELETE

Со методот PUT се овозможува ресурсот кој е идентификуван од URI во барањето, да биде ажуриран или креиран со соодветна претстава. Ако ресурсот постои на бараното URI, тогаш соодветната претстава треба да се разгледа како ажурирана верзија на ресурсот, и треба да се врати 2.04 (Changed) код за одговор. Ако ресурсот не постои, тогаш серверот креира нов ресурс со истото URI кое резултира со 2.01 (Created) код за одговор.

Со методот DELETE се овозможува бришење на ресурс, којшто е идентификуван со URI во барањето. Без оглед на тоа дали бришењето е успешно, или ресурсот не постои пред да се испрати барањето, ќе се врати 2.02 (Deleted) код за одговор.

Ако некој има позната претстава на постоечки ресурс R1 на серверот, и ако тој знае дека ресурсот R2 не постои на истиот сервер, може да креира еднонасочен скриен канал на следниот начин:

- Праќање на барање со метод PUT за ажурирање на ресурс R1 со соодветно позната претстава, за означување на бит 1.
- Праќање на барање со DELETE-метод за бришење на непостоечки ресурс R2, за означување на бинарно 0.

На овој начин, може да се испрати еден бит по порака (PRBR=1).

Шема за криење на информации:

PS11. Value Modulation

4.2.6 Скриен канал со користење на опцијата „Accept“

Опцијата Accept се користи за да покаже кој формат на содржина (Content Format) е прифатлив за клиентот. Ако не е дадена опција за прифаќање, клиентот не изразува претпочитување во однос на форматот. Ако предвидениот формат на содржина е достапен, серверот враќа одговор во тој формат, инаку, како одговор се враќа 4.06 (Not Acceptable) код за одговор, освен ако претходно нема код за грешка. Еднонасочен едно-битен скриен канал по порака (PRBR=1) може да се креира на следниот начин:

- Праќање на дадена порака без опција Accept да биде бинарно 1.
- Праќање на дадена порака со опција Accept да биде бинарно 0.

Шема за криење на информации:

PS11. Value Modulation

4.2.7 Скриен канал со користење на условни барања

Опциите „If-Match“ и „If-None-Match“ кај условните барања, му овозможуваат на серверот да го обработи барањето само ако одредени услови специфицирани со опции се исполнети. Во случај на повеќе „If-Match“ опции, клиентот може да направи условно барање на моменталната претстава или вредноста на ETag за една или за повеќе претстави на целниот ресурс. Условот не е исполнет ако ниту една од опциите не соодветствува. Со опцијата If-None-Match клиентот може да направи условно барање за моментално непостоење на даден ресурс. Ако целниот ресурс не постои, тогаш условот не е исполнет.

Ако некој знае дека даден услов C1 е исполнет (на пример: доколку ресурсот е креиран или избришан во претходна порака) а друг услов C2 не е исполнет, со користење на опциите „If-Match“ и „If-None-Match“ може да се креира еднонасочен едно-битен скриен канал по порака (PRBR) на следниот начин:

- Праќање на дадена порака без исполнет услов да биде бинарно 1 (If-Match + C2).
- Праќање на дадена порака со исполнет услов да биде бинарно 0 (If-Match + C1).

Шема за криење на информации:

PS11. Value Modulation

4.2.8 Скриен канал со користење на повторно испраќање

Доколку се користи CoAP со мала рата на грешки (со справување на недоверливата природа на UDP), може да се креира еднонасочен или двонасочен скриен канал со користење на повторни испраќања со PRBR=1, на следниот начин:

- Испраќање на дадена порака само еднаш да биде бинарно 1.
- Испраќање на дадена порака два или повеќе пати да биде бинарно 0.

На овој начин може да се испрати еден бит по порака.

Шема за криење на информации:

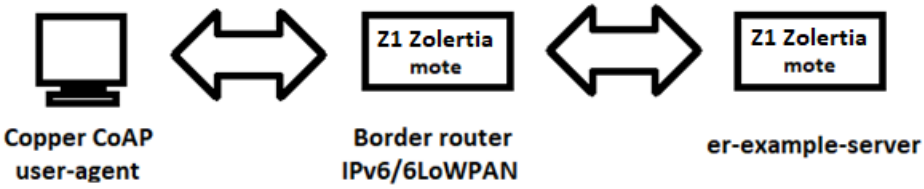
PT12. (Artificial) Retransmission

4.3 Експериментална евалуација на скриениот канал кој ги користи методите PUT и DELETE

За експериментална евалуација на скриениот канал кој ги користи методите PUT и DELETE користен беше оперативниот систем Contiki, конкретно Instant Contiki верзија 3.0, како развојна околина. Тоа е Ubuntu Linux виртуелна машина која се извршува во програмата VMWare. Овој оперативен систем ги има сите алатки, компајлери и симулатори кои може да се користат за развивање на апликации за интернет на нештата. Притоа, апликацијата може да се развие и да се тестира на некој од уредите коишто постојат во симулаторот. За нашите цели користен беше симулаторот Cooja. Со него може да се креираат различни типови на уреди за кои може да развиваме апликации. Ова е доста практично затоа што пред да ја извршиме нашата апликација на реален уред, во симулаторот може да потврдиме дека таа работи правилно и според зацртаните цели.

За целите на истражувањето користен беше уредот Z1 Zolertia. Тоа е безжичен модул со мала моќ којшто се користи кај безжичните сензорски мрежи (WSN). Z1 ја има втората генерација на MSP430F2617 микро-контролер, којшто има 16-битен RISC CPU @16MHz брзина на часовник. Тој исто така има вградена

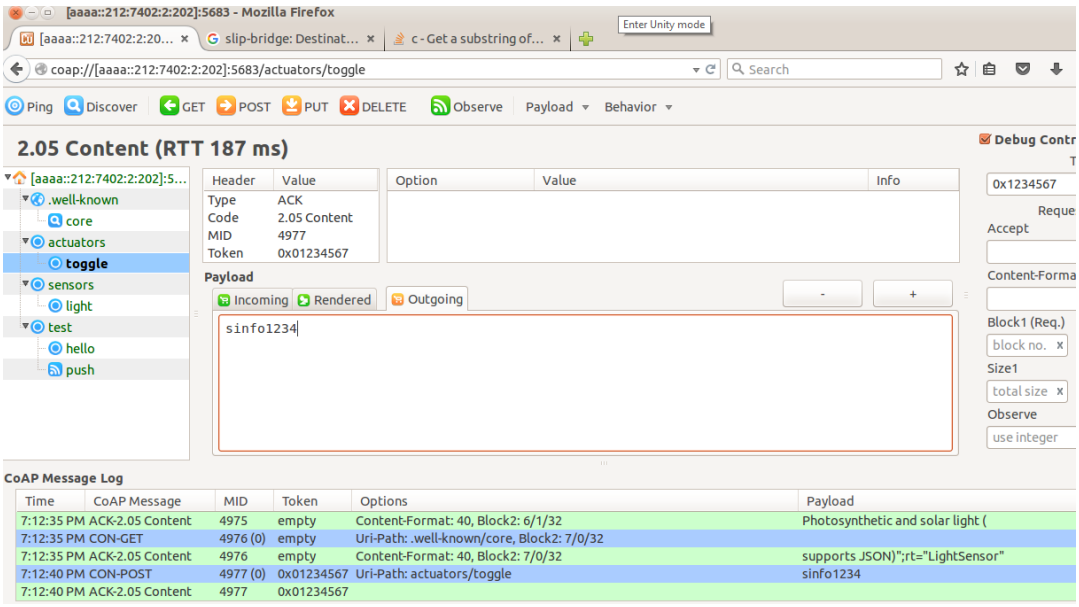
фабричка калибрација на часовник, 8KB RAM и 92KB флеш меморија. Z1 модулот вклучува CC2420 примопредавател, којшто оперира на 2.4GHz и податочна рата од 250 Kbps и го поддржува стандардот 802.15.4 за да остварува комуникација со други уреди. Овој модул има вграден температурен сензор и 3-оскен акцелерометар. Z1 овозможува флексибилно напојување со користење на батерии (2xAA или 2xAAA), батерии во вид на монети, USB или директна конекција.



Слика 4.4 Експериментално сценарио

Figure 4.4 Experimental scenario

Експерименталното сценарио е претставено на слика 4.4. За изведување на експериментите користен беше Copper (Cu) како CoAP-кориснички-агент (сл. 4.5). Тој е „Firefox“ додаток којшто инсталира управувач за „coap“ URI шема и им овозможува на корисниците да извршуваат пребарување и да се во интеракција со уредите кај интернет на нештата.



Слика 4.5 Copper (Cu) CoAP-кориснички-агент

Figure 4.5 Copper (Cu) CoAP user-agent

За истражувањата користен беше Cooja симулаторот за креирање на симулација со два Z1 Zolertia модули. Еден Z1 модул е за „Border Router“ (сл. 4.6, модул со ID 1). Како код за него користен беше изворниот код кој се наоѓа на следната локација во Instant Contiki:

/home/user/contiki-3.0/examples/ipv6/rpl-border-router

Другиот Z1 модул е за CoAP-серверот (сл. 4.6, модул со ID 2). За изведување на експериментите користена беше Erbium имплементацијата на CoAP-сервер за оперативниот систем Contiki. Изворниот код за него се наоѓа на следната локација во Instant Contiki:

/home/user/contiki-3.0/examples/er-rest-example

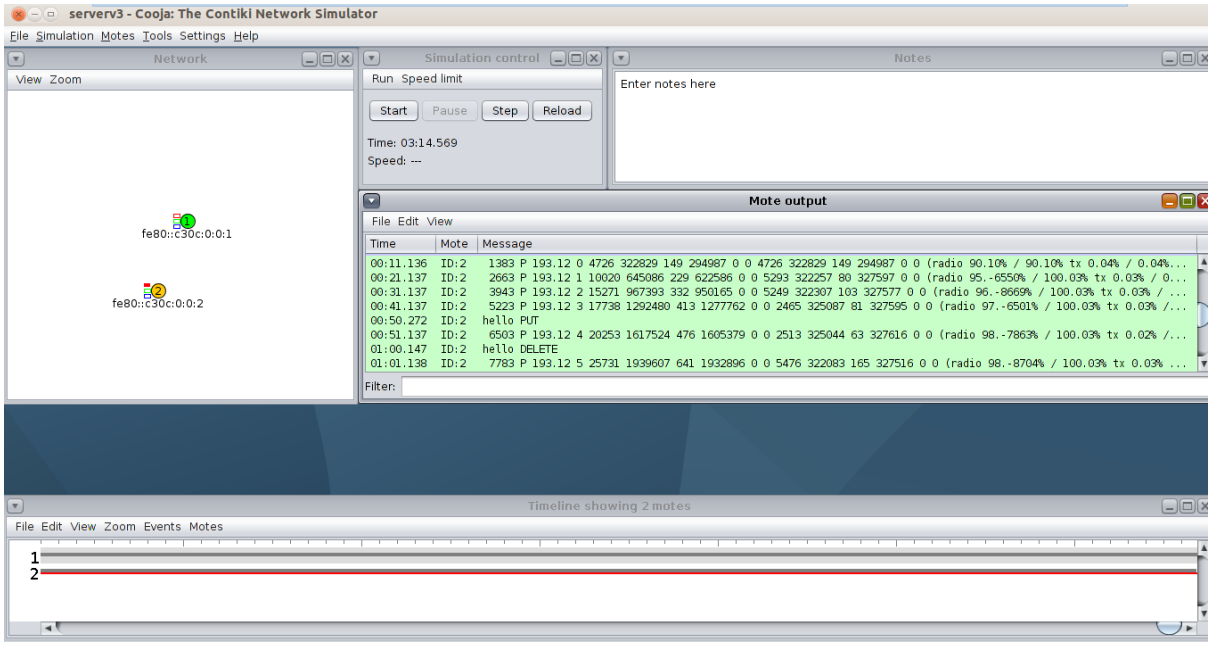
За приспособување на експериментите, направени се промени во изворниот код, посебно во документот „res-hello.c“ којшто се наоѓа на следната локација:

/home/user/contiki-3.0/examples/er-rest-example/resources

Даденото експриментално сценарио (сл. 4.4) беше користено за имплементација на скриениот канал кој ги користи методите PUT и DELETE. Со користење на Copper корисничкиот-агент, креирани беа барања со PUT и DELETE-методите (со PUT во 50-тата секунда од извршување на симулацијата и со DELETE во 60-тата секунда од извршување на симулацијата). На слика 4.5 може да се види излезот во Cooja симулаторот каде за модулот со ID 2, кој во основа ни е CoAP-сервер, во 50-тата секунда се испраќа барање со методот PUT, додека во 60-тата секунда се испраќа барање со методот DELETE. На ваков начин според предложениот скриен канал, испратеното барање со методот PUT ќе се толкува како испратен бит 1, додека испратеното барање со методот DELETE ќе се толкува како испратен бит 0.

Дополнително, направена беше анализа за потрошувачката на енергија во случај кога немаме имплементирано скриен канал и кога имаме имплементирано скриен канал. За пресметка на потрошувачката на енергија, користени беа податоците добиени со алатката „Powertrace“ за CoAP-сервер со

и без имплементиран канал. Овие податоци се печатат во излезот на Z1 модулот (симулираниот уред) во симулаторот Cooja.



Слика 4.6 Излез во Cooja симулаторот на испратени пораки преку скриениот канал кој ги користи методите PUT и DELETE

Figure 4.6 Output in Cooja simulator of sent messages via covert channel using PUT and DELETE methods

Пресметките за потрошувачката на енергија беа направени во вкупен временски интервал од 100 секунди како претходно предефиниран интервал за извршување на симулациите за двата случаја. Овие податоци го покажуваат вкупниот број на отчукувања на часовникот за различни состојби на модулот: CPU (CPU во активен мод на работа), LPM (CPU во „Low-Power“ мод на работа), TX (Transmit) и RX (Receive).

Табела 4.1 Податоци добиени со „Powertrace“ за CoAP-сервер без имплементиран скриен канал

Table 4.1 Data obtained with “Powertrace” for CoAP Server without implemented covert channel

ALL_CPU	ALL_LPM	ALL_TX	ALL_RX
4674	322863	149	294987
9879	645197	229	622586
15204	967576	412	950244
17500	1292676	412	1277763
19778	1617956	412	1605442
24933	1940346	514	1933022
27435	2265399	594	2260619
29721	2590672	594	2588298
32001	2915952	594	2915978
34271	3241241	594	3243658
39491	3563562	675	3571258

Во табела 4.1 се прикажани податоците кои се добиени со „Powertrace“ за CoAP-сервер без притоа да имаме имплементиран скриен канал. Во табела 4.2 се прикажани податоците кои се добиени со „Powertrace“ за CoAP-сервер во случај кога имаме имплементиран скриен канал со користење на методите PUT и DELETE.

За пресметка на потрошувачката на енергија користена беше следната формула:

$$Power\_consumption = \frac{Energest\_value * Current * Voltage}{RTIMER\_SECOND * Runtime}$$

Во формулата, „*Energest_value*“ е разликата помеѓу бројот на отчукувања на часовникот (во состојби CPU, LPM, TX и RX) помеѓу два временски интервали.

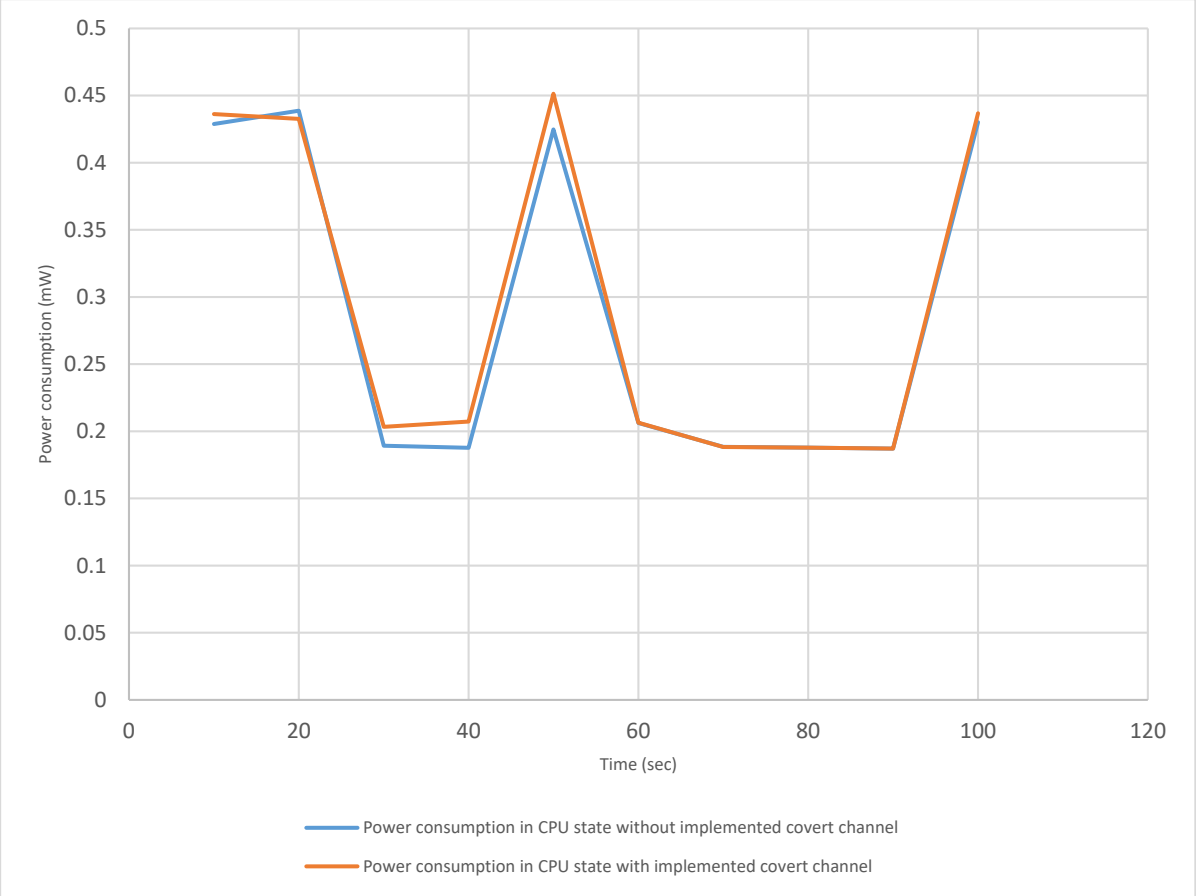
Табела 4.2 Податоци добиени со „Powertrace“ за CoAP-сервер со имплементиран скриен канал

Table 4.2 Data obtained with „Powertrace“ for CoAP server with implemented covert channel

ALL_CPU	ALL_LPM	ALL_TX	ALL_RX
4726	322829	149	294987
10020	645086	229	622586
15271	967393	332	950165
17738	1292480	413	1277762
20253	1617524	476	1605379
25731	1939607	641	1932896
28236	2264657	722	2260493
30521	2589930	722	2588173
32801	2915210	722	2915853
35071	3240499	722	3243533
40372	3562751	802	3571132

За утврдување на вредностите за „*Current*“ во различни состојби (Approximate Current Consumption of Z1 circuits: Active Mode @16MHz - < 10mA (approximate 9mA), Standby Mode - 0.5µA, RX Mode - 18.8mA, TX Mode - 17.4mA), користена беше официјалната документација (Zolertia, 2010). Вредноста за параметарот „*Voltage*“ е 3V. Вредноста за „*RTIMER_SECOND*“ е 32768. „*Runtime*“ е временскиот интервал (10 секунди во овој случај).

Слика 4.7 ја прикажува потрошувачката на енергија за Z1 модул (имплементиран како CoAP-сервер) во состојба CPU со и без имплементиран скриен канал.

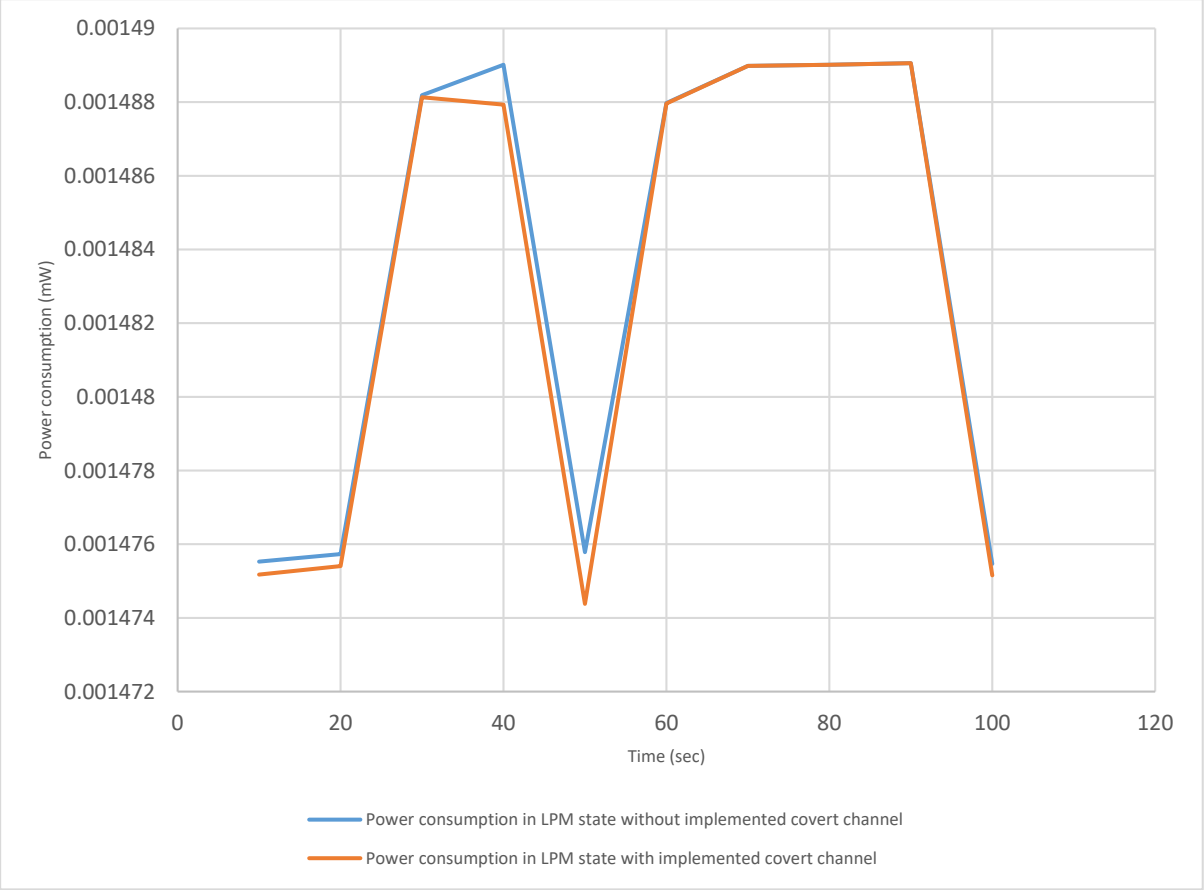


Слика 4.7 Потрошувачка на енергија за CoAP-сервер (Z1) во состојба CPU (со и без имплементиран скриен канал)

Figure 4.7 Power consumption for CoAP server (Z1) in CPU state (with and without implemented covert channel)

Просечната потрошувачка на енергија без имплементиран скриен канал е 0.28688324 mW, додека просечната потрошувачка на енергија со имплементиран скриен канал е 0.293713989 mW. Според ова може да се забележи дека просечната потрошувачка на енергија со имплементиран скриен канал е малку поголема.

Слика 4.8 ја прикажува потрошувачката на енергија за Z1 модул (имплементиран како CoAP-сервер) во состојба LPM со и без имплементиран скриен канал. Просечната потрошувачка на енергија без имплементиран скриен канал е 0.001483474 mW, додека просечната потрошувачка на енергија со имплементиран скриен канал е 0.001483119 mW.

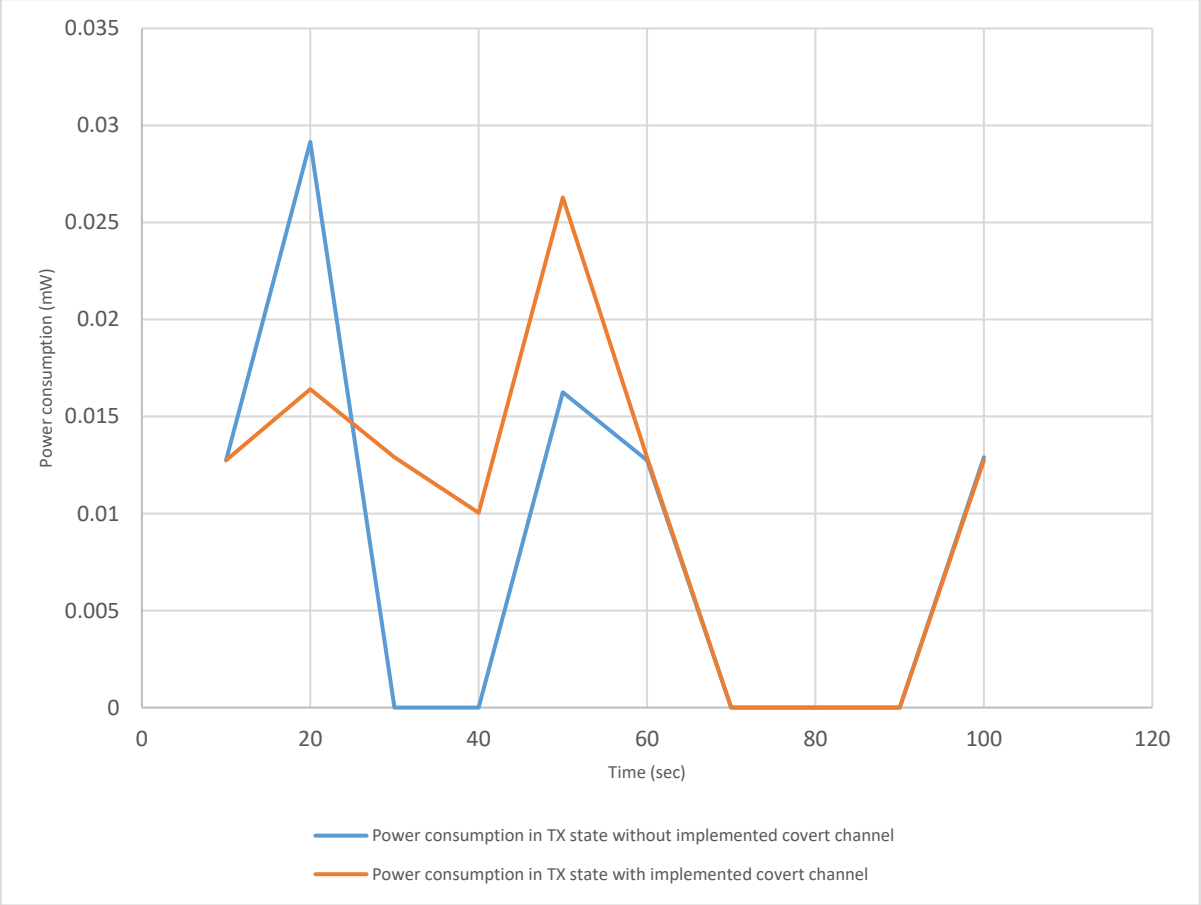


Слика 4.8 Потрошувачка на енергија за CoAP-сервер (Z1) во состојба LPM (со и без имплементиран скриен канал)

Figure 4.8 Power consumption for CoAP server (Z1) in LPM state (with and without implemented covert channel)

Може да се види дека просечната потрошувачка на енергија во случај кога имаме имплементиран скриен канал е малку помала од случајот без имплементиран скриен канал.

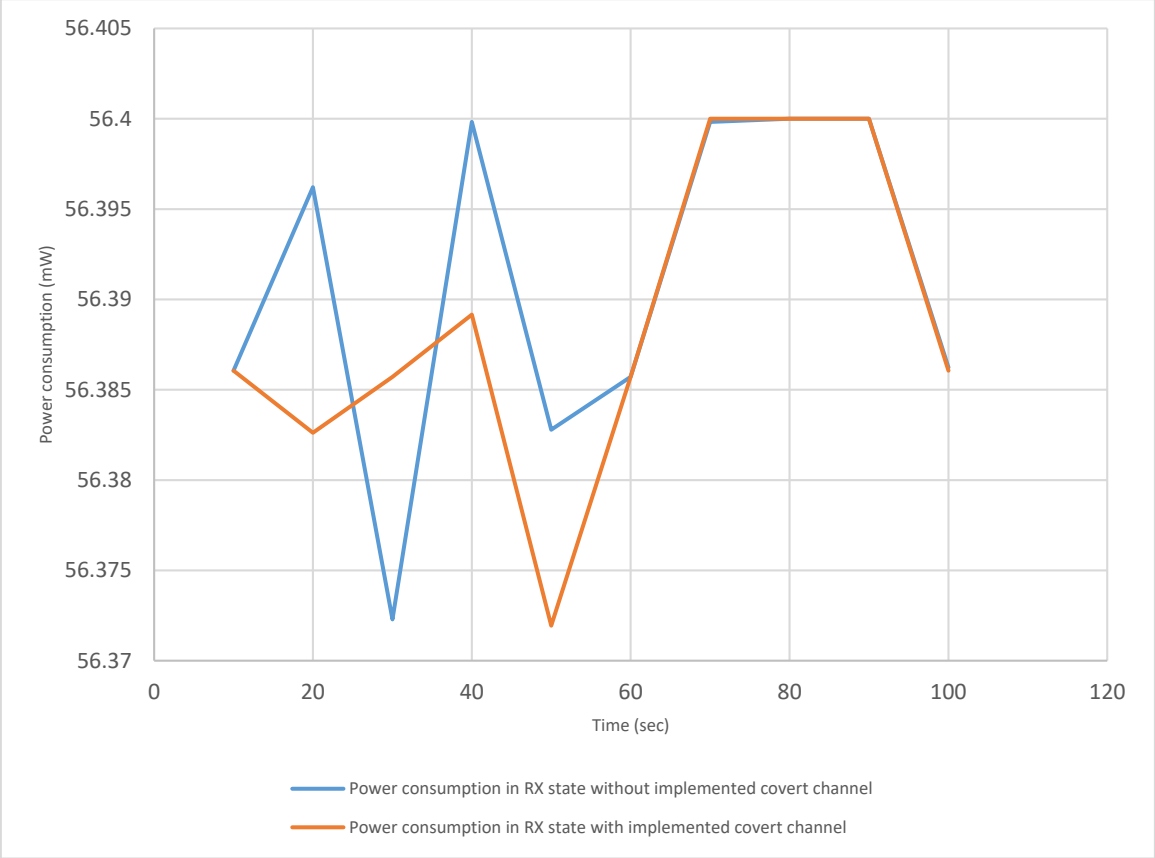
Слика 4.9 ја прикажува потрошувачката на енергија за Z1 модул (имплементиран како CoAP-сервер) во состојба TX со и без имплементиран скриен канал. Просечната потрошувачка на енергија без имплементиран канал е 0.008379272 mW, додека просечната потрошувачка со имплементиран канал е 0.010402405 mW. Може да се забележи дека потрошувачката на енергија со имплементиран канал е 1,24 пати поголема од случајот без имплементиран скриен канал.



Слика 4.9 Потрошувачка на енергија за CoAP-сервер (Z1) во состојба TX (со и без имплементиран скриен канал)

Figure 4.9 Power consumption of CoAP server (Z1) in TX state (with and without implemented covert channel)

Слика 4.10 ја прикажува потрошувачката на енергија за Z1 модул (CoAP-сервер) во состојба RX со и без имплементиран скриен канал. Просечната потрошувачка на енергија без имплементиран скриен канал е 56.3908949 mW, додека просечната потрошувачка во случајот со имплементиран скриен канал е 56.3887262 mW. Од добиените резултати може да се забележи дека просечната потрошувачка на енергија во случајот со имплементиран скриен канал е малку помала во однос на случајот без имплементиран скриен канал.



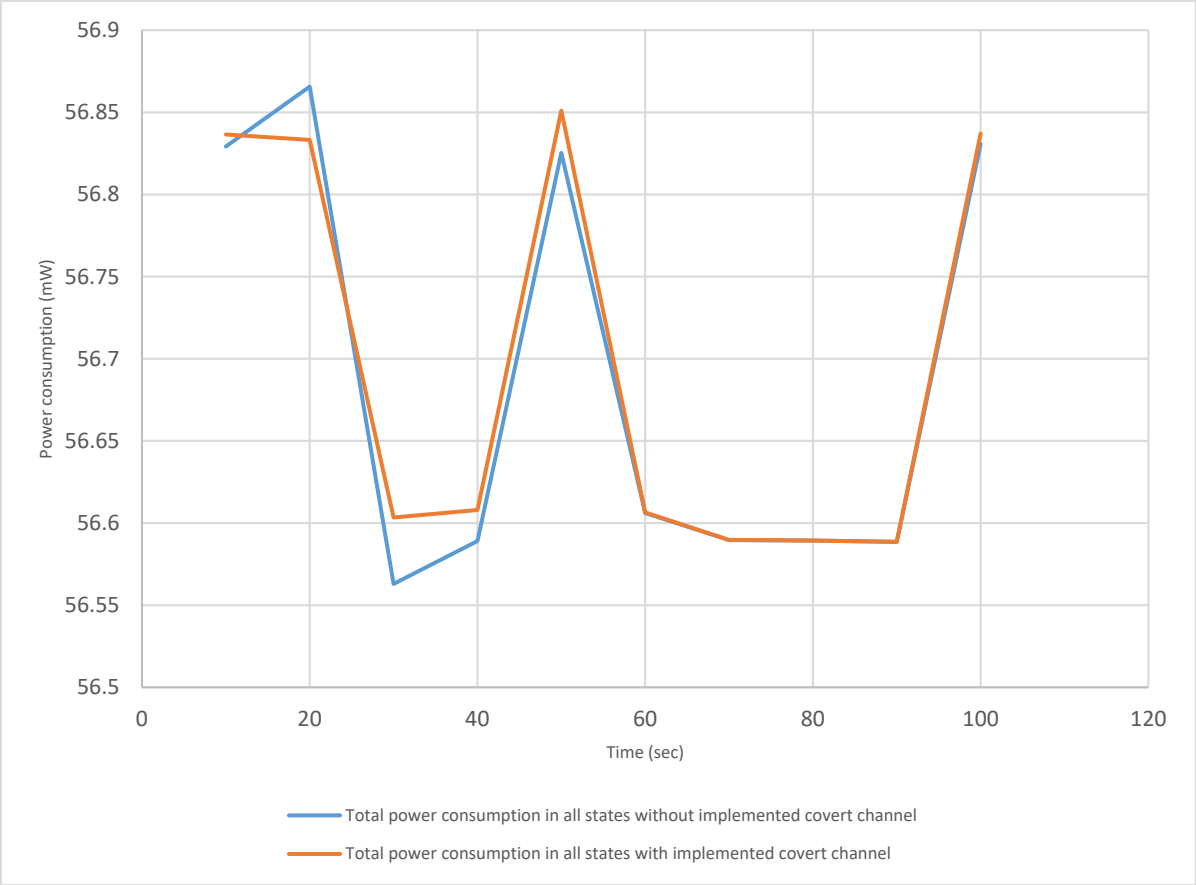
Слика 4.10 Потрошувачка на енергија за CoAP-сервер (Z1) во состојба RX (со и без имплементиран скриен канал)

Figure 4.10 Power consumption for CoAP server (Z1) in RX state (with and without implemented covert channel)

Слика 4.11 ја прикажува вкупната потрошувачка на енергија за Z1 модул (имлементиран како CoAP-сервер) во сите состојби, во секој временски интервал со и без имплементиран скриен канал. Просечната потрошувачка на енергија без имплементиран скриен канал е 56.68764088mW, додека просечната потрошувачка со имплементиран скриен канал е 56.69432571mW. Може да се забележи дека просечната потрошувачка на енергија во сите состојби за Z1 модул со имплементиран скриен канал (кога се праќаат два бита) е малку поголема во однос на случајот без имплементиран скриен канал.

Потрошувачката на енергија на Z1 модул во 50-тата секунда (времето во кое ние праќаме барање со PUT-методот) со имплементиран скриен канал е зголемена многу малку, само за 0.02580548 mW.

Во однос на 60-тата секунда (времето во кое ние праќаме барање со DELETE методот), потрошувачката на енергија со имплементиран скриен канал е зголемена исто така многу малку, за само 0.00040648 mW. Имплементацијата на скриениот канал со користење на PUT и DELETE-методите не влијае многу на потрошувачката на енергија на Z1 модулот.



Слика 4.11 Вкупна потрошувачка на енергија за CoAP-сервер (Z1) во сите состојби (со и без имплементиран скриен канал)

Figure 4.11 Total power consumption of CoAP server (Z1) in all states (with and without implemented covert channel)

Предложените скриени канали се добри за испраќање на кратки пораки според дадената евалуација на перформансите. Притоа, според истражувањата кога се испраќаат два бита со имплементиран скриен канал немаме големо зголемување на потрошувачката на енергија. Со оглед на големиот број на уреди кај интернет на нештата, се очекува дека CoAP-протоколот ќе се користи доста во иднина. Затоа идентификувањето на можни скриени канали кај CoAP и

наоѓањето на начини за нивно детектирање и спречување е доста важно. Во овој дел претставени беа 8 скриени канали.

4.4 Противмерки за предложените скриени канали кај протоколот CoAP

За подобрување на безбедноста кај интернет на нештата важно е да се обезбедат соодветни механизми за детекција како и противмерки за скриените канали.

Скриениот канал кој ги користи полињата „Message ID“ и „Token“ може да се детектира со надгледување на вредностите за овие полиња. Поставувањата на вредности од иста IP-адреса и MAC-адреса може да биде показател за постоење на овој скриен канал.

Скриениот канал кој користи „Piggybacked“ или посебен одоговор може да се детектира со набљудување на одговорите кои ги испраќа серверот. Доколку постои испраќање на многу „Piggybacked“ или посебни одговори до клиентот од страна на серверот, во тој случај постои можност за постоење на овој скриен канал.

Детектирањето на скриениот канал кој го користи товарот (payload) на пораката може да се направи со набљудување на дијагностичкиот товар (Diagnostic Payload). Доколку имаме често испраќање на пораки со дијагностички товар, повремено модулирање на клиентски и серверски грешки, често појавување на исти дијагностички пораки, сето ова може да биде показател за постоење на овој скриен канал.

За скриениот канал којшто користи „case-insensitive“ делови од униформните идентификатори на ресурси (URIs), детектирањето може да се направи со набљудување на овие делови. Доколку во опциите „URI-host“ и шема имаме често појавување на големи и мали букви, и притоа барањата потекнуваат од иста IP или MAC-адреса, ова може да значи постоење на овој скриен канал. Ова важи и за користење на опцијата Proxy-URI кога се користи посредник.

Скриениот канал кој ги користи методите „PUT“ и „DELETE“ може да се детектира со набљудување на методите во пораките. Доколку имаме често користење на овие два метода, во тој случај тоа може да означува постоење на

овој скриен канал. Ажурирањето на ресурсите со „PUT“ методот може да се прави често со оглед на природата на уредите кај интернет на нештата, но доколку ова е пропратено со пораки кои го содржат „DELETE“ методот ова би можело да биде невообичаено однесување и можно постоење на скриен канал.

Скриениот канал кој ја користи опцијата „Accept“ може да се детектира со набљудување на пораките кои се испраќаат од страна на клиентот. Доколку имаме често повторување на пораки кои ја содржат опцијата „Accept“ и пораки кои не ја содржат оваа опција и се испратени од уред со иста IP или MAC-адреса, во тој случај ова може да претставува показател за постење на овој скриен канал.

За скриениот канал којшто користи условни барања, доколку имаме често испраќање на пораки со опции како „If-Match“ или „If-Non-Match“, и тие потекнуваат од уред со иста IP или MAC адреса, ова може да претставува тајно пренесување на пораки преку скриен канал. Од друга страна, доколку серверот често обработува или не обработува барања од клиентот ова може да се разгледува како невообичаено однесување.

Скриениот канал со повторни испраќања може да се детектира со набљудување на пораките кои се испраќаат според тоа дали се испраќаат само еднаш или повеќе од еднаш. Доколку ова сценарио постојано се повторува, може да биде показател за овој скриен канал.

За детекција и спречување на претходно споменатите скриени канали може да се користат пасивни и активни чувари. Пасивните чувари ги анализираат сите пораки со цел да извршат детектирање на скриените канали. Активните чувари ги модификуваат пораките со цел да извршат отстранување на скриените канали.

5. СКРИЕНИ КАНАЛИ КАЈ ПРОТОКОЛОТ MQTT-ВЕРЗИЈА 3.1.1

Во последно време за пренесување на податоци за уредите кај интернет на нештата најмногу се користи MQTT-протоколот. MQTT-протоколот бил откриен во 1999 година од Andy Stanford-Clark (IBM) и Arlen Nipper (Arcom, now Cirrus Link). Во тој период ним им бил потребен протокол кој ќе обезбеди минимална потрошувачка на батерија и минимален пропусен опсег за да се поврзат со нафтоводите преку сателит (IBM, 1999). Оттогаш па наваму постојат повеќе случаи на користење на MQTT. Тој се користи за пренесување на податоци и пораки во автомобилската индустрија, за транспорт и логистика, во преработувачката индустрија и сл. Некои производители на автомобили како BMW, Audi и др. го користат MQTT протоколот за да креираат брзи, доверливи и скалирачки платформи за автомобилите (HiveMQ, 2019). Matternet го користи MQTT за мониторирање во реално време на нивните автономни дронови (HiveMQ, 2020a). Шведската компанија Berlex го користи MQTT за поврзување на сообраќајната сигнализација со облак (HiveMQ, 2020b). Flo Technologies го користи MQTT за пренесување на податоци за да превенира оштетувања предизвикани од вода. Овој протокол се користи и за мониторирање на поплави (Rompas et al., 2017). Во 2012 година MQTT бил применет и кај апликацијата Facebook Messenger. Тој овозможил брзо пренесување на пораки како и оптимизација на батеријата и пропусниот опсег. Притоа, пренесувањето на пораки се одвивало во стотици милисекунди, наместо за неколку секунди како порано (Facebook Engineering, 2011). Сите овие случаи како и многу други придонесуваат овој протокол да го добие епитетот стандарден протокол за интернет на нештата.

Уредите кај интернетот на нештата може да се поврзат на MQTT-сервер и со негова помош може да разменуваат пораки. Дополнително, поради автоматизација, тој обезбедува паметен центар, кој ги организира сите уреди, обезбедува логика и панел преку кој корисникот може да ги контролира уредите и тоа локално или од далечина (на пример преку мобилен телефон).

Фондацијата Shadowserver, како дел од EU CEF-проектот именуван како VARIoT (Vulnerability and Attack Repository for IoT), на дневна база извршува IPv4-скенирање за јавно достапни MQTT брокерски сервиси кои се овозможени на

порта 1883/TCP. Според нивниот извештај од 12 март 2020 година, од 71,508 IP адреси коишто одговориле на пратени проверки 48.558 брокерски инстанци овозможувале анонимен пристап (Shadowserver, 2021). Според друго истражување на Avast во 2018 година со помош на Shodan IoT-машината за пребарување било покажано дека повеќе од 49.000 MQTT сервери биле јавно видливи на интернет поради погрешно конфигуриран MQTT-протокол. Од нив повеќе од 32.000 немале заштита со лозинка (Avast Blog, 2018). Дополнително, можно е следење и на локацијата на мобилните уреди кои се користат за контрола на далечина. Ако имаме нарушување на контролата на пристап, можно е објавување во повеќе теми, вметнување на лажни податоци и извршување на „напади со повторување“. Позлонамерен напад може да настане доколку некој добие пристап до панелот на паметниот центар (вообичаено кога се извршува на иста машина како и MQTT-серверот), со што може да се добие контрола до сите прикачени уреди на MQTT-серверот. Ова значи дека напаѓачите може да пристапат до нив без многу напори и да овозможат проток на пораки преку нив. Некој дури може да се поврзе на отворен и незаштитен MQTT-брокер, да се претплати на „#“, и да ги прими сите пораки кои се испратени на одредени теми на тој брокер. Со ова напаѓачот може да го види статусот на сензорите на прозорците, системите за греење и ладење, прекинувачите и сл. Ова може да доведе до нарушување на приватноста како и злоупотреба на информациите како на пример крадење на идентитет и детално набљудување на канцеларии за индустриска шпионажа. Ситуацијата е дури и полоша бидејќи поголем број од корисниците немаат поставено контрола за пристап во конфигурацијата на брокерот. Според ова напаѓачите може да објавуваат пораки на темите кои се достапни на серверите и на тој начин да добијат контрола на сите поврзани уреди (на пример во сценарио на паметен дом). Овие незаштитени сервери може да се искористат како невини соучесници за да се овозможи создавање на ново креирани скриени канали. Дополнително, ваквите MQTT-базирани скриени канали може да се искористат за креирање на тајна комуникација со злонамерен софтвер како што е дадено и за други типови на мрежен сообраќај и средини (Cabaj et al., 2018; Mazurczyk et al., 2015). Повеќето од познатите IoT-облак платформи коишто имаат поддршка за MQTT имаат безбедносни проблеми, посебно во делот за споделување на уреди (Jia et al., 2020) како споделување

на паметни брави помеѓу туристи во хотели, изнајмувачи на станови, домашни посетители и сл.

Безбедноста на паметните домови е тема за која постојат повеќе студии (Kumar & Patel, 2014; Komninos et al., 2014). За некои од комуникациските протоколи кои се користат кај паметните домови постојат мрежни безбедносни анализи (Glanzer et al., 2016; Granzer et al., 2006; Müller et al., 2016; Badenhop et al.).

Според направеното истражување, сè уште не постојат студии кои имаат за цел да ги истражат мрежните скриени канали кои може да се креираат за пренесување на скриени пораки преку MQTT-протоколот. Ова е причината поради која во ова поглавје се прикажува една систематска студија за можните скриени канали за MQTT-базирани средини кај интернет на нештата.

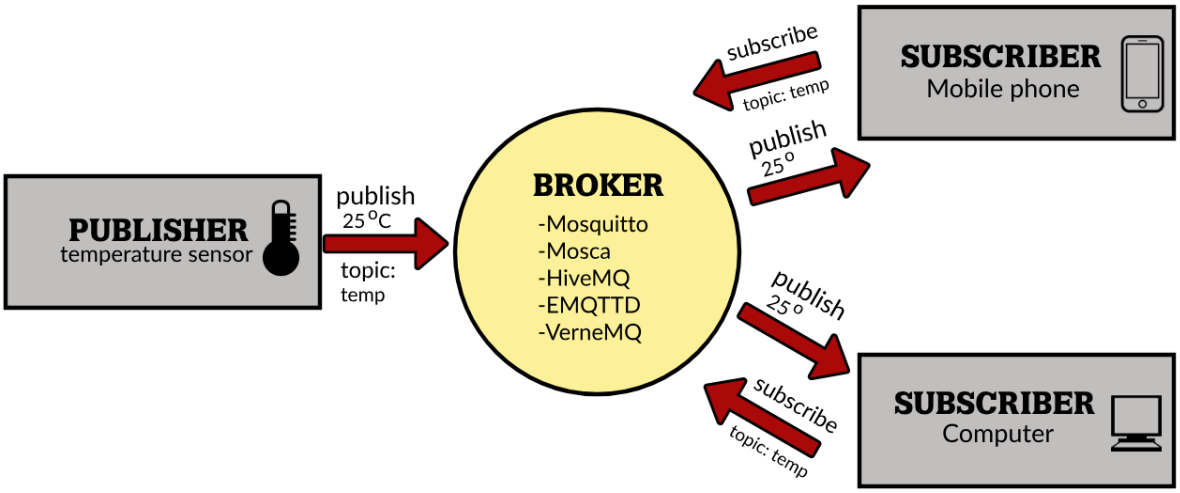
Новите придонеси за научната сфера кои се претставени во ова поглавје се:

- Предложени се 13 мрежни скриени канали за MQTT-протоколот. Во основа, претставени се 7 директни (каде тајниот испраќач и тајниот примач за да може да пренесуваат скриени пораки мора да бидат активни истовремено) и 6 индиректни скриени канали (каде тајните учесници во комуникацијата може да не бидат активни во исто време и тие во основа користат невин посредник за да ги пренесат скриените пораки.)
- Направена е евалуација и категоризација на предложените скриени канали со користење на шеми за криење на информации кои се претставени во глава 2 (Wendzel et al., 2015). Тие се опишани со користење на заедничкиот модел за опишување (Mazurczyk et al., 2016). Покрај ова, откриен е еден скриен канал кој претставува претходно непозната потшема.
- Со цел да се докаже дека MQTT - базираните скриени канали се изводливи и ефективни за IoT средини, имплементирана е избрана шема за криење на податоци и извршена е нејзина експериментална евалуација со користење на три параметри: пропусен опсег, робусност и неоткривање.

5.1 Основи на MQTT

MQTT е клиент-сервер, објави-претплати протокол за пренесување на пораки којшто е погоден за M2M/IoT поврзување но неговата намена може да биде и поширока. Моделот „објави-претплати“ кој е применет кај MQTT може да даде придонес и во други области (Gonzalez-Jimenez et al., 2013). Тој е дизајниран за уреди кои имаат ограничени ресурси, но исто така е соодветен и за мобилни апликации. Некои од претходните имиња на протоколот се: „SCADA протокол“, „MQ Integrator SCADA Device Protocol (MQIsdp)“ и „WebSphere MQTT (WMQTT)“. Библиотеки за MQTT се достапни за повеќе програмски јазици и платформи, како C, C++, C#, Java, Javascript, .NET итн.

MQTT-верзијата 3.1.1 во 2014 година станува OASIS-стандард (OASIS, 2014), додека во 2016 година станува ISO/IEC-стандард 20922:2016. Објави/претплати моделот кај протоколот MQTT е прикажан на слика 5.1. Во основа, тоа е клиент/сервер модел кој овозможува клиентите да комуницираат со крајна точка. Кај овој протокол постојат два типа на клиенти и тоа: објавувачи и претплатници. Клиентите кои испраќаат пораки се нарекуваат објавувачи. Другите клиенти кои ги добиваат пораките се нарекуваат претплатници. Овие два типа на клиенти не комуницираат директно едни со други. За да може да разменуваат пораки тие користат централен ентитет кој игра улога на сервер и се нарекува брокер.



Слика 5.1 MQTT објави-претплати модел

Figure 5.1 MQTT publish-subscribe model

Основната улога на брокерот е да ги прими пораките кои се испратени од страна на објавувачите и да ги препрати до претплатниците. Објавувачите ги испраќаат пораките на одредени теми. Претплатниците исто се претплатуваат на темите за кои сакаат да добиваат пораки. Кога брокерот ќе добие порака од страна на даден објавувач, ја утврдува темата на која е испратена пораката и истата ја препраќа до сите клиенти кои се претплатени на таа тема.

Темата (topic) на дадена порака може да се разгледува како предмет на пораката и таа претставува произволен UTF-8 кодиран стринг. Таа исто така може да биде хиерархиски структурирана со користење на знакот коса црта („/“), кој се користи како одделувач на нивоа. Според ова, темата може да има едно или повеќе нивоа. Пример за тема со три нивоа е следната:

myHome/bedroom/light

Не е потребно да се направи првична иницијализација на темата, односно не треба темата да се креира пред да се изврши објавување или претплатување. За да се креира тема потребен е најмалку еден знак. Имињата на темите се чувствителни на знаци (case-sensitive) и може да содржат празни места. Коса црта е исто така валиден стринг за тема. Клиентите исто така може да користат и „wildcards“ за да се претплатуваат на повеќе теми. Притоа постојат два типа на „wildcards“ и тоа: со едно ниво („single-level“) или со повеќе нивоа („multi-level“). За креирање на „wildcard“ со едно ниво се користи симболот „+“. Тој заменува само едно ниво. Пример во темата myOffice/+/light, симболот „+“ може да се замени со кој било стринг. За оваа тема валидни се следните примери:

myOffice/office1/light

myOffice/FirstFloor/light

Типот на „wildcard“ со повеќе нивоа се креира со користење на хаш знакот („#“). Со овој знак може да бидат заменети повеќе нивоа. Пример во темата myOffice/office2/#, симболот „#“ може да се замени со повеќе нивоа. Валидни примери за оваа тема би биле:

myOffice/office2/a1/temperature

myOffice/office2/a2/a2a/temperature

Во MQTT има теми кои се наменети за специјални цели и тие започнуваат со знакот $. Објавувачите не може да праќаат пораки на овие теми. Најчести теми од овој тип се оние кои започнуваат со $SYS, кои се користат за

прикажување на информации кои се специфични за брокерот. Некои од овие теми се прикажани во табела 5.1.

Табела 5.1 Теми кои прикажуваат специфични информации за брокерите

Table 5.1 Topics that display specific information about brokers

Специфична тема	Опис
$SYS/broker/clients/connected	Број на моментално поврзани клиенти
$SYS/broker/clients/disconnected	Вкупен број на постојани клиенти (со исклучена чиста сесија) кои се регистрирани на брокерот но моментално се дисконектирани
$SYS/broker/load/bytes/sent	Вкупен број на испратени бајти откако брокерот е стартуван
$SYS/broker/load/bytes/received	Вкупен број на примени бајти откако брокерот е стартуван
$SYS/broker/clients/maximum	Максималниот број на активни клиенти кои биле поврзани на брокерот
$SYS/broker/clients/total	Вкупен број на конектирани и дисконектирани клиенти со постојана сесија моментално поврзани и регистрирани на брокерот
$SYS/broker/messages/received	Вкупен број на пораки од каков било тип кои се примени откако брокерот е стартуван
$SYS/broker/messages/sent	Вкупен број на испратени пораки од кој било тип откако брокерот е стартуван
$SYS/broker/messages/publish/dropped	Вкупен број на објавени пораки што биле отфрлени поради ограничувања за редот

$SYS/broker/messages/retained/count	Вкупен број на задржани пораки (retained messages) кои се активни на брокерот
$SYS/broker/subscriptions/count	Вкупен број на претплати активни на брокерот
$SYS/broker/time	Моменталното време на серверот
$SYS/broker/uptime	Времето (во секунди) во кое брокерот бил онлајн
$SYS/broker/version	Верзија на брокерот

Некои од добрите практики за користење на теми се следните:

- Никогаш не користете водечка коса црта - Водечка коса црта е дозволена во MQTT. На пример, /office/office2/a1.

  Ова доведува до користење на непотребно ниво со нул знак напред и често доведува до конфузија.
- Да не се користат празни места во тема - Празните места ги прават темите тешки за читање. Како и за добрата практика да не се користат водечки коси црти така и овде, доколку нешто е дозволено не значи дека треба да се користи.
- Темата треба да биде кратка и јасна - Секоја тема е вклучена во порака која се испраќа. Кога станува збор за мали уреди, во тој случај секој бајт е важен и должината на темата има големо влијание. Затоа, темите треба да бидат кратки и јасни.
- Да се користат само ASCII знаци, да се избегнуваат знаци кои не може да се печатат - Бидејќи не-ASCII знаците често се прикажуваат несоодветно, многу е тешко да се пронајдат грешки кои се поврзани со множеството знаци. Сè додека не е апсолутно потребно, препорачливо е да се избегнува користење на не-ASCII-знаци во тема.
- Вгнездување на единствен идентификатор или клиентски идентификатор во тема - Може да биде многу значајно да се изврши вклучување на единствен идентификатор во тема на дадена порака. Овој идентификатор помага во идентификување на тоа кој ја испратил пораката. Вгнезденото ID може да се користи за авторизација. Само клиентот кој има исто ID како

и ID-то во темата, може да објавува пораки на таа тема. Пример, на клиент со ID client1 му е дозволено да објавува на тема client1/office, но не му е дозволено да објавува на тема client2/office.

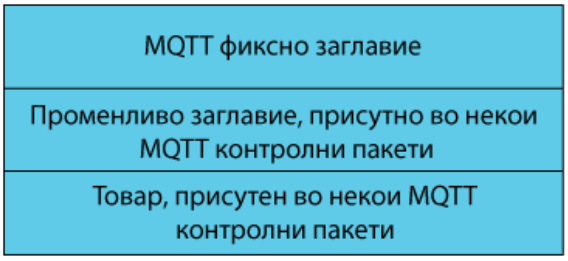
- Да не се претплатуваме на „#“ - Понекогаш е потребно да се претплатиме за сите пораки кои се пренесуваат преку брокерот, на пример за да извршиме внесување на пораките во база на податоци. Сепак ова претплатување со користење на „wildcard“ со повеќе нивоа не е препорачливо. Претплатениот клиент нема да може да ги процесира сите пораки кои се испратени и би можело да дојде до преоптоварување.
- Да не се заборави проширувањето на темите - Темите се флексибилен концепт. Важно е да се размислува за тоа како темите во иднина може да бидат проширени за да се овозможат нови особини или уреди. На пример, доколку имаме паметен дом и додадеме нови сензори во тој случај треба да размислуваме како истите можеме да ги додадеме во дрвото на теми без да се менува целата хиерархија.
- Користење специфични теми, не генерални - Темите треба да се разликуваат колку што е можно повеќе. На пример доколку во домот имаме три сензори:
  home/bedroom/brightness
  home/bedroom/temperature
  home/bedroom/humidity
  Во овој случај не треба сите вредности да се праќаат преку home/bedroom. Користењето на една тема за сите пораки не се препорачува. Специфичното именување овозможува користење на MQTT-особини како задржани пораки.

MQTT v.3.1.1 користи 14 различни контролни пакети (таб. 5.2). Тие се нумерирани со броеви од 1 до 14 и нивната максимална големина е 256 MB. Секој контролен пакет има фиксно заглавие (со типот на контролниот пакет, знаменца кои се специфични за него и останата должина). Некои контролни пакети имаат променливо заглавие и/или товар (сл. 5.2).

Табела 5.2 Контролни пакети кај MQTT

Table 5.2 Control packets in MQTT

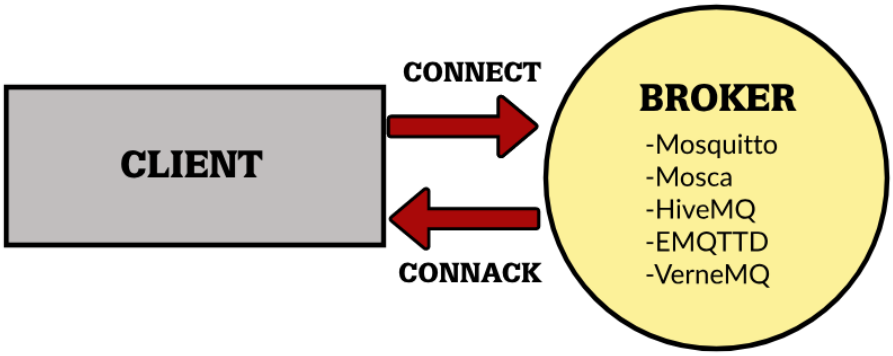
Контролен пакет	Опис
CONNECT	Клиентот испраќа барање за воспоставување врска со серверот
CONNACK	Серверот го потврдува барањето за врска од клиентот
PUBLISH	Објавување на порака
PUBACK	Потврда за објава
PUBREC	Објавата е примена (QoS 2, објавата е примена, дел 1)
PUBREL	Одговор на PUBREC-пакетот (QoS 2, објавата е примена, дел 2)
PUBCOMP	Објавата е комплетирана (QoS 2, објавата е примена, дел 3)
SUBSCRIBE	Клиентот се претплатува на теми
SUBACK	Серверот го потврдува барањето за претплата на клиентот
UNSUBSCRIBE	Клиентот ја прекинува претплатата за темите
UNSUBACK	Серверот го потврдува барањето за прекинувањето на претплатата на клиентот
PINGREQ	Клиентот испраќа PING-барање до серверот
PINGRESP	Серверот испраќа PING-одговор до клиентот
DISCONNECT	Клиентот испраќа известување за прекинување на врската до серверот за чиста дисконекција



Слика 5.2 Генерална структура на MQTT-контролен пакет

Figure 5.2 General structure of MQTT control packet

MQTT-протоколот е TCP/IP-базиран протокол и клиентот и брокерот треба да имаат имплементирано TCP/IP-склад. За транспорт на MQTT преку TCP резервирана е портата 1883. За транспорт на MQTT преку SSL/TLS резервирана е портата 8883.



Слика 5.3 Воспоставување на MQTT-врска

Figure 5.3 Establishment of MQTT connection

Клиентот испраќа CONNECT контролен пакет до брокерот, за да иницира врска. Брокерот одговара со CONNACK контролен пакет и статусен код (сл. 5.3), и ја чува конекцијата сè додека клиентот не му испрати контролен пакет DISCONNECT или додека не настане прекин на конекцијата.

Во многу случаи на користење MQTT-клиентот е лоциран зад рутерот којшто користи NAT (Network Address Translation) за преведување на приватната мрежна адреса во јавно достапна адреса. Како што беше споменато претходно, клиентот иницира конекција со испраќање на CONNECT-контролен пакет до брокерот. Бидејќи брокерот има јавна адреса и ја чува врската отворена за да овозможи двонасочно испраќање и примање на податоци, нема проблем со клиентите кои се наоѓаат зад NAT.

Ако пакетот CONNECT е неправилен (според MQTT-спецификацијата) или е поминато долго време помеѓу отворањето на мрежниот сокет и праќањето на пораката, во тој случај брокерот ја затвора конекцијата. Ова однесување ги спречува злонамерните клиенти кои може да го забават брокерот и да го нарушат неговото функционирање.

Контролниот пакет CONNECT се состои од следните полиња:

- clientID – Врши идентификување на секој MQTT-клиент кој се поврзува на MQTT-брокер. Брокерот го користи овој идентификатор за да го идентификува клиентот и моменталната состојба на клиентот. Затоа овој идентификатор треба да биде единствен за клиентот и за брокерот. Кај MQTT 3.1.1 може да се испрати празно поле за „clientID“, доколку не е потребно состојбата да биде одржувана од брокерот. Ова ќе резултира со конекција без состојба. Во тој случај знаменцето „cleanSession“ треба да биде поставено на вредност „true“, бидејќи доколку е „false“ во тој случај брокерот ќе ја затвори конекцијата.
- cleanSession – Ова знаменце му кажува на брокерот дали клиентот сака да воспостави постојана сесија или не. Ако ова поле е поставено на вредност „false“, брокерот ги чува сите претплати и сите пропуштени пораки за клиентот којшто се има претплатено на темите со квалитет на сервис 1 или 2. Ако сесијата не е постојана, односно кога вредноста за ова поле е „true“, брокерот не чува ништо за клиентот и ги брише сите информации од некои претходни постојани сесии.
- username и password – Се користат за клиентска автентикација и авторизација. Ако оваа информација не е шифрирана или хеширана, лозинката се праќа како чист текст. Се препорачува користење на кориснички имиња и лозинки заедно со безбеден транспорт. Брокерите може да направат автентикација и со SSL-сертификат и во тој случај не се потребни кориснички имиња и лозинки.
- lastWillTopic – Оваа особина е дел од „LastWill and Testament - LWT“ особината на MQTT. Оваа порака ги известува другите клиенти кога настанува неочекувано прекинување на конекцијата со брокерот. Кога даден клиент се поврзува тој може да изврши поставување на порака која брокерот ја испраќа на страна на клиентите кога настанува прекин.
- keepAlive – Ова е временски интервал во секунди кој клиентот го специфицира со брокерот кога врската е воспоставена. Овој интервал го дефинира најдолгиот временски период кој брокерот и клиентот ќе може да го издржат без да испратат порака. Клиентот испраќа редовни PING-пораки до брокерот. На ова брокерот возвраќа со PING-одговор. Овој

метод овозможува едната страна да одреди дали другата страна сè уште е достапна.

Во основа тоа се сите информации коишто се потребни за да може еден MQTT 3.1.1-клиент да се поврзе на MQTT-брокер. Некои индивидуални библиотеки може да имаат и дополнителни опции кои може да се конфигурираат. Пример за ова може да биде начинот на кој се складирани пораките кои се наоѓаат во редица во специфична имплементација.

Сесијата помеѓу клиентот и брокерот може да биде постојана или непостојана. Непостојана сесија или чиста сесија се креира со поставување на знаменцето Clean Session во контролниот пакет CONNECT на вредност 1. Ако врската се изгуби или е прекината, сите информации за клиентот кои постојат на страната на брокерот се губат, и клиентот мора повторно да се претплати на секоја тема. Повторното претплатување на теми при прекин на врската е дополнителен товар за ограничените клиенти кои имаат ограничени ресурси. За да се избегне овој проблем клиентот може да побара постојана сесија кога тој ќе се поврзе со брокерот. Постојаните сесии ги зачувуваат сите информации кои се релевантни за клиентот на брокерот. Клиентскиот идентификатор (clientID) којшто клиентот го обезбедува кога воспоставува врска со брокерот врши идентификување на сесијата.

Во постојаната сесија брокерот ги складира следните информации:

- Постоење на сесијата (дури и ако нема претплати)
- Сите претплати на клиентот
- Сите пораки со квалитет на сервис 1 или 2 што клиентот сè уште не ги потврдил
- Сите нови пораки со квалитет на сервис 1 или 2 коишто клиентот ги пропуштил додека бил исклучен
- Сите пораки со квалитет на сервис 2 кои биле примени од клиент кои сè уште не се целосно потврдени

Кога клиентот се поврзува повторно, овие информации се достапни веднаш. Брокерот ја складира сесијата сè додека клиентот не се вклучи повторно за да ги прими пораките.

Слично како и брокерот, секој MQTT-клиент мора исто така да складира постојана сесија. Кога клиентот бара од серверот да ги чува сесиските податоци, клиентот е одговорен за складирање на следните информации:

- Сите пораки со квалитет на сервис 1 или 2, кои сè уште не се потврдени од брокерот
- Сите пораки со квалитет на сервис 2 примени од брокерот кои сè уште не се целосно потврдени

Добри практики при користење на постојани сесии или чисти сесии се следните:

Постојани сесии:

- Клиентот мора да ги добие сите пораки за одредена тема, дури и ако е исклучен. Брокерот ги става пораките во редица за клиентот и ги доставува веднаш откако клиентот ќе стане достапен
- Клиентот треба да ги поврати сите пораки со квалитет на сервис 1 или 2 откако повторно ќе се поврзе
- Клиентот има ограничени ресурси. Брокерот ќе изврши складирање на информациите за претплатата на клиентот и може да ја поврати брзо прекинатата комуникација.

Чисти сесии:

- Клиентот не треба да ги добие пораките кои ги пропуштил додека бил исклучен
- Клиентот треба само да објавува пораки на темите и не треба да се претплатува на нив. Во овој случај не е потребно складирање на сесиски информации и не е потребно повторно испраќање на пораките кои не биле доставени.

По примање на контролниот пакет CONNECT, брокерот одговара со CONNACK-контролен пакет. CONNACK се состои од следните полиња:

- sessionPresent – му кажува на клиентот дали брокерот има постојана сесија која е достапна од претходните интеракции на клиентот. Кога

клиентот се поврзува со „cleanSession“ поставено на вредност „true“, знаменцето „sessionPresent“ во тој случај е поставено на вредност „false“, бидејќи нема достапна сесија. Во друг случај, доколку клиентот се поврзува со вредност „false“ за „cleanSession“, постојат две можности:

- Ако сесиските информации се достапни за клиентскиот идентификатор, и брокерот ги има складирано, знаменцето „sessionPresent“ е поставено на вредност „true“.
- Доколку брокерот нема сесиски информации за дадениот клиентски идентификатор, знаменцето „sessionPresent“ е поставено на вредност „false“. Ова знаменце е додадено во MQTT v.3.1.1 за да им помогне на клиентите да одредат дали треба да се претплатат не темите или темите сѐ уште се складирани во постојана сесија.

- returnCode – Повратниот код му кажува на клиентот дали обидот за воспоставување на врска бил успешен или не. Сите можни повратни кодови се прикажани во табела 5.3.

Табела 5.3 Повратни кодови кај CONNACK

Table 5.3 Return codes in CONNACK

Повратен код	Значење
0	Врската е прифатена
1	Врската е одбиена, неприфатлива верзија на протоколот
2	Врската е одбиена, идентификаторот е одбиен
3	Врската е одбиена, серверот не е достапен
4	Врската е одбиена, погрешно корисничко име или лозинка
5	Врската е одбиена, не е авторизирана

PUBLISH и SUBSCRIBE-контролните пакети се најчесто користени контролни пакети кај MQTT-комуникацијата. Даден MQTT-клиент откако еднаш ќе се поврзе со брокерот, понатаму може да објавува пораки со користење на PUBLISH-контролниот пакет. Секој пакет мора да содржи тема која брокерот ќе ја користи за да ја препрати пораката до заинтересираните клиенти. Во основа, пораката има и товар кој содржи податоци кои ги пренесува во бајт-формат. MQTT е податочно агностичен. Случајот на користење на клиентот одредува

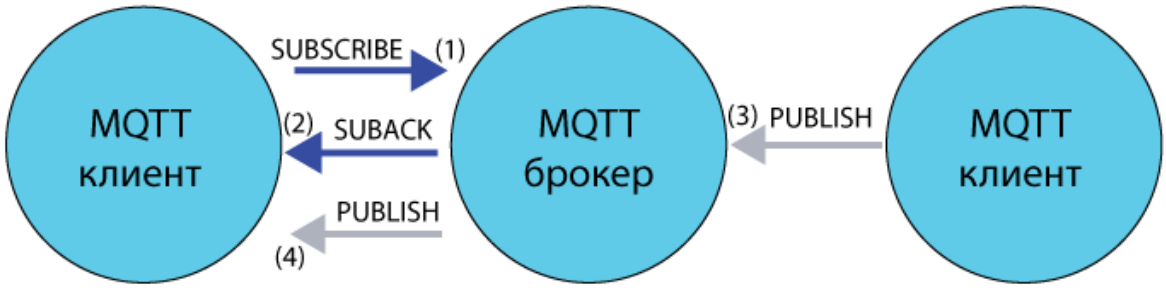
како товарот е структуриран. Клиентот кој испраќа податоци (објавувачот) одлучува за тоа дали податоците ќе ги испрати во бинарен формат, текстуален, XML или JSON. Контролниот пакет PUBLISH покрај фиксното заглавие, има и променливо заглавие кое се состои од:

- Packet Identifier – Идентификаторот на пакетот единствено ја идентификува пораката која клиентот ја испраќа на брокерот. Идентификаторот е релевантен само за ниво на квалитет на сервис кое е поголемо од 0.
- Topic Name – Темата (Topic) е едноставна низа од знаци која е хиерархиски структурирана со коси црти како одделувачи („/“).
- QoS – Го претставува нивото на квалитет на сервис (QoS 0, QoS 1 или QoS 2).
- retainFlag – Ова знаменце дефинира дали пораката се снима на страната на брокерот (задржана порака) како последна позната вредност за специфицираната тема. Кога нов клиент ќе се претплати на таа тема, тој веднаш ќе ја прими последната порака која е означена како задржана за дадената тема.
- Payload – Ова е содржината на пораката. Како што беше споменато претходно, MQTT е податочно агностичен и ова значи дека може да се испраќа текст во какво било кодирање, шифрирани податоци, слики, бинарни податоци и сл.
- dupFlag – Ова знаменце покажува дека пораката е дупликат и повторно е испратена бидејќи примателот (клиент или брокер) не потврдил дека ја примил оригиналната порака. Ова е соодветно само за квалитет на сервис поголем од 0. Вообичаено, механизамот за повторно испраќање и дупликати е управуван од клиентска библиотека или од брокерот како детал од имплементацијата.

Кога даден клиент испраќа порака до MQTT-брокер за објавување, брокерот ја чита пораката, ја потврдува пораката (според нивото на квалитет на сервис) и ја процесира пораката. Процесирањето кај брокерот вклучува одредување кои клиенти се имаат претплатено на дадената тема и испраќање на пораката до нив.

Клиентот којшто врши иницијално објавување на порака е одговорен само за доставување на PUBLISH контролниот пакет до брокерот. Откако еднаш брокерот ќе ја прими пораката, тој е одговорен за доставување на пораката до сите претплатници. Клиентот којшто врши објавување, не добива повратен одговор за тоа дали некој е заинтересиран за објавената порака или колку клиенти ја добиваат пораката од брокерот.

Контролниот пакет SUBSCRIBE е составен од фиксно заглавие, променливо заглавие кое се состои од поле „Packet Identifier“ и нула или повеќе особини и товар кој содржи листа на филтри на теми (Topic Filters), при што по секој од нив следува „Subscription Option“ бајт.



Слика 5.4 Претплата и испраќање на потврда за претплата

Figure 5.4 Subscription and sending acknowledgement for subscription

За потврда на секоја претплата, брокерот испраќа контролен пакет SUBACK до клиентот (сл. 5.4). Пораката го содржи идентификаторот на пакетот на оригиналната SUBSCRIBE порака (за идентификување на пораката) и листа на повратни кодови (таб. 5.4). Брокерот испраќа еден повратен код за секој пар на тема/QoS којшто го прима во SUBSCRIBE порака. На пример, доколку пораката SUBSCRIBE има четири претплати, пораката SUBACK содржи четири повратни кодови. Повратниот код ја потврдува секоја тема и го прикажува квалитетот на сервис којшто е доделен од брокерот. Доколку брокерот ја одбие претплатата, SUBACK содржи код за грешка за специфичната тема. На пример, доколку клиентот нема доволно пермисии да се претплати на некоја тема, или доколку темата не е правилна.

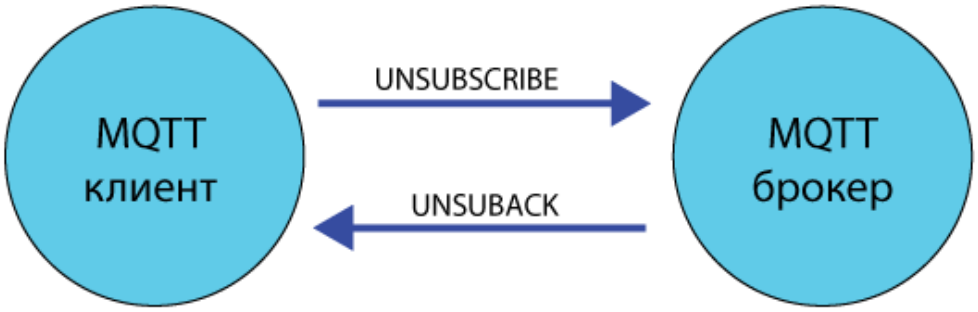
Табела 5.4 Повратни кодови кај SUBACK

Table 5.4 Return codes in SUBACK

Повратен код	Значење
0	УСПЕХ – Максимум QoS 0
1	УСПЕХ – Максимум QoS 1
2	УСПЕХ – Максимум QoS 2
128	Неуспех

Откако клиентот успешно испраќа SUBSCRIBE-контролен пакет и добива потврда SUBACK, понатаму тој ја добива секоја порака којашто одговара на некоја од темите во претплатите кои ги содржи контролниот пакет SUBSCRIBE.

Спротивен на SUBSCRIBE е UNSUBSCRIBE-контролниот пакет. Тој врши бришење на претплатите на клиентот на брокерот. Во однос на структурата тој е сличен на SUBSCRIBE-пакетот и се состои од идентификатор на пакет и листа на теми. Идентификаторот врши идентификување на пораката. Клиентска библиотака и/или брокерот е одговорен за поставување на внатрешен MQTT-идентификатор. Листата на теми може да содржи повеќе теми за кои клиентот сака да ја прекине претплатата. Потребно е само да се испрати темата (без QoS). Брокерот ја прекинува претплатата, без оглед на нивото на квалитет на сервис со кое оригинално е направена претплатата.



Слика 5.5 Прекинување на претплата

Figure 5.5 Unsubscribe

За потврда на прекинувањето на пораката, брокерот испраќа UNSUBACK потврден контролен пакет до клиентот. Овој пакет го содржи само

идентификаторот на пакет на оригиналниот UNSUBSCRIBE-контролен пакет (сл. 5.5).

Кога даден клиент се претплатува на некоја тема по некое последно ажурирање, нормално тој мора да почека за следно ажурирање, за да го види на пример моменталниот статус на даден сензор. Објавувачот пак има можност да му каже на брокерот да ја чува последната порака за одредена тема (задржана порака) со поставување на знаменцето RETAIN во PUBLISH-контролниот пакет на вредност 1. На ваков начин, секој нов претплатник ќе го види последното испратено ажурирање кое е направено со задржана порака. Само една порака може да биде задржана за дадена тема. Претплатникот исто така може да го контролира примањето на задржаните пораки со користење на две опции во „Subscription Options“ коишто следуваат по филтерот за дадена тема. Ако опцијата „Retain As Published“ е поставена на вредност 1 тогаш ажурирањата од серверот го чуваат знаменцето „RETAIN“ со коешто се објавени. Следно, опцијата „Retain Handling“ (2 бита), специфицира кога задржаните пораки ќе бидат испратени на нови претплати (0 или 1 за испраќање на задржани пораки со 1 само во случај кога претплатата моментално не постои, и 2 да не се испраќа).

Една од клучните особини кај MQTT-протоколот е квалитетот на сервис (Quality of Service – QoS). Оваа особина му овозможува на клиентот да избере ниво на квалитет кое одговара на достапноста на мрежата и на апликацискатa логика. Бидејќи MQTT управува со повторното испраќање на пораките и гарантира доставување (дури и кога основниот транспорт не е доверлив), квалитетот на сервис ја прави конекцијата кај недоверливите мрежи многу полесна.

Кога се зборува за квалитет на сервис кај MQTT мора да се разгледаат двете страни за доставување на пораките:

- Доставување на порака од клиентот кој објавува до брокерот
- Доставување на порака од брокерот до клиентот кој е претплатен

Се разгледуваат двете страни за доставување на пораките бидејќи постојат одредени разлики помеѓу нив. Клиентот кој ја објавува пораката до

брокерот го дефинира нивото на квалитетот на сервис кога ја праќа пораката до брокерот. Брокерот ја пренесува пораката до претплатените клиенти со користење на нивото на квалитет на сервис кое секој клиент кој се претплатува го дефинира за време на процесот на претплата. Ако претплатникот дефинира помало ниво во споредба на она кое го дефинира објавувачот, во тој случај брокерот ја пренесува пораката со помалото ниво на квалитет на сервис.

MQTT нуди три нивоа за квалитет на сервис за доставување на пораки од објавувачот до брокерот и од брокерот до претплатниците:

- QoS 0 - најмногу еднаш
- QoS 1 - најмалку еднаш
- QoS 2 - точно еднаш

Слика 5.6 Испраќање на порака со квалитет на сервис 0 (QoS 0)

Figure 5.6 Sending message with quality of service 0 (QoS 0)

Кај QoS 0 не постои гаранција за доставување на пораките. Примателот не го потврдува примањето на пораката и таа не се складира и испраќа повторно од испраќачот. Ова ниво на квалитет на сервис уште е познато како „fire and forget“ (сл. 5.6).

QoS 1 гарантира дека пораката е доставена најмалку еднаш до примателот. Испраќачот ја складира пораката сè додека не добие PUBACK-пакет од примателот, кој го потврдува примањето на пораката. Испраќачот користи идентификатор во секој пакет за да го поврзе PUBLISH-пакетот со соодветниот PUBACK-пакет. Доколку испраќачот не добие PUBACK-пакет во разумно време, испраќачот повторно го праќа PUBLISH-пакетот. Кога

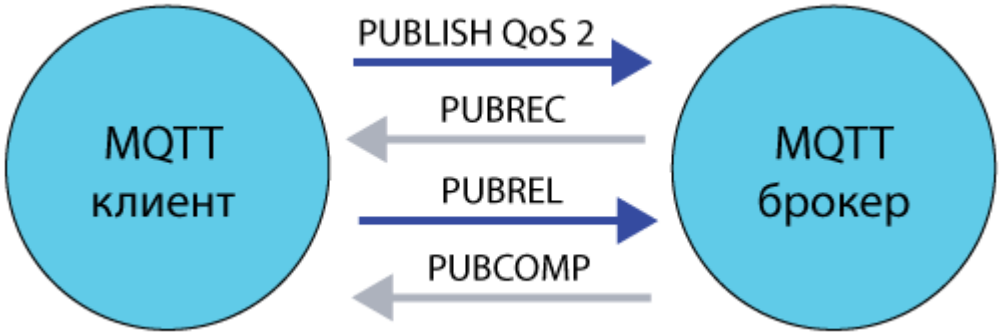
примателот ќе добие порака со QoS 1, тој може веднаш да ја процесира. Ако примателот е брокерот, тој ја препраќа пораката до сите претплатени клиенти и потоа одговара со PUBACK-пакет (сл. 5.7). Доколку пораката е испратена повторно од испраќачот, тој врши поставување на знаменцето „DUP“ (duplicate). Ова знаменце се користи само за интерни цели и не се процесира од страна на клиентот или брокерот. Примателот на пораката испраќа PUBACK-пакет, без оглед на знаменцето „DUP“.

Слика 5.7 Испраќање на порака со квалитет на сервис 1 (QoS 1)

Figure 5.7 Sending message with quality of service 1 (QoS 1)

QoS 2 е највисокото ниво на квалитет на сервис кај MQTT. Ова ниво гарантира дека пораката е примена само еднаш од страна на примателите. Ова ниво е најбезбедното и најбавното ниво на квалитет на сервис. Гаранцијата е обезбедена од најмалку два барање-одговор протоци (ракување од четири дела) помеѓу испраќачот и примателот. Тие го користат идентификаторот на оригиналната PUBLISH-порака за да извршат координирање на доставувањето на пораката. Кога примателот ќе добие QoS 2 PUBLISH-пакет од испраќачот, тој ја процесира пораката и одговара со PUBREC-пакет, кој го потврдува PUBLISH-пакетот. Ако испраќачот не добие PUBREC-пакет, тој повторно го испраќа PUBLISH-пакетот со знаменце „DUP“ сè додека не добие потврда. Еднаш откако испраќачот ќе прими PUBREC-пакет од примателот, тој може безбедно да го отфрли првичниот PUBLISH-пакет. Испраќачот го складира PUBREC-пакетот од примателот и одговара со PUBREL-пакет. Откако примателот ќе добие PUBREL-пакет, тој ги отфрла сите складирани состојби и одговара со PUBCOMP-пакет (сл. 5.8). По праќањето на овој пакет примателот складира референца до идентификаторот на пакетот на оригиналниот PUBLISH-пакет. Овој чекор е

важен за да се избегне процесирање на пораката по втор пат. Откако испраќачот ќе прими PUBCOMP-пакет, идентификаторот на пакет на објавената порака станува достапен за повторно користење. Кога QoS-протокот е комплетиран, двете страни се сигурни дека пораката е доставена и примателот има потврда за доставата. Доколку пакетот не се достави, испраќачот е должен повторно да ја испрати пораката во разумно време. Ова важи и во случај кога испраќач е MQTT-клиент и во случај кога испраќач е MQTT-брокер.



Слика 5.8 Испраќање на порака со квалитет на сервис 2 (QoS 2)

Figure 5.8 Sending message with quality of service 2 (QoS 2)

5.2 Скриени канали кај MQTT-верзија 3.1.1

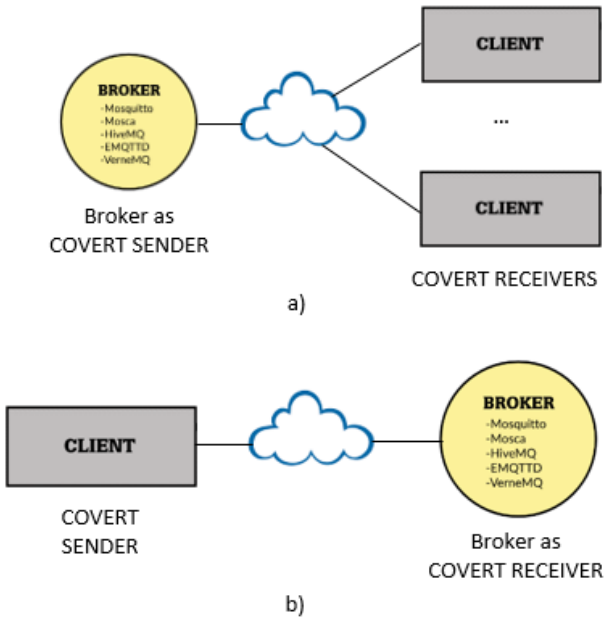
Во овој дел ќе биде претставена сеопфатна анализа на потенцијалните мрежни скриени канали кои се применливи кај протоколот MQTT-верзија 3.1.1. Скриените канали се категоризирани според шемите за криење и опишани според унифицираниот метод за опис (Wendzel et al., 2016).

5.2.1 Системски модел

Системскиот модел за MQTT - базирани скриени канали вклучува таен испраќач (CS) и еден или повеќе тајни примачи (CRs). Можно е да има повеќе тајни испраќачи, но во овој дел разгледуваме само сценарио со еден таен испраќач. Бидејќи кој било клиент може да биде објавувач, комуникацијата може да биде двонасочна, и група на корисници може да комуницираат заедно. Испраќачите и примачите може да бидат на иста мрежа или на различни мрежи.

Во овој дел правиме разлика помеѓу два различни подмодели: директни скриени канали (DCC) и индиректни скриени канали (ICC). Во системскиот подмодел DCC, испраќачот директно комуницира со примачите и во овој случај

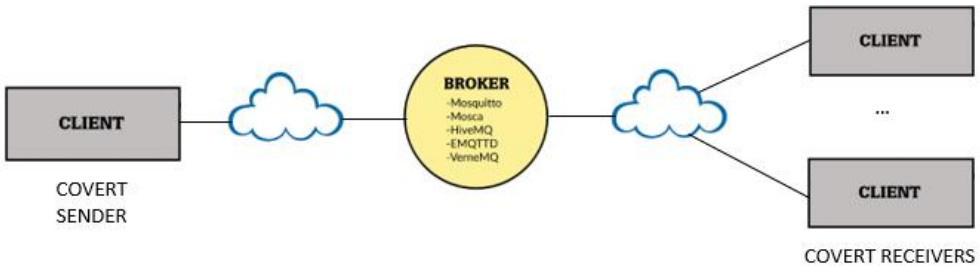
има две можности: брокерот може да биде испраќач (DCCa) или брокерот е единствениот примач (DCCb) (сл. 5.9).



Слика 5.9 Системски подмодел DCC: a) со брокерот како таен испраќач b) со брокерот како таен примач

Figure 5.9 System sub-model DCC: a) with the broker as a covert sender b) with the broker as a covert receiver

Во системскиот подмодел ICC, испраќачот индиректно комуницира со примачот преку брокерот како посреднички јазол (сл. 5.10). Ова значи дека нема директна интеракција помеѓу испраќачот и примачот и оттука тие не мора да бидат активни во исто време односно процесите за испраќање на скриени пораки и примање може да бидат одвоени.



Слика 5.10 Системски подмодел ICC (брокерот не е свесен за скриената размена на податоци)

Figure 5.10 System sub-model ICC (the broker is unaware for the secret exchange of data)

Во ова сценарио брокерот не е свесен за скриените податоци кои ги пренесува и тој само ги препраќа сите пораки кои ги прима.

Според прикажаните системски модели, предложените скриени канали може да бидат поделени во две групи и тоа: DCC (каде добро познати скриени канали се приспособени на MQTT) и ICC (каде се прикажани индиректни скриени канали кои се специфични за MQTT).

5.2.2 Директни скриени канали кај MQTT (DCC)

Неколку директни скриени канали може да се користат кај MQTT за комуникација помеѓу клиентите (објавувачи или претплатници) и брокерот и тие користат веќе познати техники кои се приспособени во контекст на MQTT. Во основа, овие скриени канали прават модификација на некои полиња во контролните пакети. Текст полињата се кодирани како UTF-8 стрингови, со големина која не надминува 65535 бајти и тие се следните:

1. *Application Message* – во товарот на PUBLISH пакетот. MQTT е „data-agnostic“, односно тој може да пренесува скоро се: слики, аудио, шифрирани податоци, текст во кој било коден формат и сл. Скриениот канал за ова поле исто така може да се користи за креирање на индиректна скриена комуникација помеѓу клиентите.
2. *Client identifier* – во товарот на CONNECT пакетот (серверите мора да дозволат големина до 23 UTF-8 кодирани бајти)
3. *Username u Password* – полиња во товарот на CONNECT-пакетот (секое до 65535 бајти).
4. *Keep Alive* – 16-битно поле во CONNECT-пакетот, кое го претставува максималниот временски интервал во секунди помеѓу два контролни пакети испратени од клиентот.
5. *Packet Identifier* – 16-битно поле во PUBLISH-пакетот кога QoS > 0 или во SUBSCRIBE, UNSUBSCRIBE, SUBACK, UNSUBSACK, PUBACK, PUBREC, PUBREL и PUBCOMP пакетите. За секој нов пакет клиентите и серверите може да назначат идентификатори на пакетите независно еден од друг. Секој пакет за потврда има ист идентификатор како и пакетот за кој се прави потврдата.
6. *Topic name* – поле во PUBLISH-пакетот (до 65.535 бајти).

7. *Topic Filters* – во товарот на SUBSCRIBE и UNSUBSCRIBE-пакетите (до 65.535 бајти).

Вгнездување на скриени битови и екстракција: За подмоделот DCCa, кога брокерот како таен испраќач сака да испрати скриена порака до примачите, тој треба да ја кодира пораката во одредено поле, зависно од избраниот скриен канал. Скриените битови може директно да се вгнездат во полиња како: Client Identifier, Password, Packet Identifier, Keep Alive, Username and Application Message.

За полињата Topic Name и Topic Filters, покрај директното вгнездување на битови, коешто не е толку скриено, подобар начин е да се користи еден од следните пристапи:

- Модулација на буквите (на пр., мали букви како бинарно 1, и големи букви како бинарно 0),
- Модулација на бројот на нивоа во името на темата (на пр., парен број на нивоа претставува бинарно 1, додека непарен претставува бинарно 0),
- Модулација на бројот на празни места во Topic Name (на пр., парен број на знаци претставува бинарно 1, додека непарен претставува бинарно 0),
- Користење на различни знаци за празни места во Topic Name (на пр. знакот „space“ – U+0020, претставува бинарно 1, додека знакот „non-breaking space“ – U+00A0, претставува бинарно 0.

Шема за криење на информации:

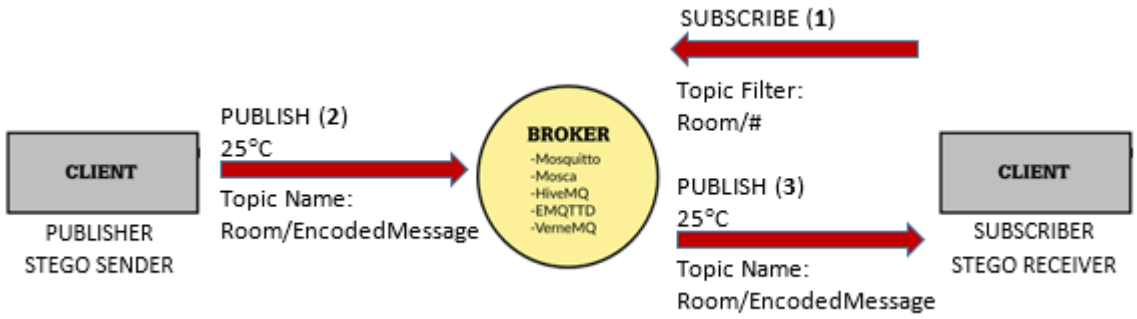
- Username и Topic Name - PS11. Value modulation
- Client Identifier, Password, Keep Alive и Packet Identifier - PS10. Random Value
- Application Message и Topic Filters - PS31. User-data value modulation & reserved/unused

5.2.3 Индиректни MQTT-специфични скриени канали (ICC)

5.2.3.1 Скриен канал со користење на полињата Topic Name и Topic Filter

Овој индиректен скриен канал (ICC.1) помеѓу објавувач и еден или повеќе претплатници е претставен на следниот начин:

- На почетокот, сите учесници во скриената комуникација мора да се договорат за некое познато прво ниво за полето Topic Name, на пример, Room. Потоа претплатниците се претплатуваат со користење на „Wildcard“ со повеќе нивоа (#) како второ ниво на темата, односно Room/# (сл. 5.11, чекор 1).



Слика 5.11 Индиректен скриен канал со користење на полињата Topic Name и Topic Filters

Figure 5.11 Indirect covert channel using the fields Topic Name and Topic Filters

Потоа, испраќачот испраќа скриена порака вгнездена во остатокот од полето Topic Name (сл. 5.11, чекор 2). На овој начин сите примачи ќе ги добијат сите ажурирања за темите кои започнуваат со Room (сл. 5.11, чекор 3), заедно со скриената порака која може да се екстрахира. Во овој случај, содржината на пораката не е важна, бидејќи пораката е скриена во името на темата. Вгнездувањето на скриени битови може да биде извршено врз основа на некој од пристапите во поглавје 5.2.2, во делот кај директните скриени канали DCC за Topic Name и Topic Filters.

Шема за криење на информации:

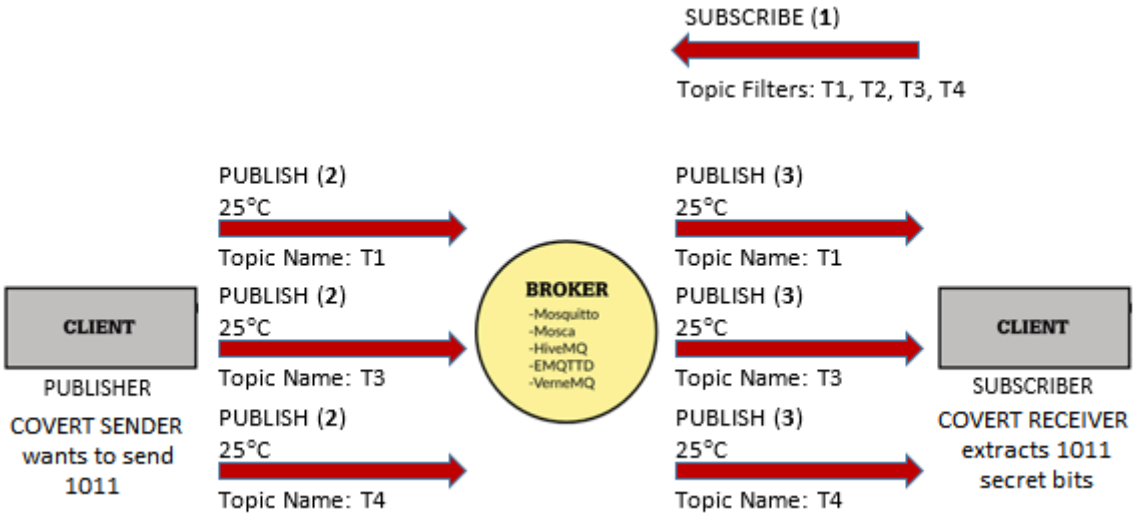
PS11. Value modulation

5.2.3.2 Скриен канал со користење на подредување на тема и присуство/отсуство на ажурирања

За овој метод за криење на податоци (ICC.2) тајниот испраќач и тајните примачи мора прво да се договорат да користат n имиња на теми $T_1,\ldots,T_n$ и да воспостават однапред дефинирана шема за вгнездување на скриени битови со подредување на темите.

Сите учесници се претплатуваат на претходно договорени теми (сл. 5.12, чекор 1), кои се нумерирани и поврзани со скриени битови. Потоа, доколку тајниот испраќач сака да испрати скриена порака, тој објавува пораки само на темите кои се мапирани со битови кои мора да бидат поставени на 1 во одредена порака. На пример, за да се пренесе 4-битна порака 1011, тајниот испраќач ќе испрати пораки на темите T_1, T_3 и T_4 (сл. 5.12, чекор 2), и сите претплатници,

односно соодветните тајни примачи, ќе ги добијат овие пораки (сл. 5.12, чекор 3).



Слика 5.12 Индиректен скриен канал со користење на подредување на теми и присуство/отсуство на ажурирања

Figure 5.12 Indirect covert channel using ordering of topics and updates presence/absence

Од присуството/отсуството на одредено ажурирање на договорените теми, и претходно договорената шема за подредување на темите, тајните примачи ќе можат да ја екстрахираат скриената порака.

Шема за криење на информации:

PT11. Value modulation

5.2.3.3 Скриен канал со користење на постојани сесии

Овој едно-битен еднонасочен скриен канал (ICC.3) ја користи можноста на клиентот да креира постојани сесии со серверот, каде што серверот постојано ја снима состојбата на клиентската сесија. Постојаните сесии се идентификувани од полето „Client Identifier“ во контролниот пакет CONNECT.

Прво, тајниот испраќач го праќа неговиот клиентски идентификатор (на пример senderID) до тајниот примач, кој подоцна и двајцата ќе го користат за креирање на врска со серверот во различни предодредени временски слотови. Ако тајниот испраќач сака да испрати скриен бит ‘1’ до тајниот примач, тогаш тој ќе креира постојана сесија со серверот, додека за испраќање на скриен бит ‘0’ тој ќе креира непостојана сесија со серверот. Во следниот временски слот, тајниот испраќач самиот ќе ја прекине врската. Тајниот примач ќе креира

постојана сесија со серверот со користење на истиот клиентски идентификатор (senderID). Серверот ќе го потврди добиениот CONNECT-пакет со CONNACK-пакет, во кој знаменцето SessionPresent ќе биде поставено на 1 (ако имало претходна постојана сесија) или на 0 (ако немало претходно постојана сесија). По добивањето, и екстрахирањето на скриениот бит од знаменцето Session Present, тајниот примач ја прекинува врската. Може да се забележи дека фреквентното воспоставување на врска со брокерот и прекинување не е неопходно да се разгледува како аномалија, бидејќи ова е типична практика доколку клиентот сака да заштеди енергија.

Шема за криење на информации:

PS11. Value modulation

5.2.3.4 Скриен канал со користење на присуство/отсуство на задржана порака (retained message)

Ако имаме складирано задржана порака за специфична тема, кога се воспоставува нова претплата која одговара на тоа име на темата, брокерот ќе му ја испрати пораката на претплатникот. Ако брокерот прими PUBLISH-контролен пакет со знаменце RETAIN поставено на 1 и не-нула-бајтен товар, брокерот ќе ја замени задржаната порака со ново добиената. Кога до брокерот се испраќа PUBLISH контролен пакет со знаменце RETAIN поставено на 1 и нула-бајтен товар, брокерот ќе ја избрише постоечката задржана порака и сите нови претплатници на таа тема нема да ја примат задржаната порака.

За едно-битен скриен канал (ICC.4), тајниот испраќач и тајниот примач прво мора да се договорат за големината на временските слотови, кои треба да бидат доволно големи така што испраќачот да може да ја испрати скриената порака и истовремено тајниот примач да има доволно време за да направи нова претплата и да ја добие задржаната порака, доколку ја има.

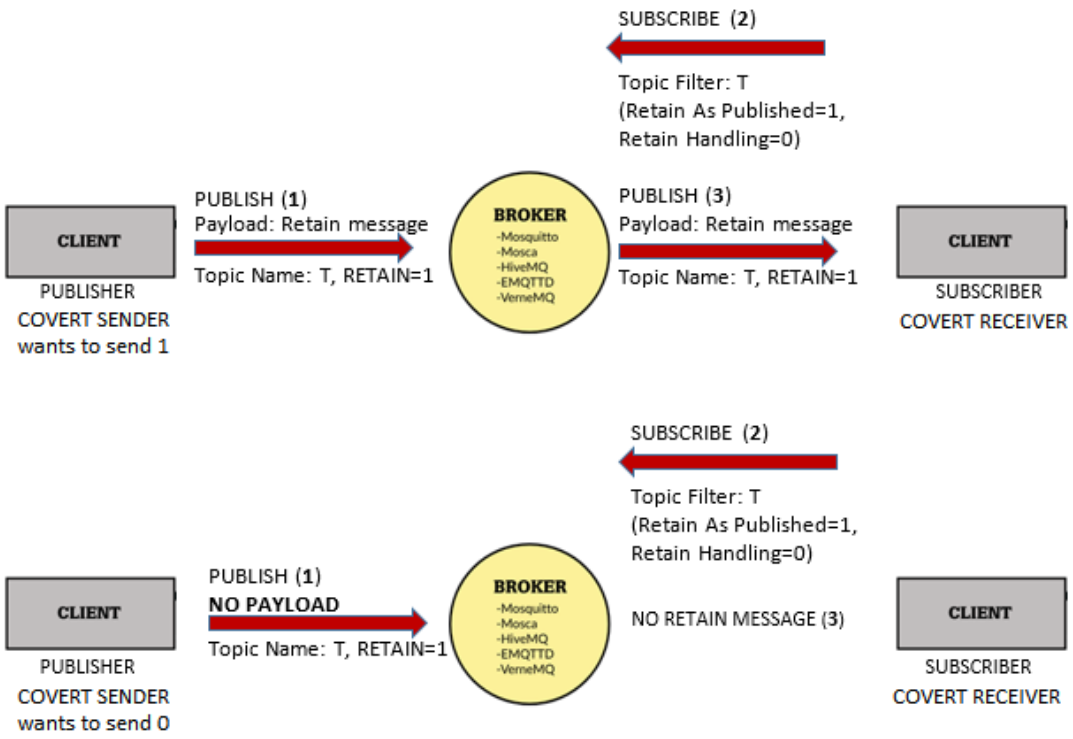
Испраќачот испраќа еден бит по слот, според следниот пристап (сл. 5.13, чекор 1):

- Присуството на задржана порака може да претставува бинарно 1. За оваа цел, тајниот испраќач ќе испрати PUBLISH-контролен пакет со знаменце RETAIN поставено на 1 и задржана порака како товар.
- Отсуството на задржана порака означува бинарно 0. За оваа цел, тајниот испраќач ќе пренесе PUBLISH-контролен пакет со знаменце RETAIN поставено на 1 и нула-бајтен товар.

Тајниот примач треба да креира нова претплата за специфична тема за временски слот со праќање на SUBSCRIBE-пакет со соодветен Topic Filter и опции за претплата: Retain As Published=1 и Retain Handling=0 (сл. 5.13, чекор 2). Испраќачот исто треба да користи непостојани (чисти) сесии, за да одбегне примање на складирани пораки. На страната на брокерот, оваа акција комплетно ја заменува постоечката претплата со нова. На овој начин тајниот примач може да воочи дали примил задржана порака или не, и ова да го толкува како скриен бит '1' или скриен бит '0'.

Шема за криење на информации:

- Хибриден канал кој ги комбинира шемите PS11. Value Modulation и PT2. Message Timing.



Слика 5.13 Индиректен скриен канал со користење на присуство/отсуство на задржана порака

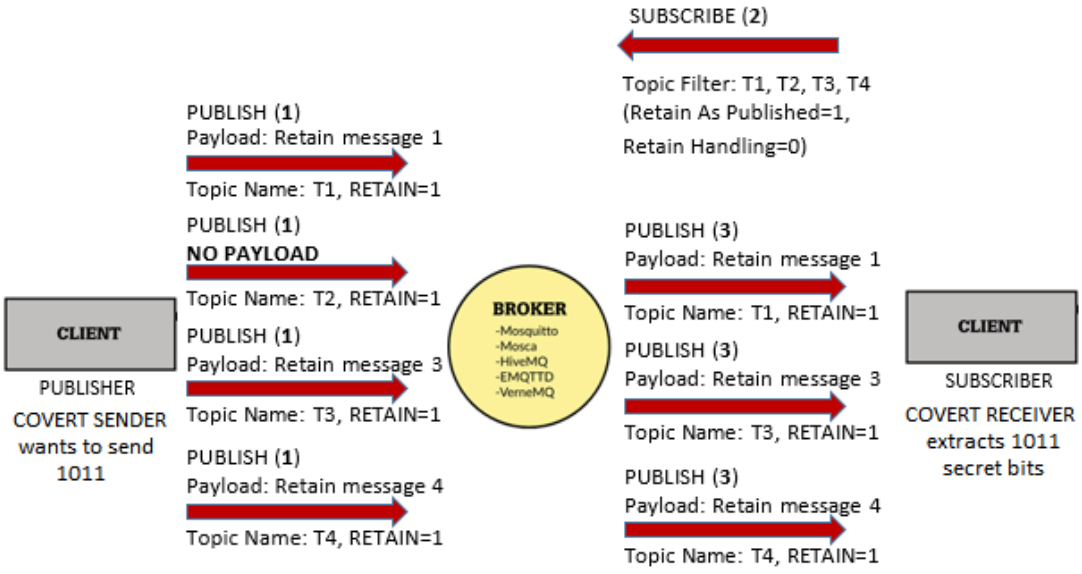
Figure 5.13 Indirect covert channel using presence/absence of retained message

5.2.3.5 Скриен канал со користење на подредување на теми и присуство/отсуство на задржани пораки

За овој скриен канал, тајниот испраќач и тајниот примач мора најпрвин да се договорат да користат n имиња на теми $T_1,\ldots,T_n$ и нивен специфичен

редослед. Потоа, индиректен скриен канал е креиран како што е дадено на слика 5.14. За праќање на n-долга скриена порака, тајниот испраќач врши модулирање на присуство/отсуство на задржани пораки (како бинарно ‘1’ или ‘0’) во сите овие теми, со праќање на PUBLISH контролен пакет со знаменце RETAIN поставено на 1 и задржана порака како товар/нула бајтен товар (како кај претходниот скриен канал).

На пример, за праќање на 4 битна порака 1011, напаѓачот ќе манипулира со присуство/отсуство на задржани пораки во сите избрани теми, со поставување и праќање на задржани пораки во темите T1, T3 и T4, и поставување на не-задржана порака само за тема T2. (сл. 5.14, чекор 1). По тоа тој ќе ја прекине врската.



Слика 5.14 Индиректен скриен канал со користење на подредување на теми и присуство/отсуство на задржани пораки

Figure 5.14 Indirect covert channel using ordering of topics and presence/absence of retained messages

Тајниот примач се претплатува на темите T_1, T_2, T_3 и T_4 (сл. 5.14, чекор 2) и прима задржани пораки само за T_1, T_3 и T_4 (сл. 5.8, чекор 3). На овој начин врз основа на претходно воспоставената шема за подредување на темите, примачот може да ја толкува скриената порака која во овој случај е „1011“.

Шема за криење на информации:

Ова е хибриден канал кој ги комбинира шемите PT11. Message Ordering, PS11. Value Modulation и PT2. Message Timing.

5.2.3.6 Скриен канал со користење на информации кои се специфични за брокерот

Постојат теми коишто прикажуваат специфични информации за брокерот. Овие теми започнуваат со $SYS/. Објавувачите не може да праќаат пораки на нив. Овие теми исто така може да се искористат за тајна комуникација.

Ако тајниот испраќач има можност да контролира конектирање и дисконектирање на одредени клиенти на серверот, ова може да се користи како скриен комуникациски канал (ICC.6) со некои тајни примачи поврзани на истиот сервер. Ова може да претставува пример сценарио каде тајниот испраќач може да контролира неколку сензори во зграда.

Тајниот испраќач и тајниот примач мора прво да ја истражат динамиката на конекциите/дисконекциите на различни клиенти. Динамиката на клиентските конекции може да се истражи со читање на бројот на моментално поврзани клиенти (NC) во одреден временски период, што може да се направи со претплата на следната тема:

$SYS/broker/clients/connected

Од друга страна, динамиката на дисконекции може да се истражи со читање на бројот на постојани (со Clean Session disabled) клиенти (ND) кои се регистрирани на брокерот но моментално се дисконектирани, во одреден временски период. Ова може да се направи со претплата на следната тема:

$SYS/broker/clients/disconnected

Двете тајни комуникациски страни на почеток треба да имаат претфаза во која ќе соберат информации за историските вредности на NC, и потоа тие ќе го започнат скриеното пренесување на пораки. За време на скриената комуникација тие ќе продолжат да добиваат информации за NC.

Тајниот испраќач и тајниот примач треба да се договорат за тројки од временски слотови (t_{i1}, t_{i2}, t_{i3}), i = 1, 2, …, во кои следните чекори треба да бидат извршени:

1. Во временскиот слот t_{i1}, CS и CR ќе ја добијат вредноста за NC и ќе го пресметаат просечниот број на поврзани клиенти (ANC) и стандардната девијација на поврзаните клиенти (SDC) за серверот, и следната вредност:

$$DT = \begin{cases} SDC - (NC - ANC), \ if \ NC > ANC \\ SDC - (ANC - NC), \ \text{инаку} \end{cases}$$

2. Праќање на скриен бит во временски слот t_{i2}:

- За праќање на бинарно 1, ако NC > ANC, тајниот испраќач ќе конектира DT клиенти, инаку тој ќе дисконектира DT клиенти.
- За праќање на бинарно 0, ако NC > ANC тајниот испраќач ќе дисконектира SDC-DT клиенти, инаку тој ќе конектира SDC-DT клиенти.

3. Примање на скриениот бит во временски слот t_{i3}. CR ќе ја добие вредноста за NC за временскиот слот t_{i3}, и доколку оваа вредност е околу просечниот број на поврзани клиенти, ANC, тој тоа ќе го интерпретира како бинарно 0. Инаку, ако тоа е околу минималната вредност, ANC – SDC или максималната вредност, ANC + SDC, тој тоа ќе го интерпретира како бинарно 1.

   Сличен канал може да се креира со користење на просечниот број на дисконектирани постојани клиенти (AND), стандардната девијација на дисконектирани постојани клиенти (SDD), и бројот на дисконектирани постојани клиенти (ND).

Шема за криење на информации:

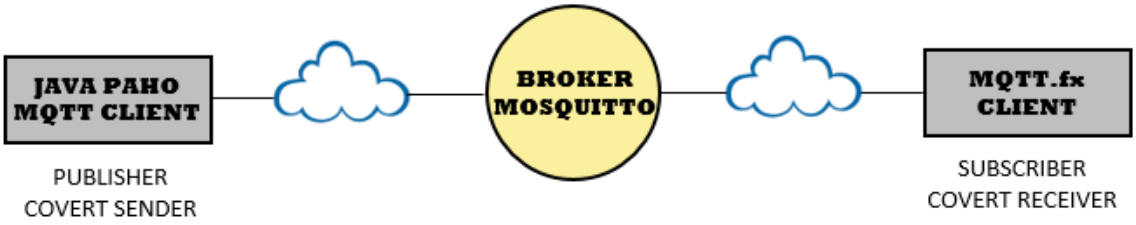
PS11. Value modulation

Според прикажаниот новитет, со овој канал извршено е проширување на генералната шема за криење на информации со што на потшемата PS11 е додадена нова гранка именувана како PS11c. Value Influencing Pattern. Тајниот испраќач може да биде и дистрибуиран со што овој скриен канал може да се разгледува и како дистрибуиран скриен канал (Mazurczyk et al., 2018).

5.2.4 Експериментална евалуација на ICC.2

Како доказ за изводливоста и ефективноста на предложените скриени канали во овој дел претставен ќе биде прототип на ICC.2-скриениот канал. Бидејќи не постоеше податочен сет којшто можеме да го искористиме за да утврдиме постоење на скриени канали, генериран беше реален MQTT-податочен сет од експериментално сценарио. Најпрвин креиравме EC2-инстанца на Amazon Web Services (AWS, 2021) за Mosquitto брокер v.1.4.8 (Mosquitto, 2021). Како објавувач (или таен испраќач) имплементиран беше MQTT-клиент со користење на Java Paho клиентската библиотека v.0.4.0 (Eclipse Foundation, 2021). Како претплатник (или таен примач) користена беше

клиентската програма MQTT.fx v.1.7.1 (MQTT.fx, 2021) (сл. 5.15). За снимање на сообраќајот користена беше програмата Wireshark v.2.6.8 (Wireshark, 2021).



Слика 5.15 Експериментално сценарио

Figure 5.15 Experimental scenario

Истражени беа три варијанти на ICC.2, каде различен број на теми беа искористени за криење на податоци:

- Варијанта 1: две теми (T1 и T2)
- Варијанта 2: три теми (T1, T2 и T3)
- Варијанта 3: четири теми (T1, T2, T3 и T4)

Прво, снимани беа 100 примероци од мрежниот сообраќај за секоја верзија со легитимен сообраќај (без имплементиран скриен канал). За легитимниот сообраќај имаме ситуација каде објавувачот испраќа пораки на 2, 3 или 4 теми, зависно од ICC.2 варијантата, со случајно избирање во секоја секунда за да објави / или да не објави за секоја тема одделно. Исто така, времето помеѓу секој настан за објавување е случајно (во милисекунди). Секој примерок од сообраќај е снимен во временски период од 10 мин.

По ова, креиран беше скриен канал за секоја од погорните варијанти од ICC.2 и испраќани беа тајни пораки во шифрирана форма (шифрирани со AES), повторно со снимање на сообраќајот во временски период од 10 минути. Скриениот канал беше имплементиран со користење на програмскиот јазик Java.

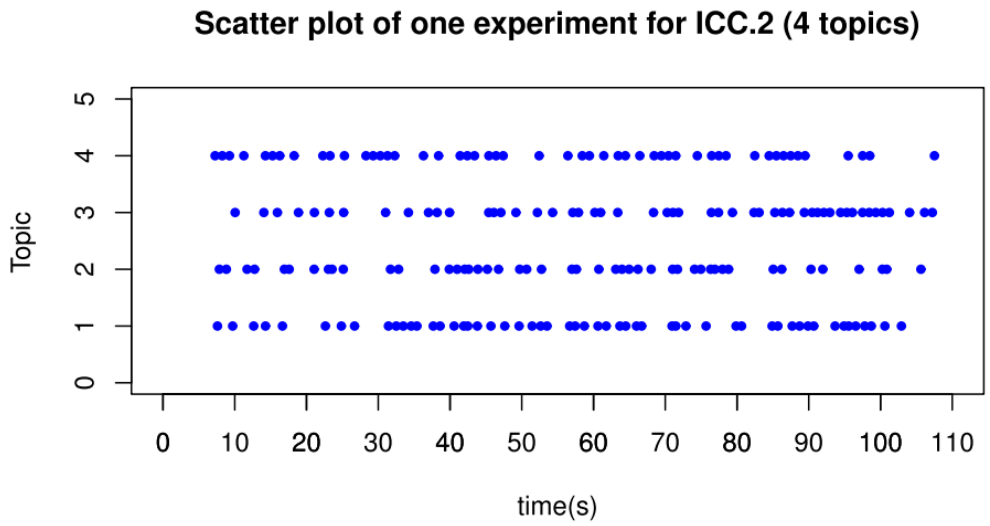
За ICC.2 со користење на 4 теми (варијанта 3) извршен беше друг експеримент. Испраќани беа пораки во чист ASCII-формат, за да се види како ова влијае на откривањето во споредба со шифрираниот сообраќај. Слично како и во претходните случаи снимени беа 100 примероци од сообраќајот во временски период од 10 минути.

Кодовите во Java за испраќање на легитимен сообраќај и испраќање на ASCII-пораки со користење на ICC.2-скриениот канал се прикажани во прилог 1.

Кодот за испраќање на AES-шифрирани пораки е многу сличен на ASCII каде единствената разлика е тоа што стрингот што се испраќа (str1) е шифриран со AES.

5.2.4.1 Пропусен опсег

Со цел да се воспостави остварлив пропусен опсег за ICC.2, како што претходно споменавме, експериментите беа извршени во 3 варијанти, каде што 2, 3 или 4 теми беа користени за криење на податоци. Бидејќи користиме временски интервал од 1 секунда за испраќање или неиспраќање во 2, 3 или 4 теми, експерименталниот пропусен опсег е 2 bps за 2 теми, 3 bps за 3 теми и 4 bps за 4 теми. Во секој случај, во секоја секунда испраќањето на бит 1 одговара на објавување на порака на одредена тема, додека испраќањето на бинарен бит 0 одговара на необјавување на одредена тема.



Слика 5.16 ICC.2 со 4 теми, испраќање на 400 скриени битови во 100 секунди

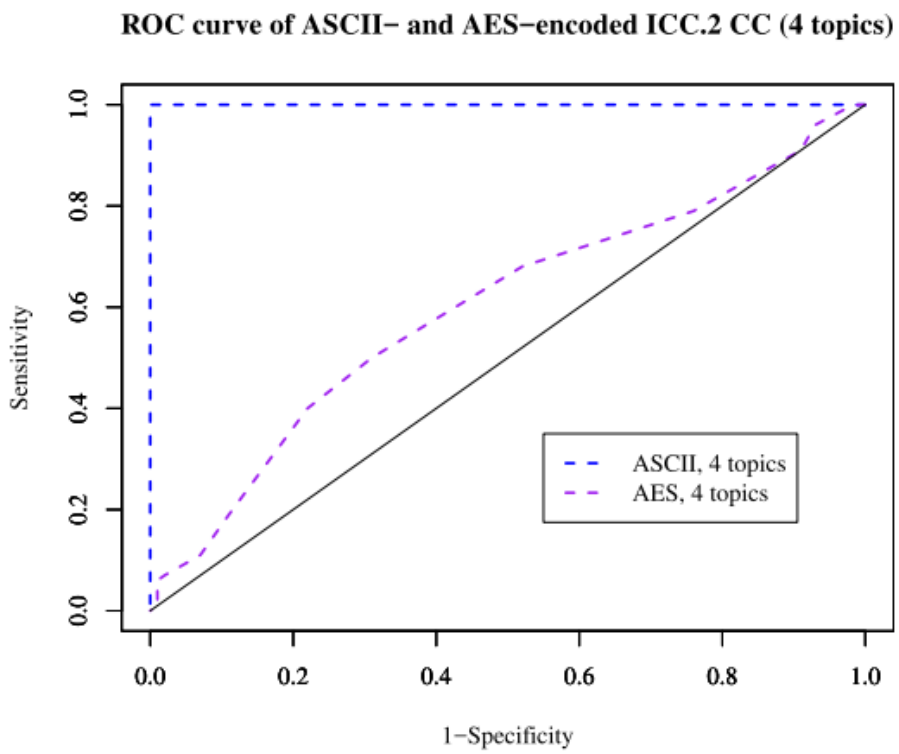
Figure 5.16 ICC.2 with 4 topics, sending 400 secret bits in 100 seconds

На слика 5.16 може да се видат првите 100 секунди од еден од 100-те извршени експерименти за индиректниот скриен канал ICC.2 со 4 теми. Во основа, оваа слика го претставува пренесувањето на 400 скриени битови во 100 секунди од седмата до сто и седмата секунда, и исто така ги прикажува точките во време кога се објавува во T1, T2, T3 или T4. Првите 7 секунди се користат за поврзување и синхронизација. Резултатите во однос на другите варијанти се аналогни на ова.

5.2.4.2 Неоткривање

За да се испита пристапот за детектирање, истражено беше неоткривањето на ICC.2-скриениот канал. Во првиот експериментален обид извршено беше детектирање на ICC.2 врз основа на податочно рударење со надгледување на редоследот на темите во мрежниот сообраќај. Генерирани беа 100 примероци за ICC.2 и беше направена споредба со појавувањето на темите во 100 примероци од легитимен сообраќај. Прво, снимени беа сите теми, во редослед според нивното појавување во даден стринг S. На пример доколку протокот содржи „Publish Message [T3]“, проследено со „Publish Message [T2]“, „Publish Message [T4]“ and „Publish Message [T1]“ тогаш резултирачкиот стринг ќе биде „3241“. Потоа извршено беше компресирање на стрингот и споредена беше неговата оригинална должина со компресираната должина што резултира со компресирачки резултат k. Направено беше и утврдување на оптималната должина на стринг S за тестирање на сет на параметри помеѓу 500 и 2.000 теми. Според ова добиен беше резултат дека 1.100 е добар избор.

Пристапот за детекција со компресирање е базиран на методот за детекција на скриен временски канал (Cabuk et al., 2009), но не може да се примени директно на експерименталниот скриен канал бидејќи редоследот на темите се користи како влез (наместо времето на пристигнување), генерирајќи различен стринг и потребно е да се одреди големината на прозорецот и на прагот k. Ова приспособување на противмерката за скриениот канал е познато како варијација на контрамерката (Wendzel et al., 2019). На крај направен беше обид за оптимална комбинација на големина на стринг и k-праг за да се изврши класифицирање помеѓу легитимен и сообраќај од скриен канал.



Слика 5.17 Откривање на ICC.2 варијанта 3 (со користење на 4 теми) со ASCII и AES-шифрирана содржина

Figure 5.17 Detection of ICC.2 variant 3 (using 4 topics) with ASCII and AES-encrypted content

Кога експерименталниот канал кодира скриени податоци во тривијален ASCII-формат со користење на 4 теми (присуството на 4 теми значи 4 бита од ASCII-карактерите), детекцијата обезбедува одлични резултати (100 % F-Score). Во случај на AES-шифриран сообраќај којшто исто така е кодиран со 4 теми, откривањето се намалува значително (66,9 % F-Score). Како за споредба на слика 5.17 може да се видат ROC кривите на двете варијанти за AUC (Area Under Curve) каде за ASCII-каналот вредноста е 0.995, додека за AES каналот е 0.5996.

5.2.4.3 Робусност

Со цел да се испита колку е робусен ICC.2-скриениот канал, истражено беше како тој се однесува под влијание на зголемени доцнења и загуба на пакети.

Табела 5.5 го прикажува бројот на превртени битови (бројот на битови примени со грешка) на страната на примачот. Мерењето е направено на проток од 120 скриени бита, со 3 различни мрежни доцнења, и со 3 експерименти.

Табела 5.5 Број на превртени битови во 120 пренесени скриени битови што се должи на прекини во мрежата

Table 5.5 Number of flipped bits in 120 transferred bits due to network delays

	10 ms		50ms		100 ms	
	0 во 1	**1 во 0**	**0 во 1**	**1 во 0**	**0 во 1**	**1 во 0**
Експеримент 1	1	1	1	2	4	7
Експеримент 2	0	1	1	2	4	6
Експеримент 3	1	5	2	6	3	7
Просечно	0.67	2.33	1.33	3.33	3.67	6.67

Одделно се прикажани бројот на превртени 1 и превртени 0. Грешки при трансмисија настануваат бидејќи некои од задоцнетите пораки се испратени во следната секунда, наместо во моменталната секунда. Резултатите покажуваат дека за 10 ms доцнење, 2,5 % од битовите се примени со доцнење, за 50ms, 3,9 % се примени со грешка, и за 100ms, 8,6 % од битовите се примени со грешка.

5.2.5 Противмерки за предложените скриени канали кај MQTT-верзија 3.1.1

Можна противмерка за скриениот канал кој го користи полето „Application Message“ е да се гледа содржината на ова поле која е невообичаена за специфичен IoT-уред. Често, „Application Message“ има некоја структура којашто зависи од типот на уред којшто ја објавува пораката. На пример, сензорот за температура праќа само податоци за температурата, сензорот за влажност праќа податоци за влажноста, и сл.

Еден можен метод за детекција на скриените канали кои ги користат полињата „Client Identifier“, „User Name“ и „Password“ во CONNECT-контролниот пакет е да се гледа за различни вредности за исти IP-адреси или за исти MAC-адреси.

Скриениот канал кој го користи полето „Keep Alive“ може да се детектира со надгледување на вредностите во последователни CONNECT-контролни

пакети за исти IP-адреси и за MAC-адреси. Појавувањето на неколку различни вредности би можело да биде индикатор за присуство на скриен канал.

Скриениот канал кој го користи полето „Packet Identifier“ најтешко може да се детектира бидејќи се очекува секој нов пакет да има нова случајна вредност за ова поле. Една можност да се детектира тајна комуникација е да се гледаат пакетите за потврда кои треба да имаат иста вредност за ова поле како вредноста во пакетот за кој се прави потврдата (PUBACK: Packet Identifier треба да биде ист како во PUBLISH-пакетот кој го потврдува).

Една можна противмерка за детекција на директните скриени канали кои ги користат полињата Topic Name и Topic Filter, како и за индиректниот скриен канал ICC.1, е да се врши испитување на имињата на темите во објавените пораки (или Topic Fillters во SUBSCRIBE и UNSUBSCRIBE-пакетите). Притоа, потребно е надгледување на невообичаено однесување како објавување во многу различни теми, во теми со долги имиња или имиња со случајни знаци, теми со чудни имиња во кои се користат големи и мали букви или различен број на празни места.

Подредувањето на темите кое е искористено од скриените канали ICC.2 и ICC.5 може да се детектира со споредба на редоследот на темите во даден проток со легитимниот редослед на теми (на пр., со испитување на ентропијата).

Можна противмерка за детекција на скриениот канал ICC.3 е да се чува запис на (Client Identifier, MAC address) парови, и да се гледа за постоење на најмалку два пара со ист „Client Identifier“ и различни MAC-адреси. Како алтернатива на ова може да се врши набљудување на датотеката со логови, за постоење на неколку CONNECT-контролни пакети со ист „Client Identifier“ и различни IP или MAC-адреси, во релативно кратки временски интервали.

ICC.4 и ICC.5 може да бидат детектирани со броење на SUBSCRIBE-контролните пакети за клиент, и бројот на PUBLISH-контролните пакети без товар за клиент. Доколку администраторот на брокерот има некоја статистика за овие броеви во легитимниот сообраќај, овие скриени канали може да се детектираат доколку постои голема разлика споредено со вредностите од легитимниот сообраќај.

Противмерката за детекција на скриениот канал ICC.6 може да се базира на броење на CONNECT и DISCONNECT-контролните пакети за даден клиент на страната на брокерот. Доколку постои голема разлика во однос на легитимниот сообраќај, ова може да биде показател за постоење на тајна комуникација.

6. СКРИЕНИ КАНАЛИ КАЈ ПРОТОКОЛОТ MQTT-ВЕРЗИЈА 5.0

Во 2019 година развиена беше нова верзија на MQTT (MQTT 5.0) и истата година стана OASIS-стандард (OASIS, 2019). Во основа верзиите 3.1.1 и 5.0 не се компатибилни помеѓу себе бидејќи новата верзија има прилично различен формат на пораки и повеќе функционалности во споредба со верзијата 3.1.1. Некои компании како HiveMQ обезбедуваат слој за компатибилност за да се овозможи коегзистирање на двете верзии (HiveMQ, 2018).

Иако моментално најкористена верзија на MQTT-протоколот е верзијата 3.1.1, во иднина се очекува дека најголем дел од уредите ќе ја користат новата верзија MQTT 5.0. Во претходното поглавје претставена беше анализа за скриени канали кај MQTT 3.1.1. За верзијата 5.0 сè уште немаше истражувања за скриени канали. Тоа е причината поради која во овој дел претставуваме една сеопфатна анализа на мрежни скриени канали за оваа верзија на протоколот.

Клучни придонеси за научната сфера кои се претставени во ова поглавје се:

- Предложени се 18 директни и 5 индиректни мрежни скриени канали
- Обезбеден е доказ за концептот на еден индиректен скриен канал

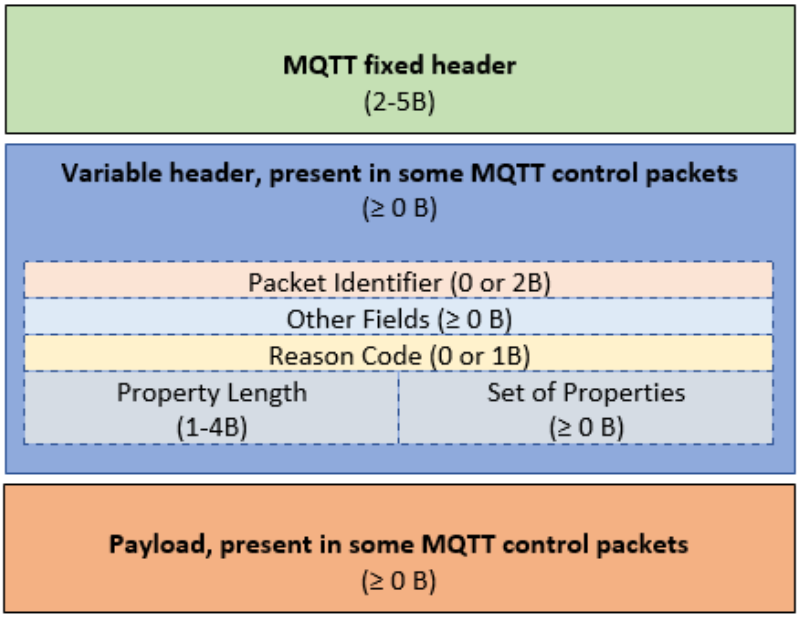
6.1 Основи на MQTT-верзија 5.0

Основните работи кои се однесуваа за MQTT-протоколот беа објаснети во претходното поглавје. Како што беше споменато верзијата 5.0 на MQTT-протоколот се разликува од претходната верзија во форматот на пораките и во неа исто така се додадени нови функционалности.

MQTT-верзија 5.0 користи 15 контролни пакети кои се нумерирани од 1 до 15. Таа има еден пакет повеќе во однос на MQTT 3.1.1. Секој контролен пакет се состои од фиксно заглавие и варијабилно заглавие, со големина на пакет помеѓу 2B (на пр. PINGREQ пакетот) и 256 MB (сл. 6.1).

Варијабилното заглавие има променлива структура кај различни контролни пакети, кое може да се состои од 2B идентификатор на пакет (Packet Identifier) или други полиња, единичен 1B Reason Code, и дел за особини (Properties) кој се состои од задолжително поле именувано како Property Length и опционален сет од особини (Set of Properties). Ако сетот на особини е празен,

Property Length мора да биде поставено на 0. Дополнително, Will Properties полето во товарот на CONNECT-контролниот пакет може да има сет на особини и SUBACK/UNSUBACK-пакети кои може да содржат листа на Reason кодови во товарот.



Слика 6.1 Структура на MQTT v.5.0 контролен пакет

Figure 6.1 Structure of MQTT v.5.0 control packet

Една од најважните разлики во форматот на контролните пакети помеѓу верзиите 3.1.1 и 5.0 на MQTT-протоколот е тоа што 13 контролни пакети во MQTT 5.0 (без PINGREQ и PINGRESP) содржат Properties дел во варијабилното заглавје. Седум од нив содржат и дел за Reason Code којшто не постои кај верзија 3.1.1.

6.2 Нови особини кај MQTT-верзија 5.0

MQTT 5.0 наследува најголем дел од особините од претходната верзија на овој протокол (MQTT 3.1.1). Оваа верзија е најголемото ажурирање во однос на воведувањето нови особини за MQTT-протоколот. Брокерите како што е случајот со HiveMQ, веќе имаа имплементирано особини како што се споделени претплати (Shared Subscriptions) и Time to Live за пораки и клиентски сесии во MQTT 3.1.1. Подоцна со развивањето на верзијата 5.0 овие особини биле вклучени како дел од официјалниот стандард. Основната цел на новата спецификација е подобрување на скалабилноста кај големите системи.

Во 2017 година, во едно истражување (HiveMQ, 2017) на тимот на HiveMQ брокерот, тие си поставиле за цел да достигнат 10 милиони истовремено поврзани MQTT-клиенти и притоа направиле тестови за нивоата на квалитет на сервис (QoS) 0 и 1. Во тест-сценариото имало 10 милиони уреди кои биле претплатени на 1.000 теми. Во две различни сценарија, извршено било праќање на пораки со QoS 0 (4 пораки/секунда) и QoS 1 (2 пораки/секунда) со 40 MQTT објавувачи кои биле еднакво дистрибуирани на одредени теми за да генерираат 1,6 милиони пораки/секунда и 800 илјади пораки/секунда, соодветно. Притоа, успешно биле поврзани 10 милиони клиенти и со QoS 0 се обезбедил излезен пропусен опсег од повеќе од 1,7 милиони пораки во секунда. Како за споредба, излезниот пропусен опсег на HiveMQ е 2,5 пати поголем од вкупниот број на пораки кои WhatsApp апликацијата ги процесирала во еден ден во 2014 година. Според податоците, HiveMQ може да процесира 6,26 милијарди излезни пораки на час и околу 150 милијарди пораки во еден ден. Во вториот тест за QoS 1 забележан бил пропусен опсег од 780 илјади пораки во секунда и излезни и дојдовни во исто време. Вкупниот број на излезни пораки во еден час бил 2,49 милијарди, додека дојдовните пораки биле 2,81 милијарди. Сето ова ја потврдува важноста од воведување на нови особини со кои во иднина ќе се обезбеди подобро управување со уредите кои го користат MQTT-протоколот, подобра скалабилност и одржување.

Следните особини се новододадени особини во MQTT-верзија 5.0:

- *Подобрување на управувањето со сесиите*: Со воведување на опционални полиња за интервали за истекување на сесија и на порака (Session Expiry и Message Expiry). Во MQTT 5.0, постојаните сесии може да истечат и нивната состојба може да биде избришана од серверската страна. Знаменцето „CleanSession“ во CONNECT-контролниот пакет е заменето со „Clean Start“, кое укажува на тоа дали сесијата е чиста или постојана. 32-битно поле именувано како „Session Expiry Interval“ покажува колку долго (во секунди) да се складира постојаната сесија откако клиентот ќе се дисконектира. Интервалот за истекување на пораки (Message Expiry Interval) се применува и за онлајн пораките и за оние кои се наоѓаат во ред за чекање (PUBLISH) пораки. Ова значи дека некој клиент со постојана сесија којшто е дисконектиран, може да не ги добие

овие пораки кога повторно ќе се конектира, токму поради интервалот за истекување на пораки.

- *Подобрено известување за грешки*: Со опционален Reason Code и Reason String за сите потврди (ACKs). Основниот бенефит од оваа особина е тоа што DISCONNECT-пораките им овозможуваат на клиентите да одредат зошто тие се дисконектирани од брокерот. За претходната верзија на протоколот имало доста забелешки од корисниците во однос на недостатокот на транспарентност. За истражување на причината за дисконектирање, истражување зошто пораките не ја достигнуваат целта, или осигурување на постојаноста на развиените MQTT клиенти помеѓу повеќе тимови, MQTT-верзија 3 клиентите морале да ги прошират особините на протоколот и да користат технологии како што е HiveMQ Extension SDK (GitHub, 2019). За да се надминат овие предизвици и да се обезбеди соодветна стандардизација, оваа особина е воведена во MQTT 5.0.
- *Нови механизми за проширување*: Со воведување на „Payload Format Indicator“ за бинарен или UTF-8 кодиран стринг, опционален MIME-стил „Content Type“ за товарот и со неограничен број на кориснички особини (user properties), вообичаено дефинирани од клиентските апликации. Корисничките особини овозможуваат додавање на податоци (до 250 MB) во заглавието на контролните пакети, и овие податоци може да се користат за создавање на сопствени протоколи преку MQTT.
- *Споделени претплати*: MQTT-корисниците барале воведување на ваква карактеристика многу порано. HiveMQ ја имал воведено оваа карактеристика уште пред да биде имплементирана во MQTT 5.0 (HiveMQ, 2016). Сега секој брокер којшто сака целосно да биде во согласност со MQTT 5 спецификацијата треба да поддржува споделени претплати. Кај стандардните MQTT претплати, секој претплатен клиент прима копија од секоја порака која е испратена на дадена тема. Со воведувањето на оваа особина, сите клиенти кои споделуваат иста претплата во иста група на претплата примаат пораки на наизменичен начин. Клиентите може да споделат иста претплата на брокерот и брокерот ќе ја достави секоја порака само до еден избран клиент (претплатник) за даден

идентификатор на група. На овој начин, брокерот може да ги дистрибуира потоците од пораки до повеќе претплатници што е добро за зголемување и намалување на клиентите. Овој механизам уште се нарекува и балансирање на оптоварувањето на клиентот (client load balancing), бидејќи товарот од пораки на дадена тема е дистрибуиран помеѓу сите претплатници. MQTT-клиентите може да се претплатуваат на споделени претплати со стандардните MQTT-механизми. Сите познати MQTT-клиенти може да се користат без притоа да се прават промени на клиентска страна. Споделените претплати ја користат следната структура на тема:

$share/GROUPID/TOPIC

Темата се состои од 3 дела и тоа:

- Статички идентификатор за споделена претплата ($share)
- Идентификатор на група (GROUPID)
- Тема за претплата (може да вклучува и „wildcards“)

Пример за споделена претплата може да биде следниот:

$share/group1/office/groundfloor/+/temperature

- *Ограничувања на клиентот*: Клиентите кои имаат ограничени ресурси може да ја специфицираат максималната големина на пакет кој може да го примат, максималниот број на пораки со квалитет на сервис QoS 1 и QoS 2 кои може да бидат испратени, и „Will Delay“ со што се специфицира временско одложување на праќање „will“ пораки. Овие пораки се испраќаат до претплатниците кога клиентите се дисконектираат кога се јавува некоја I/O грешка, се случува грешка кај серверот или кога клиентот губи контакт за време на даден период. На овој начин клиентите може да го контролираат нивниот товар и да одбегнат преоптоварување. Овие пораки се поставуваат кога клиентот се поврзува.

- *Ограничувања на серверот*: Серверот може да дефинира множество на особини коишто не ги поддржува или да обезбеди соодветни ограничувања. Пример за ограничувања може да бидат максимален број

на прекари (aliases) на теми, предефиниран интервал за истекување на сесија и сл.

- *Барање-одговор мод на операција*: MQTT е базиран на објави-претплати модел и притоа можни се еден-на-многу врски. Со воведувањето на особината барање-одговор во најновата верзија на протоколот се овозможува еден-на-еден комуникација како кај HTTP-протоколот. Во повеќето случаи посебно карактеристични за IoT-комуникација, барањето предизвикува специфична акција за приемникот додека одговорот содржи резултат од таа акција.

  Барање-одговор модот на операција е овозможен со користење на:

  - Response Topic: Оваа тема е опционален UTF-8 стринг кој е достапен во кој било PUBLISH или CONNECT-пакет. Во CONNECT-пакетот, темата за одговор (Response Topic) се однесува на „will“ пораките. Доколку „Response Topic“ содржи вредност, испраќачот автоматски го идентификува соодветното објавување како барање. Ова поле во основа ја претставува темата на која се очекува добивање на пораки од приемниците како одговор на барањето. Добра практика за испраќачот на оригиналната порака како барање е да се претплати на темата за одговор пред да ја испрати пораката.
  - Correlation Data: Ова се податоци кои следуваат по темата за одговор. Испраќачот на пораките ги користи овие податоци за идентификување на специфичното барање на кое се однесува одговорот којшто ќе биде примен подоцна. Со користење на податоци за корелација, оригиналниот испраќач може да управува со асинхрони одговори кои може да бидат испратени од повеќе примачи. Овие податоци не се важни за MQTT брокерот но тие служат како средство за идентификување на врската помеѓу испраќачот и примачот.
  - Response information: За обезбедување на транспарентна имплементација и подобра стандардизација, MQTT 5 спецификацијата ја воведува особината за информација за одговор (Response information). Клиентот може да бара информација за одговор од брокерот со поставување на „boolean“ поле во CONNECT-

пакетот. Кога „request response information“ е поставено на вредност „true“, брокерот може да испрати опционално UTF-8 стринг поле (response information) во CONNACK-пакетот за да проследи информации за темите за одговор (response topics) кои се очекуваат да бидат користени. Со оваа особина корисниците може да дефинираат дрво од теми на брокерот, кои подоцна може да бидат користени од сите клиенти кои индицираат дека сакаат да го користат овој механизам при воспоставување на врска.

- *Подобрена автентикација*: Во новата верзија е воведена предизвик-одговор автентикација (Challenge-Response Authentication). Во споредба со традиционалниот пристап за автентикација којшто е базиран на акредитиви, со овој пристап серверот го автентицира клиентот со презентирање на предизвик на којшто клиентот мора да одговори со валиден одговор. Подобрената автентикација се базира на три типови на MQTT-контролни пакети и тоа: CONNECT, CONNACK и AUTH. Првите два типа веќе се присутни во MQTT 3.1.1 додека AUTH е нов MQTT 5.0 пакет. CONNECT-пакетите се карактеристични за клиентите додека CONNACK за серверите (брокерите). Овие пакети се користат само еднаш при автентикација. AUTH-пакетите може да се користат повеќе пати од серверот или клиентот. Две особини се основата на текот за автентикација. Тоа се метод за автентикација (Authentication Method) кој е идентификуван како бајт 21 и податоци за автентикација (Authentication Data) идентификувани како бајт 22. Овие особини се поставуваат во секоја порака која учествува во подобрениот автентикациски проток. Методот за автентикација се користи за избирање и за опишување на начинот за автентикација за кој се договориле клиентот и серверот. Ова се прави со метод стрингови на пример: SCAM-SHA-1 за SCAM, со SHA-1 или GS2-KRB5 за Kerberos. Во основа со овој метод се обезбедува значење на податоците кои се разменуваат за време на подобрената автентикација и истиот не треба да се менува. Податоците за автентикација се користат за пренесување на шифрирани тајни или чекори на протоколот во повеќе итерации. Притоа, содржината е зависна од специфичниот механизам кој се користи при автентикација и тој е специфичен за апликацијата.

- *Алијас (прекар) на тема*: Оваа особина е посебно корисна кога имаме поврзано голем број на уреди и кога имаме често испраќање на мали пораки. Со алијасите на темите се овозможува нивна замена со цел број. Испраќачот може да испрати „Topic Alias“ вредност во PUBLISH-порака, следејќи го името на темата. Приемникот тогаш ја процесира пораката како и секоја друга PUBLISH-порака и прави мапирање помеѓу алијасот на темата и името на темата. Секоја следна PUBLISH-порака за истото име на тема може да се испрати со празно име на тема, само со користење на дефинираниот алијас. Дополнително, може да постојат ограничувања за бројот на алијаси за теми. За ова се користи особината „Topic Alias Maximum“ која може да се постави за време на воспоставување на врска. Клиентот може да ја постави во CONNECT-пакет додека брокерот може да ја постави во CONNACK-пакет. Клиентот мора да користи вредности за „Topic Alias“ помеѓу 1 и „Topic Alias Maximum“ вредноста која е поставена во CONNACK-пакетот испратен од брокерот. На истиот начин брокерот може да користи само вредности помеѓу 1 и максимумот којшто е испратен преку CONNECT-пакет од страна на клиентот. Доколку не е поставена вредност за „Topic Alias Maximum“, се претпоставува вредност 0 со што користењето на алијаси на теми не е дозволено. Оваа особина е доста корисна во случаи кога имаме големи имиња на теми како на пример:

  Europe/Germany/north/area/munich/office/office13/id478521/light/id345/consumption

  Алијасите се најмногу корисни кога испраќаме пораки со кратка содржина во реално време со големи имиња на теми. Со нив може значително да се намали мрежниот сообраќај.

- *Идентификатор на претплата* (Subscription ID): За SUBSCRIBE-контролниот пакет може да се специфицира нумерички претплатнички идентификатор и тој може да биде вратен во секоја порака кога е доставен од брокерот. Ова му овозможува на клиентот да одреди која претплата или претплати предизвикуваат пораката да биде доставена. Дополнително, обезбедени се нови опции за претплата.

- *Контрола на проток:* MQTT-клиентите често имаат различни карактеристики во однос на серверите како што се брзината на процесирање, можностите за складирање и сл. Со ова клиентите имаат различни нивоа на толерантност за управување со пораките во лет кои се испраќаат (in-flight messages). Овие пораки се PUBLISH-пораки со квалитет на сервис 1 или 2 кои сè уште не се потврдени. Еден IoT-уред може да биде поврзан на повеќе брокери коишто имаат различни ограничувања за бројот на пораки во лет кои се испраќаат од даден MQTT-клиент. За да се разрешат овие различни состојби помеѓу MQTT-клиентите и брокерите, во MQTT 5 воведена е особината за контрола на проток. Клиентите и брокерите ги договараат ограничувањата за пораки во лет во време на воспоставување на сесија. Клиентот може да постави опционална особина наречена „Receive Maximum“ во CONNECT-контролниот пакет (Client Receive Maximum). Оваа вредност му го укажува на брокерот максималниот број на непотврдени (unacknowledged) PUBLISH-пораки кои клиентот може да ги прими. Брокерот одговара со опционална вредност за „Receive Maximum“ во CONNACK пакет (Server Receive Maximum). Оваа вредност му го покажува на клиентот максималниот број на непотврдени PUBLISH пораки коишто може да бидат примени од брокерот. Доколку вредноста недостасува, во тој случај стандардна вредност е 65535. Ако даден MQTT 5 клиент праќа повеќе непотврдени пораки отколку вредноста за „Server Receive Maximum“), во тој случај брокерот испраќа DISCONNECT-пакет со код за причина (Reason Code) 0x93 (Receive Maximum exceeded). Брокерите и клиентите може да изберат да праќаат помалку пораки во лет отколку што дозволува соодветниот „Receive Maximum“.

6.3 Скриени канали кај MQTT-верзија 5.0

6.3.1 Системски модел

За скриените канали кај MQTT 5.0 ќе биде користен истиот системски модел којшто беше предложен за MQTT 3.1.1 скриените канали во поглавје 5.2.1. Моделот се состои од еден таен испраќач (CS) и еден или повеќе тајни примачи CRs). Овој модел има два подмодели и тоа: директни скриени канали (DCC) каде брокерот може да биде таен испраќач (DCCa) или единствениот таен примач

(DCCb), и индиректни скриени канали (ICC) каде брокерот ја игра улогата на посредник во скриената комуникација помеѓу тајните испраќачи и примачи. За еден од скриените канали овде е воведен и нов системски модел каде тајниот испраќач влијае на мрежниот сообраќај со правење на повторни конекции. За утврдување на скриената порака тајниот примач врши анализирање на мрежниот сообраќај.

6.3.2 Применливост на постоечки скриени канали од MQTT 3.1.1 кај MQTT 5.0

Како што беше споменато претходно, во овој дел ќе бидат опфатени скриените канали кои може да се креираат со користење на новите особини кои се додадени во MQTT 5.0. Вреди овде да се напомене дека најголем дел од скриените канали кои беа претставени во поглавјето 5, и се однесуваа на верзија 3.1.1 на MQTT може да се применат и кај верзија 5.0. За некои од нив потребна е соодветна модификација сѐ со цел да се овозможи правилно функционирање. Тоа се следните скриени канали:

- 16-битно поле за клиентски идентификатор – Во верзијата 5.0 серверот дополнително може да постави назначен клиентски идентификатор (Assigned Client Identifier) во CONNECT контролниот пакет па овој скриен канал може да се разгледува како DCCa или DCCb во новата верзија.
- 16-битно поле „Keep Alive“ – Во верзијата 5.0 серверот може да постави „Server Keep Alive“ во CONNACK-пакет со што овој скриен канал исто така може да се разгледува како DCCa или DCCb.
- Скриен канал со постојани сесии (ICC.3) – Поставувањето на полето „CleanStart“ на вредност 1 и „Session Expiry Interval“ на вредност 0 во CONNECT-контролниот пакет е исто со поставување на полето „Clean Session“ во MQTT 3.1.1 CONNECT-контролниот пакет на вредност 1. Со ова се креира непостојана сесија со серверот. Доколку „Clean Start“ е поставено на вредност 0 а „Session Expiry Interval“ на 0xFFFFFFFF, ова е исто со поставување на Clean Session на вредност 0 во CONNECT-контролниот пакет кај MQTT 3.1.1. На ваков начин се креира постојана сесија со серверот. Друга разлика е тоа што кога серверот испраќа CONNACK-контролен пакет до тајниот примач, тој исто така мора да го постави следниот код за причина (Reason Code) 0x00 што означува успех.

6.3.3 Нови директни скриени канали кај MQTT-верзија 5.0

6.3.3.1 Скриени канали со кодирање на скриени податоци во полиња

Во MQTT 5.0 постојат повеќе нови полиња во променливото заглавие или во товарот кои може да се искористат за креирање на директни скриени канали. Притоа, новите полиња во променливото заглавие ќе ги именуваме како група D1 додека новите полиња во товарот ќе ги именуваме како група D2. Во табела 6.1 може да се погледнат полињата од група D1.

Табела 6.1 Нови полиња во променливото заглавие кај MQTT-верзија 5.0

Table 6.1 New fields in the variable header in MQTT version 5.0

Code	Поле	Контролен пакет	Големина	Дополнителни карактеристики
D1.1	Message Expiry Interval	PUBLISH или CONNECT (Will properties field)	32 бита	Вредноста на ова особина се модификува кога брокерот испраќа порака и таа го содржи и времето за кое пораката чека во брокерот
D1.2	Response Topic, Correlation Data, Content type	PUBLISH или CONNECT (Will properties)	Секое до 65.535 бајти	/
D1.3	Subscription Identifier	PUBLISH и SUBSCRIBE	28 бита	Може да има вредност од 1 до 268.435.455
D1.4	Topic Alias	PUBLISH	16 бита	/
D1.5	Session Expiry Interval	CONNECT, CONNACK и DISCONNECT	32 бита	/
D1.6	Receive Maximum, Topic Alias Maximum	CONNECT и CONNACK	16 бита	/
D1.7	Maximum Packet Size	CONNECT и CONNACK	32 бита	/
D1.8	Server Keep Alive	CONNACK	16 бита	/
D1.9	Assigned Client Identifier	CONNACK	До 65.535 бајти	/
D1.10	Authentication Data	CONNECT, CONNACK и AUTH	До 65.535 бајти	/

D1.11	Response Information	CONNACK	До 65.535 бајти	/
D1.12	User Property	Во 13 контролни пакети	Секој пар од вредности до 2 * 65.535 бајти	Само PINGREQ и PINGRESP контролните пакети не може да го содржат полето „User Property”

Каналите од групата D1 означени со 2 (Response Topic, Correlation Data, Content Type) и 12 (User Property) може да се користат исто така и како индиректни скриени канали.

Полињата од група D2, кои се наоѓаат во товарот на контролните пакети може да се погледнат во табела 6.2.

Табела 6.2 Нови полиња во товарот на контролните пакети кај MQTT-верзија 5.0

Table 6.2 New fields in the payload of control packets in MQTT version 5.0

Код	Поле	Контролен пакет	Големина	Дополнителни карактеристики
D2.1	Will Delay Interval	CONNECT	32 бита	Во полето „Will properties“ во товарот на контролниот пакет CONNECT
D2.2	Response Topic, Correlation Data, Content type	CONNECT (Will properties)	Секое до 65.535 бајти	Во полето „Will properties“ во товарот на CONNECT-пакетот

Шема за криење на информации:

Скриените канали од група D1 припаѓаат на шемата PS.10 Random Value, додека оние од група D2 припаѓаат на шемата PS31. User-data value modulation & reserved/unused.

6.3.3.2 Скриен канал со користење на споделени сесии

Како што беше објаснето претходно, со споделените сесии се овозможува наизменично дистрибуирање на пораките помеѓу претплатниците, а со тоа и балансирање на нивното оптоварување. Оваа особина може да се користи за креирање на скриени канали и притоа постојат два начини како може да се направи тоа односно: со два претплатници кои соработуваат или со еден

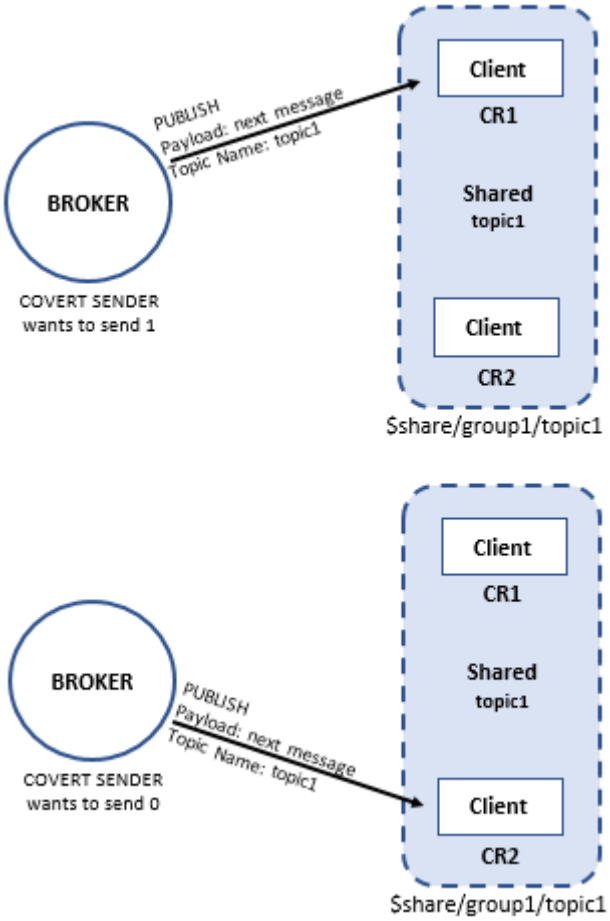
претплатник. Бинарниот, еднонасочен и директен скриен канал (D3a) помеѓу брокерот како таен испраќач и два претплатници кои соработуваат може да се имплементира на следниот начин:

- На почетокот два претплатници (CR1 и CR2) кои соработуваат мора да се договорат за некој идентификатор за споделена претплата кој ќе го користат. Потребно е да се изврши специфицирање на „Share Name“ и „Topic Filter“. На пример може да ја користат следната тема: $share/group1/topic1. По тоа тие треба да се претплатат на оваа споделена претплата.

- Ако брокерот како таен испраќач сака да пренесе таен бит 1 во тој случај тој ја испраќа следната порака на тема која е специфицирана со „Topic Filter“ до CR1. Ако брокерот сака да испрати скриен бит 0 во тој случај тој испраќа порака до претплатникот CR2 (сл. 6.2). Двата претплатници соработуваат еден со друг и тие секогаш знаат кој бит е испратен. Во основа овој канал претставува дистрибуиран скриен канал со повеќе примачи на пораки (најмалку CR1 и CR2). Во однос на начинот на испраќање на пораките тој извршува распоредување базирано на домаќин.

Шема за криење на информации:

PS. 31 User-data Value Modulation, со користење на распоредување кое е базирано на домаќин (host-based scattering).



Слика 6.2 Скриен канал со користење на споделени сесии кај MQTT-верзија 5.0

Figure 6.2 Covert Channel using shared subscriptions in MQTT version 5.0

На почетокот од поглавјето беше споменато дека со користење на карактеристиката за споделени сесии може да се креираат скриени канали на два начини. Првиот тип со користење на два претплатници беше објаснет. Вториот тип користи само еден претплатник. Притоа се креира еднонасочен, директен скриен канал (D3b) помеѓу брокерот како таен испраќач и еден претплатник како таен примач.

- На почетокот откако брокерот и претплатникот ќе се договорат за „Share Name“ и „Topic Filter“ на споделената претплата, тајниот примач мора да се претплати на споделената претплата со два различни клиентски идентификатори: CID1 и CID2.

- Ако брокерот како таен испраќач сака да испрати скриен бит 1, тој ја испраќа следната порака на тема специфицирана од „TopicFilter“ до CID1. Доколку тој

сака да испрати скриен бит 0 тој праќа порака до CID2. Тајниот примач ги знае идентификаторите, а со тоа тој може да ја протолкува скриената порака.

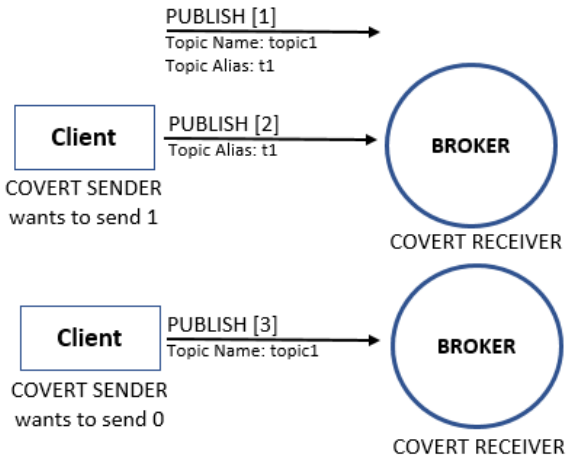
Шема за криење на информации:

PS31 User-data Value Modulation

6.3.3.3 Скриен канал со користење на алијас на тема

Скриен канал може да се креира и со користење на алијас на тема и тоа на следниот начин:

- За овој бинарен, директен скриен канал (D4) помеѓу објавувач како скриен таен испраќач и брокер како таен примач или помеѓу брокер како таен испраќач и еден или повеќе претплатници како тајни примачи, најпрвин тајниот испраќач треба да испрати PUBLISH пакет со вредности за „Topic Name“ и „Topic Alias“ кои не се празни, со што примачот може да направи мапирање помеѓу нив (сл. 6.3, чекор 1).



Слика 6.3 Скриен канал со користење на алијас на тема

Figure 6.3 Covert channel using topic alias

- Ако тајниот испраќач сака да испрати скриен бит 1, тој испраќа порака со користење на алијасот на темата (Topic Alias) и празна вредност за Topic Name (сл. 6.3, чекор 2). Ако сака да испрати скриен бит 0, тој испраќа порака со користење на името на темата за нејзино идентификување (сл. 6.3, чекор 3).

Шема за криење на информации:

PS11. Value modulation

6.3.3.4 Скриен канал со користење на интервалот за истекување на сесија

Може да се креира еднонасочен, бинарен и директен скриен канал (D5) помеѓу објавувач како таен испраќач и брокер како таен примач со користење на интервалот за истекување на сесија (Session Expiry Interval) во CONNECT контролниот пакет на следниот начин:

- Тајниот испраќач треба да користи мал временски интервал за полето „Session Expiry Interval“ и потоа тој треба да се дисконектира.

- Ако тајниот испраќач се конектира пред истекување на „Session Expiry Interval“ на страната на брокерот, се испраќа бинарно 1 до брокерот. Доколку тајниот испраќач се поврзе по истекување на интервалот за сесија, во тој случај се испраќа бинарно 0 до брокерот.

Шема за криење на информации:

Ова е скриен канал којшто припаѓа на две различни шеми за криење на информации. Најпрвин контролниот канал се реализира со конфигурирање на тајниот испраќач и на брокерот за полето „Session Expiry Interval“. Овој канал ја користи шемата PS31 User Data Value Modulation and Reserved/Unused, бидејќи се селектира 1 од n вредности за „Session Expiry Interval“. Потоа се користи шемата PT2 Message Timing, бидејќи времетраењето на повторното поврзување на тајниот испраќач го одредува скриениот бит кој ќе биде испратен.

6.3.3.4 Скриен канал со присуство/отсуство на код за причина

Една од новите особини која е воведена во MQTT 5.0 е подобрувањето на известувањето за грешки, со опционално користење на код за причина (Reason Code) и/или стринг за причина (Reason String), кои може да се најдат во ACK, AUTH и DISCONNECT-контролните пакети. Нормалниот код за причина за успех е 0. Контролните пакети CONNECT, DISCONNECT, PUBREC, PUBACK, PUBREL, AUTH и PUBCOMP имаат единствен код за причина како дел од променливото заглавие, додека SUBACK и UNSUBACK-пакетите содржат листа на еден или повеќе кодови за причина во товарот. За CONNACK-контролниот пакет користењето на код за причина е задолжително. За другите контролни пакети ова е опционално во променливото заглавие. Отсуството на код за причина е

еквивалентно на вредност 0 за „Reason Code“. Ова може да се користи за креирање на нов еднонасочен, бинарен скриен канал (D6) од брокерот кој се однесува како таен испраќач до клиентот кој ја има улогата на таен примач. Притоа, ова може да се овозможи на следниот начин:

- Најдобро е да се користат потврди (ACKs) за PUBLISH-контролниот пакет. Ова значи дека клиентот треба да користи QoS 1 или QoS 2 за испраќање на пораки. Прво, клиентот испраќа некои произволни пораки.

- Ако брокерот додаде код за причина 0 во потврдите (PUBACK-пакет за QoS 1, и PUBREC и PUBCOMP пакети за QoS 2), се испраќа бинарно 1. Ако кодот за грешка не постои во потврдите во тој случај се испраќа бинарно 0.

Шема за криење на информации:

PS.11 Value modulation

6.3.4 Нови индиректни скриени канали кај MQTT-верзија 5.0

6.3.4.1 Скриен канал со подредување на особини

Со подредување на особини може да се креира двонасочен, директен скриен канал помеѓу клиент и брокер, или индиректен скриен канал помеѓу објавувачи и претплатници (I1). За директниот канал кој било контролен пакет којшто содржи сет на особини може да биде користен, додека за индиректниот канал, логично е да се користи PUBLISH-контролниот пакет. Притоа индиректниот скриен канал може да се креира на следниот начин:

- На почетокот, две страни кои соработуваат мора да се согласат за редоследот на даден сет на особини (p_1, p_2, p_3…..p_P). За PUBLISH-контролниот пакет, има 8 различни особини кои може да се најдат во варијабилното заглавие, од кои 6 може да бидат проследени од брокерот до претплатниците. Згора на тоа, 5 од нив се праќаат непроменети. Тие се „Payload Format Indicator“, „Content Type“, „Response Topic“, „Correlation Data“ и „User Property“. Една од особините, Message Expiry Interval) може да се испрати со модификувана вредност. Дополнително, „User Property“ може да се испрати повеќе пати. Во овој случај единственото ограничување е големината на контролниот пакет.

Скриениот канал со подредување на особини може да се реализира на следниот начин:

- За бинарен канал, ако тајниот испраќач сака да испрати таен бит 1, тој ја испраќа особината p_1 пред особината p_2, додека за таен бит 0, тој ја испраќа особината p_2 пред p_1. Со вакво подредување може да се испратат p-бита за даден контролен пакет.

Шема за криење на информации:

PS2. Sequence Modulation pattern (доколку секвенцата се интерпретира)

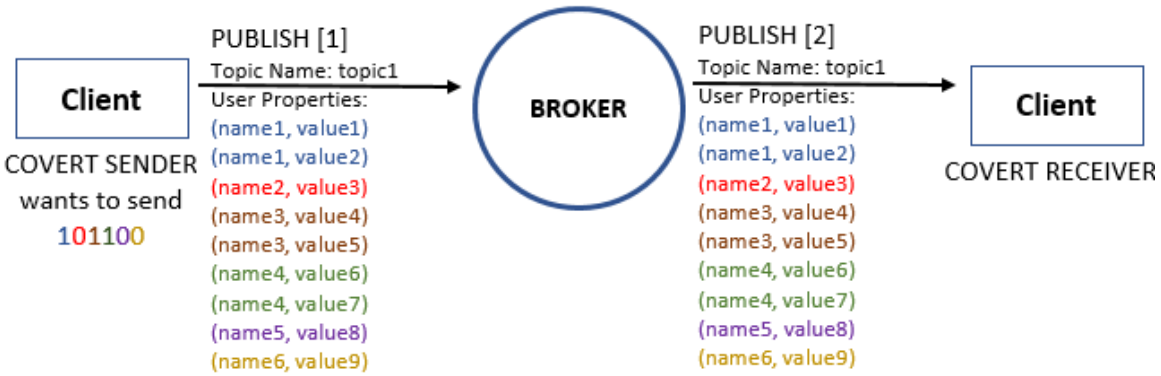
PS2.a Position pattern (ако само позицијата на еден елемент се интерпретира)

6.3.4.2 Скриен канал со удвојување на кориснички особини

Индиректен скриен канал помеѓу објавувачот и претплатниците (I2) може да се креира со користење на парови од (име, вредност) повеќе пати со исто име во еден PUBLISH-пакет. Брокерот мора да ги прати непроменето сите кориснички особини кога препраќа порака до претплатниците.

За имплементација на овој скриен канал потребно е следното:

- Тајните примачи треба да бидат претплатени на иста тема на која тајниот испраќач ги испраќа пораките.



Слика 6.4 Скриен канал со дуплирање на корисничките особини

Figure 6.4 Covert channel with duplication of user property

- Доколку тајниот испраќач сака да испрати скриен бит 1, во тој случај тој ќе користи два последователни пара од (име, вредност) со истото име. Доколку тајниот испраќач сака да испрати скриен бит 0, тој ќе употреби само еден пар на

(име, вредност). Целиот процес на криење битови може да се погледне на слика 6.4.

Шема за криење на информации:

PS3. Add Redundancy

6.3.4.3 Скриен канал со барање-одговор шема

Во новата верзија на MQTT-протоколот додадена е нова особина која во основа ја претставува барање-одговор особината која е основно правило за HTTP-клиент-сервер комуникација. Основната разлика е тоа што кај MQTT комуникацијата помеѓу клиентот и серверот не се остварува директно, туку овде имаме индиректна комуникација преку брокер. Во овој случај еден од клиентите се однесува како сервер.

Скриениот канал со барање-одговор шема се реализира на следниот начин:

- Еден од клиентите (C1) ќе ја има улогата на клиент во оваа комуникација и ќе биде претплатен на некоја тема за одговор (пример: /response/C1). Некој може да изврши конфигурирање на листата за контрола на пристап на брокерот така што само клиентот C1 може да се претплати на дадената тема.

- Друг клиент (S) ќе ја има улогата на сервер. Тој може да поседува база на податоци, централен сервис за логирање итн. Тој ќе се претплати на некоја тема (на пример: /topicS).

- Клиентот C1 може да испрати порака за барање до клиентот S, кој ја има улогата на сервер, со објавување порака на тема /topicS и притоа специфицирајќи ја темата за одговор во PUBLISH-контролниот пакет.

- Клиентот кој ја игра улогата на сервер (S) ќе испрати порака со одговор до C1, со објавување на темата за одговор која претходно беше предвидена (/response/C1). За поврзување на одговорот со оригиналното барање може да се користи особината „Correlation Data“.

Оваа особина може да се користи за креирање на еднонасочен, индиректен скриен канал (I3) од клиентот C1 до серверот S.

- На почетокот C1 треба да се претплати на две различни теми за одговор RT1 и RT2, додека клиентот S треба да се претплати на тема /topicS.

- Клиентот C1 како таен испраќач врши кодирање на пораката на следниот начин:

За испраќање на бинарно 1, тој ќе објави порака на тема /topicS со поставување на RT1 како тема за одговор.

За испраќање на бинарно 0, тој ќе објави порака на тема /topicS со поставување на RT2 како тема за одговор.

Шема за криење на информации:

PS11. Value modulation

6.3.4.4 Скриен канал со различни теми

Со користење на различни теми може да се креира n-битен двонасочен скриен канал (I4). Тој може да се имплементира и кај постарите верзии на MQTT-протоколот (на пример, кај MQTT 3.1.1). Неговата реализација се одвива на следниот начин:

- За испраќање на m-бита, учесниците во скриената комуникација треба да се согласат за користење на 2^n различни теми како T_0, T_1 ,…, T_{2^m-1} што ќе одговараат на броевите 0, 1, ... , M. Сите учесници во скриената комуникација треба да се претплатат на овие теми.

- Тајниот испраќач врши кодирање на скриени податоци на следниот начин: За праќање на бројот M претставен во бинарна форма со m-бита, тајниот испраќач објавува на тема T_m. Сите тајни примачи ќе ги добијат ажурирањата на пораките од темата T_M. На ваков начин тие може да ја заклучат скриената порака M.

Шема за криење на информации:

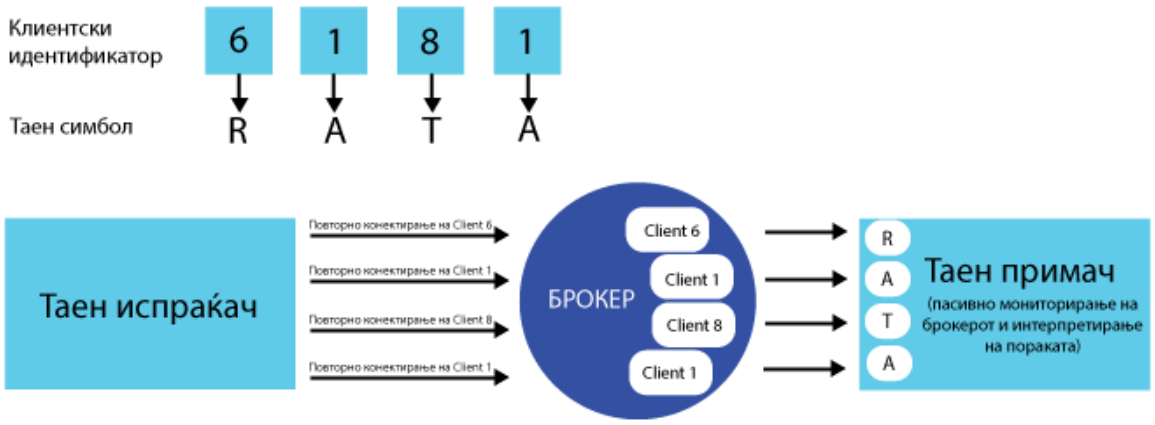
PS31. User-data Value Modulation and Reserved/Unused

6.3.4.5 Скриен канал со вештачки повторни конекции

Скриениот канал со повторни конекции се реализира со испраќање на скриени битови со тоа што тајниот испраќач врши дуплирање на клиентскиот идентификатор (Client ID) на друг клиент за да принуди повторно остварување

на конекција помеѓу клиент и брокер во специфичен редослед. Тајниот примач може да ја прочита пораката со прислушување на мрежниот сообраќај. За овој скриен канал, секој клиент i претставува скриен симбол Ci. Ова се договора однапред.

Конектирање на даден клиент со постоечки клиентски идентификатор, врши принудување на постоечкиот оригинален клиент за повторно остварување на конекција. При секоја принудна повторна конекција, сите знаци од клиентскиот идентификатор се доставуваат до брокерот. Тајниот примач го надгледува мрежниот сообраќај и со тоа може да ја интерпретира скриената порака која се состои од знаци од клиентскиот идентификатор. Притоа, скриената порака не е скриена во клиентскиот идентификатор, туку од повторната конекција на даден клиент според претходно договорениот симбол за клиенски идентификатор се толкува скриената порака. Со разгледување на сите прекини на конекции, скриената порака може да се определи со конкатенирање на симболите во редослед на ниво појавување ($C_1||C_2||C_3….||C_n$).



Слика 6.5 Скриен канал со вештачки повторни конекции

Figure 6.5 Covert channel with artificial reconnections

Шема за криење на информации:

Ова е нова шема за криење на информации со која се врши проширување на постоечката класификација. Таа е именувана како „PT15. Artificial Reconnections“ (Mileva et al., 2021).

6.3.5 Експериментална евалуација

Со цел да се докаже ефективноста и изводливоста на предложените скриени канали во овој дел е прикажана имплементација на I4 скриениот канал. Притоа овој канал е земен како пример за експерименти поради новитетот и значењето во однос на другите канали. Другите индиректни скриени канали не се евалуирани бидејќи концептот за кориснички особини (User Properties) и барање одговор мод на операција (Request/Response mode of operation) се воведени во MQTT 5.0 и притоа не постојат некои реални примери за нивна примена. Според ова не би можеле да креираме примероци од сообраќај кои ќе одговараат на легитимна употреба. Можеби во иднина, кога овие особини ќе се користат почесто, ќе се овозможи креирање на доказ за концептот за имплементација на овие скриени канали. Бидејќи не постојат реални податочни сетови кои се јавно достапни, извршено беше вештачко креирање на легитимен податочен сет. Како резултат на ова, во овој дел е претставена експериментална евалуација на I4 со испитување на следните карактеристики: неоткривање, пропусен опсег и робустност.

Во однос на предложените директни скриени канали, не е обезбедена соодветна експериментална евалуација, бидејќи има голем број на слични методи кои се достапни во литературата (Cabuk et al., 2004; Liang et al., 2018; Rowland, 1997; Murdoch & Lewis, 2005; Houmansadr & Borisov, 2011).

6.3.5.1 Експериментална евалуација на скриен канал I4

За евалуација на скриениот канал I4 и за изведување на експериментите поврзани со него, користени беа три EC2 инстанци на Амазон веб-сервиси (AWS) (сл. 6.6). На една од EC2 инстанците инсталиран беше HiveMQ брокер верзија 4.3.1. Оваа верзија на брокерот има поддршка за MQTT 5.0. Со користење на Јава програмскиот јазик имплементиран беше таен испраќач (објавувач) на друга EC2 инстанца и таен примач (претплатник) на трета EC2 инстанца. За оваа цел користена беше HiveMQ Јава клиентската библиотека, верзија 1.1.4, која исто така обезбедува поддршка за MQTT 5.0. За снимање на мрежниот сообраќај користена беше алатката Wireshark, верзија 3.2.1.

Слика 6.6 Експериментално сценарио за скриениот канал I4

Figure 6.6 Experimental scenario for covert channel I4

За изведување на експериментите за скриениот канал I4 користени беа 4 теми, каде m = 2, или $2^2 = 4$ различни теми (T0, T1, T2 и T3). За детекција на овој скриен канал користен беше методот за варијација на контрамерка (Velinov et al., 2019; Wendzel et al, 2018) и таканаречениот резултат на компресибилност, k (compressibility score). За пресметка на вредноста на k, снимени беа околу 7.500 MQTT-пакети од дадена конекција и извлечени беа темите кои се појавуваат во конекцијата. За постигнување на ова, користени беа различни големини на прозорци, односно бројот на теми кои беа разгледувани во секој проток. Понатаму, извршено беше енумерирање на сите теми на протокот (првата тема со вредност 0, втората со вредност 1 итн.) и извршено беше конкатенирање на енумерираните броеви на темите во стринг. На крај, пресметан беше резултатот на компресибилност, k, со делење на оригиналната големина на стрингот со компресираната должина на стрингот. Како втора метрика, пресметан беше бројот на промени на темите помеѓу пакетите кои се успешно испратени за првите 1.000 MQTT-пакети за проток.

Експериментите беа направени во 3 сценарија и тоа:

- *Сценарио 1*: Во кое се користи само легитимен сообраќај,
- *Сценарио 2*: Имплементиран скриен канал со тајни пораки во ASCII-формат,
- *Сценарио 3*: Имплементиран скриен канал со тајни пораки кои се шифрирани со AES (Advanced Encryption Standard).

Во прилог 2 може да се погледнат кодовите во Јава кои се однесуваат на сценариото со легитимен сообраќај (без имплементиран скриен канал) и

сценариото со имплементиран скриен канал за испраќање на порака во ASCII-формат. Кодот за праќање на скриена порака која е шифрирана со AES е сличен на сценариото со ASCII. Кај него само пораката е шифрирана, останатиот дел од имплементацијата е ист.

За секое од сценаријата, ние генериравме примероци од сообраќајот. Од првото сценарио ние ги добивме легитимните примероци. Објавувачот праќа пораки со случајно одлучување во секоја секунда дали да објави или не, за секоја од четирите теми одделно и со случајно време (во милисекунди) помеѓу секој од настаните за објавување.

Во второто сценарио, извршена е имплементација на скриениот канал I4, и кај него се праќаат скриени пораки во обичен ASCII-формат (кодирани со присуството на одделни теми). Притоа, за ова сценарио креирани беа 3 варијанти во однос на испраќањето на пораки во една секунда и тоа:

- Варијанта 1: Испраќање на порака во 1 тема од вкупно 4
- Варијанта 2: Испраќање на пораки во 2 теми од вкупно 4
- Варијанта 3: Испраќање на пораки во 3 теми од вкупно 4

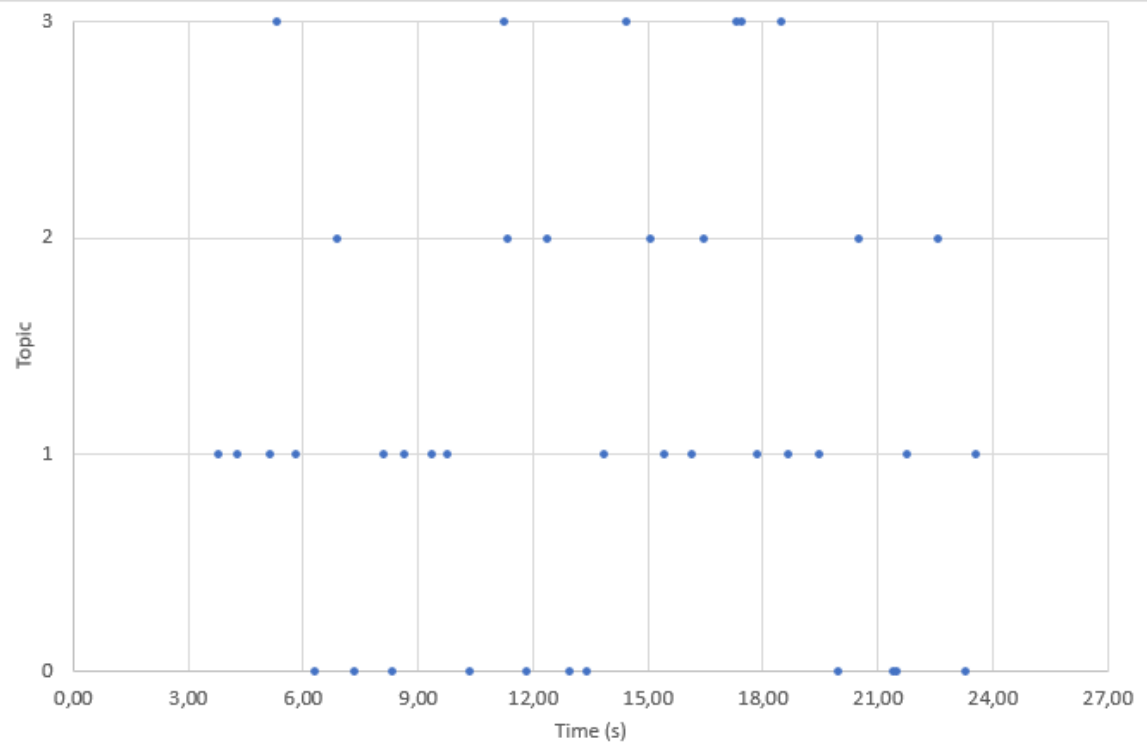
Кај секоја од варијантите се испраќаат 2 бита/секунда (варијанта 1), 4 бита/секунда (варијанта 2) и 6 бита/секунда (варијанта 3). Изборот за објавување на 2 теми од вкупно 4, е мотивиран од најголемиот број на комбинации за две теми, од што следува дека веројатноста на настанот да се објавува во две теми во секунда е најголема, споредено со другите можни настани. Дополнително извршени беа експерименти за 1 и 3 теми (секоја со 4 можни комбинации) во секунда, за да се испита што ќе се случи во тој случај. Времето помеѓу последователните објавувања на пораки е случајно (во милисекунди).

Во третото сценарио, повторно е извршена имплементација на скриениот канал I4, каде скриените пораки претходно се шифрирани со AES. Повторно и за ова сценарио беа креирани трите варијанти како и кај сценарио 2 (испраќање на пораки во 1, 2 и 3 теми во секунда, од вкупно 4). Времето помеѓу последователните настани за објавување за секоја од варијантите е случајно (исто во милисекунди). Ова сценарио се користеше за да се прикаже како

шифрирањето влијае на неоткривањето споредено со нешифрираниот сообраќај.

6.3.5.1.1 Пропусен опсег

Експериментите беа извршени за три различни пропусни опсези, 2, 4 и 6 бита/секунда, што одговара на објавување во 1, 2 и 3 теми, соодветно. Дополнително, со експериментите сакавме да утврдиме што се случува со неоткривањето кога го избираме најверојатниот настан (објавување на 2 теми) споредено со помалку веројатните настани (објавување во 1 или 3 теми). Првите 20 s од еден од 100-те извршени експерименти за скриениот канал I4 со 4 теми може да се погледнат на слика 6.7.



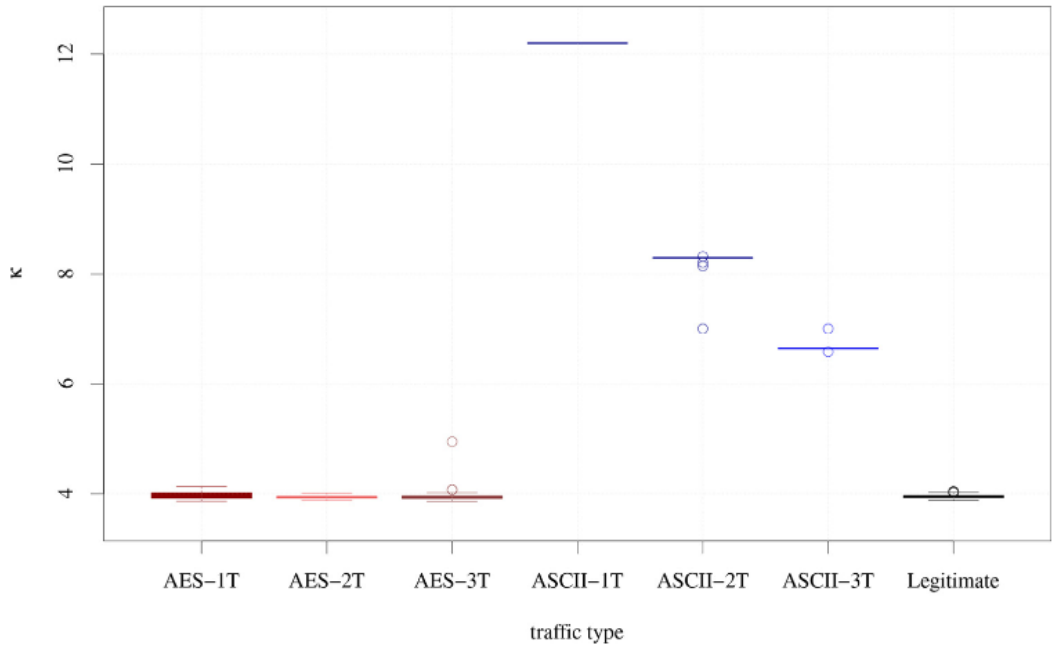
Слика 6.7 Скриен канал I4 со 4 теми, испраќање на 80 скриени битови во 20 сек.

Figure 6.7 Covert channel I4 with 4 topics, sending 80 secret bits in 20 s.

За време на овие 20 секунди се пренесуваат 80 тајни битови од 3,6 до 23,6 секунда. Слика 6.7 ги прикажува точките во време кога се случува објавување во темите T0, T1, T2 и T3.

6.3.5.1.2 Неоткривање

Добиените резултати за компресибилност се слични за AES-кодираниот скриен канал и за легитимниот сообраќај. Во двата случаја, вредноста за k е околу 4.0 за големина на прозорец (бројот на теми во проток за разгледување за компресирачки цели) од 1750 (сл. 6.8)



Слика 6.8 Резултати за компресибилност (k) за сите сценарија

Figure 6.8 Compressibility scores (k) for all scenarios

Ова се должи на фактот дека двата случаи во основа извршуваат случајна селекција за следната тема на која ќе се испрати порака. Според ова, AES скриениот канал имплементиран во овие експерименти не може да се детектира.

Компресирачките резултати кај скриениот канал кој е базиран на праќање на ASCII пораки, се разликуваат од оние на AES-кодираниот канал и на легитимниот канал. Причината за ова е тоа што ASCII податоците содржат помалку информации по бит, па според тоа може да се направи лесна компресија. Ова доведува до поголеми вредности на компресирачкиот резултат. Големини на прозорец од 1.250 до 2.500 комбинирани со соодветен праг, доведуваат до целосно (100 %) откривање, кога ASCII-каналот е потребно да се оддели од AES или легитимниот канал. За големини на прозорец од 1.750, оптималниот праг е k = 5.5, додека за поголема големина на прозорец прагот малку се зголемува.

Во однос на втората предложена метрика (бројот на промени на теми во конекција за прозорци од 500 до 3.000 пакети) не добиваме некоја корисна разлика. Бројот на промени на темите за легитимниот сообраќај се поклопува со бројот на промени на темите за двете други сценарија.

6.3.5.1.3 Робусност на скриениот канал I4

Скриениот канал I4 не е временски канал и оттука во случај на какво било константно мрежно доцнење, скриените битови ќе бидат примени во точен редослед, бидејќи сите скриени пораки ќе бидат примени со исто доцнење. Во табела 6.3 е прикажан бројот на превртени битови на страната на примачот (бит-грешки) ако воведеме различни доцнења на секоја втора секунда. Притоа, експериментите беа направени за 6 различни доцнења (50, 100, 200, 300, 400 и 500 милисекунди) и со објавување во 1, 2 или 3 теми во секунда. Грешките при трансмисија се случуваат бидејќи редоследот на примените пораки е променет во некои случаи. Ако некоја порака пристигне подоцна од нејзините последователни пораки тогаш се можни три случаи. Ако последователните пораки припаѓаат на иста тема, битовите се без грешка. Ако последователните теми се T0 и T3 во тој случај грешки ќе има кај сите четири бита. За секоја друга комбинација на теми, само 2 бита нема да бидат успешно пренесени.

Табела 6.3 Број (и процент) на превртени битови во пренесени 80 тајни битови што се должи на мрежни доцнења (на секоја втора секунда)

Table 6.3 Number (and percentage) of flipped bits transmitted in 80 secret bits due to network delays (in each 2nd second)

	50 ms	100 ms	200 ms	300 ms	400 ms	500 ms
1	0	0	0	0	0	2 (2,5 %)
2	0	0	2 (2,5 %)	4 (5 %)	8 (10 %)	14 (17,5 %)
3	2 (2,5 %)	4 (5 %)	4 (5 %)	10 (12,5 %)	22 (27,5 %)	28 (35 %)
Просечно / Average	0,67 (0,84 %)	1,33 (1,66 %)	2,00 (2,5 %)	4,67 (5,84 %)	10,00 (12,50 %)	14,67 (18,34 %)

Просечно, експерименталните резултати покажуваат дека за 50 милисекунди доцнење, 0,84 % од сите битови се примени со грешки, за 100 милисекунди се примени 1,66 %, за 200 милисекунди се примени 2,5 %, за 300 милисекунди се примени 5,84 %, за 400 милисекунди се примени 12,50 % и за 500 милисекунди доцнење се примени 18,34 % битови со грешки. Според ова може да се забележи дека со зголемување на доцнењето, се зголемуваат и битовите со грешки.

Губењето на пакетите се случува само за време кога примачот е дисконектиран од брокерот, бидејќи во други случаи, TCP обезбедува повторно пренесување на пакети. Така, при губење на еден пакет за 20 објавени пораки (5 %), ќе имаме 2 изгубени тајни битови за 40 пренесени тајни битови (5 %).

6.3.6 Противмерки за предложените скриени канали кај MQTT-верзија 5.0

Некои потенцијални противмерки може да се користат за детектирање, лимитирање или отстранување на предложените скриени канали за MQTT 5.0. Противмерките може да бидат применети само ако истиот тип на канал е користен повеќе од еднаш (или во повеќе од еден контролен пакет).

Скриените канали D1.1, D1.3, D1.4, D1.5, D1.6, D1.7, D1.8 и D2.2 може да се детектираат со надгледување на поставките на даден клиент за „Message Expiry Interval“, „Subscription Identifier“, „Topic Alias“, „Session Expiry Interval“, „Receive Maximum“, „Topic Alias Maximum“, „Maximum Packet Size“, „Server Keep Alive“ и „Will Delay Interval“. Потребно е да се извршат испитувања за појавувањата на овие поставки за ист „Client Identifier“ или исти IP-адреса или MAC адреса. За D1.3 може да се надгледува и непостоење на „Subscription Identifier„ во PUBLISH-контролните пакети.

„Response Topic“ и „Correlation Data“ од D1.2 и „Response Information“ од D1.11 се поврзани со барање-одговор (Request/Response) мод на операција. Ако некој ги користи овие полиња во овој контекст, тоа може да биде знак за користење на скриен канал. Многу различни вредности за „Response Topic“ од иста IP-адреса може исто така да се користат како потенцијален индикатор за детекција. За скриениот канал D1.2, поточно за особината „Content Type“ во PUBLISH-контролниот пакет и за особината „User Property“ која се користи кај скриениот канал D1.12, доколку некој го познава апликацискиот контекст, може

лесно да се детектира манипулацијата со овие особини. Истото ова се однесува и за скриениот канал D2.1.

Скриениот канал D1.9 може да се детектира со надгледување на назначениот клиентски идентификатор (Assigned Client Identifier) во CONNACK пакетот и со утврдување на повеќе различни вредности за иста IP-адреса или MAC адреса.

Содржината на полето „Authentication Data“ се користи кај скриениот канал D1.10. Таа е дефинирана од претходно договорен метод за автентикација. Ова поле може да содржи токен за контекст доколку се користи Kerberos (Authentication Method=„GS2-KRB5“). Овие токени вообичаено се случајно генерирани, со што овој канал тешко може да се детектира во ова сценарио. Во други случаи, кога се очекува содржината да не биде случајна, која било друга вредност може да поттикне сомневање.

Втората верзија на скриениот канал D3, D3b, може лесно да се детектира со утврдување на бројот на различни клиентски идентификатори за иста IP или MAC-адреса на ист брокер, во исто време. Првата верзија на овој скриен канал, D3a, е скоро невозможно да се детектира. Ова е така бидејќи нема правило за тоа кој претставник од групата брокерот ќе го избере за испраќање на дадена порака. Единствениот начин е да се побара колаборативна комуникација помеѓу клиентите во групата за да се овозможи детектирање на каналот.

Еден начин за детектирање на скриениот канал D4 е да се снима мапирањето помеѓу „Topic Names“ и „Topic Aliases“ во различни комуникации со цел да се открие невообичаено однесување како објавување на дадена тема понекогаш со користење на „Topic Name“ а понекогаш со „Topic Alias“.

За детектирање на скриениот канал D5, може да се изврши снимање на полето „Session Expiry Interval“ во CONNECT-контролниот пакет. Присуството на многу мали вредности за ова поле може да биде знак за постоење на овој скриен канал, бидејќи мали вредности се потребни за пренесување на бинарни нули.

Скриениот канал D6 може да се детектира со надгледување на присуство/отсуство на „Reason Code“ 0 во потврдите (ACKs). Доколку има повеќе

потврди со „Reason Code“ 0, и повеќе потврди без него во одредено време, ова може да биде знак за постоење на овој скриен канал.

Индиректниот скриен канал I1 може да се детектира со надгледување на редоследот на особините во PUBLISH-контролните пакети. Овој скриен канал може да се детектира дури и кога брокерот активно влијае на MQTT-пакетите со постојано променување на моменталниот редослед на особините според претходно специфициран начин.

За детектирање на скриениот канал I2 може да се изврши броење на појавувањата на последователни (name, value) парови со истото име. Овој канал не може да се спречи бидејќи брокерот треба да ги испрати сите кориснички особини (User Properties) непроменети и мора да го зачува нивниот редослед кога ја проследува пораката.

Индиректниот канал I3 е многу тежок за детектирање бидејќи ситуацијата во која клиент игра улога на сервер и нуди два или повеќе различни сервиси на другите клиенти со користење на барање-одговор мод на операција е прилично општа. Како и да е, доколку знаеме дека одреден клиент (како сервер) нуди само еден сервис, набљудувањето на две различни теми за одговор (Response Topics) за истиот клиентски идентификатор или иста IP или MAC-адреса може да се третира како аномалија и да се докаже постоењето на скриен канал.

Еден можен начин за детектирање на скриениот канал I4 е со мерење на времињата помеѓу последователни објавувања од ист клиентски идентификатор (или иста IP или MAC-адреса) на дадена тема. Значајните разлики споредени со записите од легитимниот сообраќај може да бидат индикатор за постоење на скриен канал.

За детектирање на скриениот канал I5, може да се изврши снимање на (Client Identifier, MAC address) парови и надгледување на најмалку два различни пара со ист клиентски идентификатор или многу различни парови со иста MAC-адреса. Втората опција е со следење на бројот на дисконекции и реконекции со код за причина (Reason Code) 0x8E кој укажува на преземена сесија (Session taken over) кај DISCONNECT-контролен пакет. Доколку овој број е голем, ова може да биде знак за постоење на скриениот канал I5. Покрај тоа, големиот број

на повторни поврзувања може да се разгледува и како DoS (Denial od Service) напад.

7. ЗАКЛУЧОК

Бројот на уреди кај интернет на нештата во минатото постојано се зголемуваше. Во иднина се очекува дека овој тренд ќе продолжи. Според предвидувањата се очекува дека бројот на уреди до 2025 година ќе изнесува околу 75 милијарди. Станува збор за голем број на уреди и затоа од огромна важност е да се обезбедат соодветни безбедносни механизми сè со цел да имаме соодветна доверлива комуникација и безбедно пренесување на податоци. Протоколите кои се користат често се подложни на тајно пренесување на податоци преку легитимните податочни текови. На ваков начин се создаваат скриени канали кои може да се користат за пренесување на податоци. Од огромна важност е нивната детекција и развивањето на механизми за соодветно управување и отстранување.

Кај интернетот на нештата посебно се важни протоколите кои се наоѓаат на највиското ниво од стекот на протоколи односно апликациското ниво или на мрежниот слој од предложената 3-слојна архитектура. Најпознати протоколи кои се користат кај интернет на нештата се: CoAP, MQTT, XMPP и AMQP. Пред почетокот на истражувањата во однос на подложноста на скриени канали, податоци постоеја само за XMPP-протоколот. За него предложени беа неколку складирачки канали. За другите протоколи сè уште не постоеја истражувања за скриени канали. Затоа истражувањата ги насочивме кон откривање на скриени канали кај CoAP и MQTT-протоколите. Притоа, за протоколот MQTT, разгледана беше можноста за подложност на скриени канали за неговите две верзии 3.1.1 и 5.0.

За CoAP-протоколот направени се следните придонеси за научната сфера:

- Предложени се вкупно 8 скриени канали од кое 6 се складирачки додека 2 се временски.
- Направена е експериментална евалуација на скриениот канал кој ги користи методите PUT и DELETE.
- Со помош на Cooja симулаторот разгледана е потрошувачката на енергија за Zolertia Z1 модул со имплементиран и без имплементиран скриен канал. Според добиените податоци имплементацијата на скриениот канал

со PUT и DELETE методите не влијае многу на потрошувачката на енергија.

- Предложени се соодветни противмерки за детекција, лимитирање или отстранување на новите скриени канали.

Клучните придонеси за научната сфера за скриени канали кај протоколот MQTT-верзија 3.1.1 се следните:

- Предложени се 13 мрежни скриени канали од кои 6 се директни додека 7 индиректни.
- Направена е експериментална евалуација на скриениот канал кој користи подредување на теми и ажурирања со присуство/отсуство.
- Разгледани се три особини за предложениот скриен канал и тоа: пропусен опсег, неоткривање и робусност.
- Пропусниот опсег за овој скриен канал зависи од бројот на темите кои се користат за испраќање на пораки. Во експрименталното сценарио користени се најмногу 4 теми, според што максималниот пропусен опсег изнесува 4 bps.
- Направено е истражување за детекција на скриениот канал во случај кога имаме легитимен сообраќај, испраќање на ASCII-пораки и испраќање на пораки кои се шифрирани со AES. Според добиените резултати детекцијата значително се намалува во случајот кога имаме шифриран сообраќај со AES.
- За робусноста е откриено дека со зголемување на времето на доцнење на пакетите се зголемува и превртувањето на битовите. Резултатите покажуваат дека за 10 ms доцнење, 2,5 % од битовите се примени со доцнење, за 50ms, 3,9 % се примени со грешка, и за 100ms, 8,6 % од битовите се примени со грешка.
- Предложени се соодветни противмерки за детекција, лимитирање или отстранување на новите скриени канали.

Клучни придонеси за научната сфера кои се однесуваат на скриени канали кај протоколот MQTT-верзија 5.0 се следните:

- Предложени се 18 директни и 5 индиректни скриени канали.

- Направена е експериментална евалуација на скриениот канал кој користи различни теми.
- Испитување на пропусниот опсег, неоткривањето и робусноста за предложениот скриен канал.
- Во однос на пропусниот опсег, експериментите се извршени за три различни пропусни опсези, 2, 4 и 6 бита/секунда, што одговара на објавување во 1, 2 и 3 теми, соодветно.
- За неоткривањето ги имаме истите сценарија како и кај верзијата 3.1.1 на MQTT-протоколот: испраќање на пораки преку легитимен комуникациски канал, испраќање на ASCII пораки и испраќање на пораки кои се шифрирани со AES. Според добиените резултати имплементираниот AES-скриен канал не може да се детектира.
- За робусноста, експерименталните резултати покажуваат дека за 50 милисекунди доцнење, 0,84 % од сите битови се примени со грешки, за 100 милисекунди се примени 1,66 %, за 200 милисекунди се примени 2,5 %, за 300 милисекунди се примени 5,84 %, за 400 милисекунди се примени 12,50 % и за 500 милисекунди доцнење се примени 18,34 % битови со грешки. Според ова може да се забележи дека со зголемување на доцнењето, се зголемува и процентот на битови со грешки.
- Предложени се соодветни противмерки за детекција, лимитирање или отстранување на новите скриени канали.

Нашите идни истражувања ќе бидат насочени кон развивање на активни и пасивни чувари. Тие ќе имаат за цел да вршат детекција и отстранување на предложените мрежни скриени канали кај протоколите кои се користат кај интернет на нештата.

КОРИСТЕНА ЛИТЕРАТУРА (REFERENCES)

Lampson, B. W. (1973). A note on the confinement problem. Communications of the ACM, 16(10), 613-615.

Mazurczyk, W., Wendzel, S., Zander, S., Houmansadr, A., & Szczypiorski, K. (2016). Information hiding in communication networks: fundamentals, mechanisms, applications, and countermeasures. John Wiley & Sons.

Zander, S., Armitage, G., & Branch, P. (2007). A survey of covert channels and countermeasures in computer network protocols. IEEE Communications Surveys & Tutorials, 9(3), 44-57.

Mileva, A., & Panajotov, B. (2014). Covert channels in TCP/IP protocol stack-extended version. Central European Journal of Computer Science, 4(2), 45-66.

Murdoch, S. J. (2007). Covert channel vulnerabilities in anonymity systems (No. UCAM-CL-TR-706). University of Cambridge, Computer Laboratory.

Feamster, N., Balazinska, M., Harfst, G., Balakrishnan, H., & Karger, D. R. (2002, August). Infranet: Circumventing Web Censorship and Surveillance. In USENIX Security Symposium (pp. 247-262).

Wendzel, S., & Keller, J. (2014). Hidden and under control. annals of telecommunications-annales des télécommunications, 69(7), 417-430.

Patel, A., Shah, M., Chandramouli, R., & Subbalakshmi, K. P. (2007). Covert channel forensics on the internet: Issues, approaches, and experiences. IJ Network Security, 5(1), 41-50.

Ulz, T., Feldbacher, M., Pieber, T. W., & Steger, C. (2019, January). Sensing Danger: Exploiting Sensors to Build Covert Channels. In ICISSP (pp. 100-113).

Caviglione, L., Merlo, A., & Migliardi, M. (2018). Covert channels in IoT deployments through data hiding techniques. In 2018 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA) (pp. 559-563). IEEE.

CEH: Certified Ethical Hacker (2012). Overt and Covert Channels. Преземено на 26 април 2021 г. http://certifiedethicalhackerceh.blogspot.com/2012/01/overt-and-covert-channels.html

Ogurtsov, N., Orman, H., Schroeppel, R., O'Malley, S., & Spatscheck, O. (1996). Covert channel elimination protocols. TR96-14. The University of Arizona.

Urbanski, M., Mazurczyk, W., Lalande, J. F., & Caviglione, L. (2017). Detecting local covert channels using process activity correlation on android smartphones.

International Journal of Computer Systems Science and Engineering, 32(2), 71-80.

Lucena, N. B., Lewandowski, G., & Chapin, S. J. (2005). Covert channels in IPv6. In International Workshop on Privacy Enhancing Technologies (pp. 147-166). Springer, Berlin, Heidelberg.

Forte, D. V., Maruti, C., Vetturi, M. R., & Zambelli, M. (2005). SecSyslog: An approach to secure logging based on covert channels. In First International Workshop on Systematic Approaches to Digital Forensic Engineering (SADFE'05) (pp. 248-263). IEEE.

Mazurczyk, W., & Kotulski, Z. (2006). New security and control protocol for VoIP based on steganography and digital watermarking. arXiv preprint cs/0602042.

Houmansadr, A., Kiyavash, N., & Borisov, N. (2009, February). RAINBOW: A Robust And Invisible Non-Blind Watermark for Network Flows. In NDSS (Vol. 47, pp. 406-422).

Wang, X., & Reeves, D. S. (2003). Robust correlation of encrypted attack traffic through stepping stones by manipulation of interpacket delays. In Proceedings of the 10th ACM conference on Computer and communications security (pp. 20-29).

Wang, X., Chen, S., & Jajodia, S. (2005). Tracking anonymous peer-to-peer voip calls on the internet. In Proceedings of the 12th ACM conference on Computer and communications security (pp. 81-91).

Mehic, M., Slachta, J., & Voznak, M. (2016). Whispering through DDoS attack. Perspectives in Science, 7, 95-100.

Girling, C. G. (1987). Covert channels in LAN's. IEEE Transactions on software engineering, 13(2), 292.

Rowland, C. H. (1997). Covert channels in the TCP/IP protocol suite.

Handel, T. G., & Sandford, M. T. (1996, May). Hiding data in the OSI network model. In International Workshop on Information Hiding (pp. 23-38). Springer, Berlin, Heidelberg.

Hintz, A. (2002). Covert channels in TCP and IP headers. Presentation at DEFCON, 10, 16.

Kundur, D., & Ahsan, K. (2003). Practical Internet steganography: data hiding in IP. Proc. Texas wksp. security of information systems.

Wolf, M. (1989, April). Covert channels in LAN protocols. In Local Area Network Security Workshop (pp. 89-101). Springer, Berlin, Heidelberg.

Berk, V., Giani, A., & Cybenko, G. (2005). Detection of covert channel encoding in network packet delays.

Cabuk, S. (2006). Network covert channels: Design, analysis, detection, and elimination (Doctoral dissertation, Purdue University).

IBM (1999). IBM Podcast. Преземено на 5 мај 2021 г. https://www.ibm.com/podcasts/software/websphere/connectivity/piper_diaz_nipper_mq_tt_11182011.pdf

HiveMQ (2019). Using MQTT and HiveMQ to Build the Connected Car. Преземено на 5 мај 2021 г. https://www.hivemq.com/blog/creating-the-connected-car-platform/

HiveMQ (2020a). HiveMQ‘s Reliable IoT Communication Enables Real-time Monitoring of Matternet’s Autonomous Drones. Преземено на 6 мај 2021 г. https://www.hivemq.com/case-studies/matternet/

HiveMQ (2020b). Berlex Connects Traffic Signals to the Cloud with HiveMQ. Преземено на 6 мај 2021 г. https://www.hivemq.com/case-studies/berlex/

Rompas, P. S., & Wardana, A. A. (2017, October). Robust flood monitoring platform using message queueing telemetry transport protocol. In 2017 International Conference on Information Technology Systems and Innovation *(ICITSI)* (pp. 234-238). IEEE.

Facebook Engineering (2011). Building Facebook Messenger. Преземено на 7 мај 2021 г. https://engineering.fb.com/2011/08/12/android/building-facebook-messenger/

OASIS (2014). MQTT Version 3.1.1. OASIS Standard. Преземено на 7 мај 2021 г. http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html

AWS (2021). Amazon Web Services. Преземено на 7 мај 2021 г. https://aws.amazon.com/

Mosquitto (2021). Eclipse Mosquitto™ - An open source MQTT broker. Преземено на 7 мај 2021 г. https://mosquitto.org/

Eclipse Foundation (2021). Eclipse Paho Java Client. Преземено на 7 мај 2021 г. https://www.eclipse.org/paho/index.php?page=clients/java/index.php

MQTT.fx (2021). The JavaFX based MQTT Client. Преземено на 7 мај 2021 г. https://mqttfx.jensd.de/

Wireshark (2021). Wireshark tool. Преземено на 7 мај 2021 г. https://www.wireshark.org/

OASIS (2019). MQTT version 5.0. OASIS Standard. Преземено на 8 мај 2021 г. https://docs.oasis- open.org/mqtt/mqtt/v5.0/mqtt- v5.0.html

HiveMQ (2018). HiveMQ 4. Преземено на 9 мај 2021 г. https://www.hivemq.com/hivemq-4-whats-new/

HiveMQ (2017). 10,000,000 MQTT Clients - HiveMQ Cluster Benchmark Paper. Преземено на 10 мај 2021 г. https://www.hivemq.com/downloads/hivemq-10-mio-benchmark.pdf

GitHub (2019). HiveMQ Extension SDK. Преземено на 11 мај 2021 г. https://github.com/hivemq/hivemq-extension-sdk

HiveMQ (2016). Load Balancing With Shared Subscriptions - MQTT Client. Преземено на 12 мај 2021 г. https://www.hivemq.com/blog/mqtt-client-load-balancing-with-shared-subscriptions/

Shadowserver (2021). Open MQTT Report - Expanding the hunt for vulnerable IoT devices. Преземено на 07.04.2021г. https://www.shadowserver.org/news/open-mqtt-report-expanding-the-hunt-for-vulnerable-iot-devices/

Avast Blog (2018). Are smart homes vulnerable to hacking?. Преземено на 7.4.2021г. https://blog.avast.com/mqtt-vulnerabilities-hacking-smart-homes

Cabaj, K., Caviglione, L., Mazurczyk, W., Wendzel, S., Woodward, A., & Zander, S. (2018). The new threats of information hiding: The road ahead. IT professional, 20(3), 31-39.

W. Mazurczyk and L. Caviglione, ‘‘Information hiding as a challenge for malware detection,’’ IEEE Security Privacy, vol. 13, no. 2, pp. 89–93, Mar./Apr. 2015.

Jia, Y., Xing, L., Mao, Y., Zhao, D., Wang, X., Zhao, S., & Zhang, Y. (2020, May). Burglars' iot paradise: Understanding and mitigating security risks of general messaging protocols on iot clouds. In 2020 IEEE Symposium on Security and Privacy (SP) (pp. 465-481). IEEE.

Wendzel, S., Zander, S., Fechner, B., & Herdin, C. (2015). Pattern-based survey and categorization of network covert channel techniques. ACM Computing Surveys (CSUR), 47(3), 1-26.

W. Mazurczyk, S. Wendzel, Z. Zander, A. Houmansadr, and K. Szczypiorski, Information Hiding in Communication Networks: Fundamentals, Mechanisms, Applications, and Countermeasures. Hoboken, NJ, USA: Wiley, 2016.

Kumar, J. S., & Patel, D. R. (2014). A survey on internet of things: Security and privacy issues. International Journal of Computer Applications, 90(11).

Komninos, N., Philippou, E., & Pitsillides, A. (2014). Survey in smart grid and smart home security: Issues, challenges and countermeasures. IEEE Communications Surveys & Tutorials, 16(4), 1933-1954.

Glanzer, H., Krammer, L., & Kastner, W. (2016). Increasing security and availability in KNX networks. Sicherheit 2016-Sicherheit, Schutz und Zuverlässigkeit.

Granzer, W., Kastner, W., Neugschwandtner, G., & Praus, F. (2006). Security in networked building automation systems. In 2006 IEEE International Workshop on Factory Communication Systems (pp. 283-292). IEEE.

Müller, C., Armknecht, F., Benenson, Z., & Morgner, P. (2016). On the security of the ZigBee light link touchlink commissioning procedure. Sicherheit 2016-Sicherheit, Schutz und Zuverlässigkeit.

Badenhop, C. W., Graham, S. R., Ramsey, B. W., Mullins, B. E., & Mailloux, L. O. (2017). The Z-Wave routing protocol and its security implications. computers & security, 68, 112-129.

Wendzel, S. (2012). Covert and side channels in buildings and the prototype of a building-aware active warden. In 2012 IEEE International Conference on Communications (ICC) (pp. 6753-6758). IEEE.

Patuck, R., & Hernandez-Castro, J. (2013). Steganography using the extensible messaging and presence protocol (XMPP). arXiv preprint arXiv:1310.0524.

Wendzel, S., Mazurczyk, W., & Haas, G. (2017). Don't you touch my nuts: Information hiding in cyber physical systems. In 2017 IEEE Security and Privacy Workshops (SPW) (pp. 29-34). IEEE.

Mileva, A., Velinov, A., Stojanov, D. (2018) New Covert Channels in Internet of Things. In: The 12th International Conference on Emerging Security Information, Systems and Technologies - SECURWARE 2018, September 16-20, 2018, Venice, Italy.

Velinov, A., Mileva, A., & Stojanov, D. (2019). Power consumption analysis of the new covert channels in coap. International Journal On Advances in Security, 12(1 & 2), 42-52.

C. Rowland. (1997). Covert Channels in the TCP/IP Protocol Suite. Преземено на 8.4.2021г., https://firstmonday.org/ojs/index.php/fm/article/view/528/449

I. Zelenchuk. (2004). Skeeve–ICMP Bounce Tunnel. Преземено на 8.4.2021г., http://www.gray-world.net/poc_skeeve.shtml

Danezis, G. (2008). Covert communications despite traffic data retention. In International Workshop on Security Protocols (pp. 198-214). Springer, Berlin, Heidelberg.

Seclists.org. (2005). DNS Covert Channels and Bouncing Techniques. Преземено на 08.04.2021г., https://seclists.org/fulldisclosure/2005/Jul/att-452/

SecuriTeam. (2005). ‘Skeeve – Software For Creating Cover Channel With ICMP Tunnel’. Преземено на 8.4.2021г., https://seclists.org/fulldisclosure/2005/Jul/att-452/

Schmidbauer, T., Wendzel, S., Mileva, A., & Mazurczyk, W. (2019, August). Introducing dead drops to network steganography using ARP-caches and SNMP-walks. In Proceedings of the 14th International Conference on Availability, Reliability and Security (pp. 1-10).

Ambrosin, M., Conti, M., Gasti, P., & Tsudik, G. (2014). Covert ephemeral communication in named data networking. In Proceedings of the 9th ACM symposium on Information, computer and communications security (pp. 15-26).

Denney, K., Uluagac, A. S., Akkaya, K., & Bhansali, S. (2016). A novel storage covert channel on wearable devices using status bar notifications. In 2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC) (pp. 845-848). IEEE.

Islam, M. N., Patil, V. C., & Kundu, S. (2017). Determining proximal geolocation of IoT edge devices via covert channel. In 2017 18th International Symposium on Quality Electronic Design (ISQED) (pp. 196-202). IEEE.

Statista (2016). Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025. Преземено на 26 април 2021 г. https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/

Ericsson (2010). CEO to shareholders: 50 billion connections 2020. Преземено на 26 април 2021 г. https://www.ericsson.com/en/press-releases/2010/4/ceo-to-shareholders-50-billion-connections-2020

Cisco (2011). The Internet of Things How the Next Evolution of the Internet Is Changing Everything. Преземено на 26 април 2021 г. https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf

IHS (2016). IoT platforms: enabling the Internet of Things. Преземено на 26 април 2021 г. https://cdn.ihs.com/www/pdf/enabling-IOT.pdf

Gartner (2017). Leading the IoT - Gartner Insights on How to Lead in a Connected World. Преземено на 26 април 2021 г. https://www.gartner.com/imagesrv/books/iot/iotEbook_digital.pdf

IEEE Spectrum (2016). Popular Internet of Things Forecast of 50 Billion Devices by 2020 Is Outdated. Преземено на 26 април 2021 г. https://spectrum.ieee.org/tech-talk/telecom/internet/popular-internet-of-things-forecast-of-50-billion-devices-by-2020-is-outdated

Techjury (2021). How Many IoT Devices Are There in 2021? [All You Need To Know]. Преземено на 27 април 2021 г. https://techjury.net/blog/how-many-iot-devices-are-there/#gref

IDC (2020). IoT Growth Demands Rethink of Long-Term Storage Strategies, says IDC. Преземено на 27 април 2021 г. https://www.idc.com/getdoc.jsp?containerId=prAP46737220

Google Trends (2021). Compere MQTT, CoAP, XMPP and AMQP. Преземено на 28 април 2021 г. https://trends.google.de/trends/explore?hl=en&date=today%205-y&q=mqtt,coap,xmpp,amqp

Lin, J., Yu, W., Zhang, N., Yang, X., Zhang, H., & Zhao, W. (2017). A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications. IEEE Internet of Things Journal, 4(5), 1125-1142.

Frontiers in ICT (2019). Toward Industry 4.0 With IoT: Optimizing Business Processes in an Evolving Manufacturing Factory. Преземено на 28 април 2021 г. https://www.frontiersin.org/articles/10.3389/fict.2019.00017/full

Network Information Hiding Patterns. (2019). New Sub-pattern PS11.c (Value Influencing Pattern). Преземено на 9.4.2021г., http://ih-patterns.blogspot.com/p/references-1-s.html

Mileva, A., Velinov, A., Hartmann, L., Wendzel, S., & Mazurczyk, W. (2021). Comprehensive analysis of MQTT 5.0 susceptibility to network covert channels. computers & security, 104, 102207.

Shelby, Z., Hartke, K., & Bormann, C. (2014). The constrained application protocol (CoAP).

Zolertia (2010). Z1 Datasheet. Преземено на 30 април 2021 г. http://zolertia.sourceforge.net/wiki/images/e/e8/Z1_RevC_Datasheet.pdf

IANA (2012). Constrained RESTful Environments (CoRE) Parameters. Преземено на 25 мај 2021. https://www.iana.org/assignments/core-parameters/core-parameters.xhtml

Gonzalez-Jimenez, J., Galindo, C., Melendez-Fernandez, F., & Ruiz-Sarmiento, J. R. (2013, July). Building and exploiting maps in a telepresence robotic application. In 10th international conference on informatics in control, automation and robotics (ICINCO).

Wendzel, S., Mazurczyk, W., & Zander, S. (2016). Unified Description for Network Information Hiding Methods. *J. UCS*, 22(11), 1456-1486.

Mazurczyk, W., Wendzel, S., & Cabaj, K. (2018, August). Towards deriving insights into data hiding methods using pattern-based approach. In Proceedings of the 13th International Conference on Availability, Reliability and Security (pp. 1-10).

Cabuk, S., Brodley, C. E., & Shields, C. (2009). IP covert channel detection. ACM Transactions on Information and System Security (TISSEC), 12(4), 1-29.

Wendzel, S., Link, F., Eller, D., & Mazurczyk, W. (2019). Detection of Size Modulation Covert Channels Using Countermeasure Variation. J. UCS, 25(11), 1396-1416.

Cabuk, S., Brodley, C. E., & Shields, C. (2004). IP covert timing channels: design and detection. In Proceedings of the 11th ACM conference on Computer and communications security (pp. 178-187).

Liang, C., Tan, Y. A., Zhang, X., Wang, X., Zheng, J., & Zhang, Q. (2018). Building packet length covert channel over mobile VoIP traffics. Journal of Network and Computer Applications, 118, 144-153.

Murdoch, S. J., & Lewis, S. (2005, June). Embedding covert channels into TCP/IP. In International Workshop on Information Hiding (pp. 247-261). Springer, Berlin, Heidelberg.

Houmansadr, A., & Borisov, N. (2011, May). CoCo: coding-based covert timing channels for network flows. In International Workshop on Information Hiding (pp. 314-328). Springer, Berlin, Heidelberg.

Прилог 1

Кодови напишани во Јава за испраќање на легитимен сообраќај без имплементиран скриен канал (ICC.2) и испраќање на скриени пораки во ASCII-формат кај MQTT-верзија 3.1.1

Код за испраќање на легитимен сообраќај (без имплементиран скриен канал):

```
package org.eclipse.pahodemo;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.Random;
import java.util.concurrent.TimeUnit;
import java.util.zip.ZipEntry;
import java.util.zip.ZipInputStream;
import java.util.zip.ZipOutputStream;
import org.eclipse.paho.client.mqttv3.MqttClient;
import org.eclipse.paho.client.mqttv3.MqttException;
@SuppressWarnings("deprecation")
public class PahoDemo {

  MqttClient client;
  int arr[][]= new int[600][4];
  int i,j;
```

```
double num1, num2, num3, num4, s;
public PahoDemo() {}
public static void main(String[] args) throws Exception {
  new PahoDemo().doDemo();
}
public void doDemo() throws Exception {
  try {
    client = new MqttClient("tcp://18.191.134.106:1883", "pahomqttpublish12");
    client.connect();
    if(client.isConnected())
    {
      for(i=0;i<600;i++)
      {
          for(j=0;j<4;j++)
          {
                  Random rand=new Random();
                  int k=rand.nextInt();
                  arr[i][j]=Math.abs(k)%2;
                  System.out.print(arr[i][j]+" ");
          }
          System.out.print("\n");
      }
      for(i=0;i<600;i++)
      {
          Random rnd=new Random();
          num1=rnd.nextDouble();
          num2=rnd.nextDouble();
          num3=rnd.nextDouble();
          num4=rnd.nextDouble();
```

```
s=(num1+num2+num3+num4)/1000;
num1 /=s;
num2 /=s;
num3 /=s;
num4 /=s;
System.out.print("num1="+num1+"  num2="+num2+"  num3="+num3+" num4="+num4+"\n");
if(arr[i][0]==1)
{
        TimeUnit.MILLISECONDS.sleep((long) num1);
        String str="message1";
            String topic="T1";
            // TODO Auto-generated method stub
            try {
                client.publish(topic, str.getBytes(), 0, false);
                System.out.print("Publish in T1: "+str+"\n");
            } catch (MqttException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }

}
else if(arr[i][0]==0)
{

        TimeUnit.MILLISECONDS.sleep((long) num1);

}

if(arr[i][1]==1)
```

```
{
        TimeUnit.MILLISECONDS.sleep((long) num2);
        String str="message2";
                String topic="T2";
                // TODO Auto-generated method stub
                try {
                        client.publish(topic, str.getBytes(), 0, false);
                        System.out.print("Publish in T2: "+str+"\n");
                } catch (MqttException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
}
else if(arr[i][1]==0)
{
        TimeUnit.MILLISECONDS.sleep((long) num2);
}

if(arr[i][2]==1)
{

        TimeUnit.MILLISECONDS.sleep((long) num3);
        String str="message3";
                String topic="T3";
                // TODO Auto-generated method stub
                try {
                        client.publish(topic, str.getBytes(), 0, false);
                        System.out.print("Publish in T3: "+str+"\n");
                } catch (MqttException e) {
```

```
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
    }
    else if(arr[i][2]==0)
    {

            TimeUnit.MILLISECONDS.sleep((long) num3);
    }

    if(arr[i][3]==1)
    {
            TimeUnit.MILLISECONDS.sleep((long) num4);
            String str="message4";
                String topic="T4";
                // TODO Auto-generated method stub
                try {
                        client.publish(topic, str.getBytes(), 0, false);
                        System.out.print("Publish in T4: "+str+"\n");
                } catch (MqttException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
    }
    else if(arr[i][3]==0)
    {
                                TimeUnit.MILLISECONDS.sleep((long) num4);
    }
}
```

```
      }
      else
      {
        System.out.print("MQTT service connect failed");
      }
    } catch (MqttException e) {
      e.printStackTrace();
    }
  }
}
```

Код за испраќање на скриени пораки во ASCII-формат:

```
package org.eclipse.pahodemo;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.Random;
import java.util.concurrent.TimeUnit;
import java.util.zip.ZipEntry;
import java.util.zip.ZipInputStream;
import java.util.zip.ZipOutputStream;
```

```
import org.eclipse.paho.client.mqttv3.MqttClient;
import org.eclipse.paho.client.mqttv3.MqttException;

@SuppressWarnings("deprecation")
public class PahoDemo {

 MqttClient client;
 int arr[][]= new int[600][4];
 int i,j;
 double num1, num2, num3, num4, s;
 public PahoDemo() {}

 public static void main(String[] args) throws Exception {
  new PahoDemo().doDemo();
 }

 public void doDemo() throws Exception {
  try {
   client = new MqttClient("tcp://18.223.113.103:1883", "pahomqttpublish12");
   client.connect();
   if(client.isConnected())
   {

    String str1="Once upon a midnight dreary, while I pondered, weak and weary\r\n"
   +
            "Over many a quaint and curious volume of forgotten lore-\r\n" +
            "While I nodded, nearly napping,suddenly there came a tapping\r\n" +
            "As of some one gently rapping,rapping at my chamber door.\r\n" +
            "Tis some visitor, I muttered, tapping at my chamber door- \r\n" +
            "";
```

```
String str_bin=strToBinary(str1);
System.out.print(str_bin);
System.out.print("\n");
int[] str_in=new int[str_bin.length()];

for(int i=0;i<str_bin.length();i++)
        str_in[i]=Character.digit(str_bin.charAt(i),10);

int count=0;

for(int i=0;i<600;i++)
{
    for(int j=0;j<4;j++)
    {
        arr[i][j]=str_in[count];
        System.out.print(arr[i][j]+" ");
        count++;
    }
    System.out.print("\n");
}

for(i=0;i<600;i++)
{
    Random rnd=new Random();
    num1=rnd.nextDouble();
    num2=rnd.nextDouble();
    num3=rnd.nextDouble();
    num4=rnd.nextDouble();
```

```
s=(num1+num2+num3+num4)/1000;
num1 /=s;
num2 /=s;
num3 /=s;
num4 /=s;
System.out.print("num1="+num1+"  num2="+num2+"  num3="+num3+" num4="+num4 +"\n");
if(arr[i][0]==1)
{
        TimeUnit.MILLISECONDS.sleep((long) num1);
        String str="message1";
            String topic="T1";
            // TODO Auto-generated method stub
            try {
                client.publish(topic, str.getBytes(), 0, false);
                System.out.print("Publish in T1: "+str+"\n");
            } catch (MqttException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }

}
else if(arr[i][0]==0)
{

        TimeUnit.MILLISECONDS.sleep((long) num1);

}
```

```
if(arr[i][1]==1)
{
	TimeUnit.MILLISECONDS.sleep((long) num2);
	String str="message2";
		String topic="T2";
		// TODO Auto-generated method stub
		try {
			client.publish(topic, str.getBytes(), 0, false);
			System.out.print("Publish in T2: "+str+"\n");
		} catch (MqttException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
}
else if(arr[i][1]==0)
{
	TimeUnit.MILLISECONDS.sleep((long) num2);
}

 if(arr[i][2]==1)
{
	TimeUnit.MILLISECONDS.sleep((long) num3);
	String str="message3";
		String topic="T3";
		// TODO Auto-generated method stub
		try {
			client.publish(topic, str.getBytes(), 0, false);
			System.out.print("Publish in T3: "+str+"\n");
		} catch (MqttException e) {
```

```
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
        }
        else if(arr[i][2]==0)
        {
                TimeUnit.MILLISECONDS.sleep((long) num3);
        }

        if(arr[i][3]==1)
        {
                TimeUnit.MILLISECONDS.sleep((long) num4);
                String str="message4";
                        String topic="T4";
                        // TODO Auto-generated method stub
                        try {
                                client.publish(topic, str.getBytes(), 0, false);
                                System.out.print("Publish in T4: "+str+"\n");
                        } catch (MqttException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        }
        }
        else if(arr[i][3]==0)
        {
                TimeUnit.MILLISECONDS.sleep((long) num4);
        }
   }
}
```

```
    else
    {
      System.out.print("MQTT service connect failed");
    }
  } catch (MqttException e) {
    e.printStackTrace();
  }
}

static String strToBinary(String s)
{
      byte[] bytes = s.getBytes();
    StringBuilder binary = new StringBuilder();
    for (byte b : bytes)
    {
       int val = b;
       for (int i = 0; i < 8; i++)
       {
          binary.append((val & 128) == 0 ? 0 : 1);
          val <<= 1;
       }
      // binary.append(' ');
    }
    return binary.toString();
}

static String byteToBinary(byte[] bytes)
{
```

```
    StringBuilder binary = new StringBuilder();
    for (byte b : bytes)
    {
       int val = b;
       for (int i = 0; i < 8; i++)
       {
          binary.append((val & 128) == 0 ? 0 : 1);
          val <<= 1;
       }
      // binary.append(' ');
    }
    return binary.toString();
  }
}
```

Прилог 2

Кодови напишани во Јава за објавување на пораки без имплементиран скриен канал и објавување на пораки во ASCII-формат со имплементиран скриен канал (I4) кај MQTT-верзија 5.0

Код за објавувач без имплементиран скриен канал (Испраќање на податоци на 2 теми во секунда , односно 2 теми x 2 бита = 4 бита/секунда):

```
import java.util.Random;

import java.util.UUID;

import java.util.concurrent.TimeUnit;

import com.hivemq.client.mqtt.datatypes.MqttQos;

import com.hivemq.client.mqtt.mqtt5.Mqtt5BlockingClient;

import com.hivemq.client.mqtt.mqtt5.Mqtt5Client;


public class PahoDemo {
  int arr[][]= new int[608][4];
  //int arr_time[][] = new int[1216][2];
  int arr1[][]= new int[608][2];
  int i,j, count1;
  double num1, num2, num3, num4, s;
  public PahoDemo() {}


  public static void main(String[] args) throws Exception {
    new PahoDemo().doDemo();
  }
  public void doDemo() throws Exception {
        Mqtt5Client client1 = Mqtt5Client.builder()
                .identifier(UUID.randomUUID().toString())
                .serverHost("18.211.71.125")
                .serverPort(1883)
                .buildBlocking();
```

```
   Mqtt5BlockingClient client = client1.toBlocking();


      client.connect();


     //client.publishWith().topic("test/topic").qos(MqttQos.AT_LEAST_ONCE).paylo
  ad("1".getBytes()).send();

  if(client.getState().isConnected())

  {


    int count=0;


    for(int i=0;i<608;i++)

    {

        for(int j=0;j<4;j++)

        {

                Random rand=new Random();

                int k=rand.nextInt();

                arr[i][j]=Math.abs(k)%2;

                System.out.print(arr[i][j]);


        }

        System.out.print("\n");

    }

System.out.println("----------------------------------");


    int co1=0;

    for(int i=0;i<608;i++)

    {

        for(int j=0;j<2;j++)
```

```
    {
            Random rand=new Random();
            int k=rand.nextInt();
            arr1[i][j]=Math.abs(k)%2;;
            System.out.print(arr1[i][j]);
    }
    System.out.print("\n");
}
  System.out.println("----------------------------------");

for(i=0;i<608;i++)
{
    Random rnd=new Random();
    num1=rnd.nextDouble();
    num2=rnd.nextDouble();
      s=(num1+num2)/1000;
    num1 /=s;
    num2 /=s;
    System.out.print("num1="+num1+" num2="+num2 +"\n");
    if(arr[i][0]==0 && arr[i][1]==0)
    {
            if(arr1[i][0] == 1)
            {
            TimeUnit.MILLISECONDS.sleep((long) num1);
            String str="message0";
                    String topic="T0";
                    // TODO Auto-generated method stub
```

```
   client.publishWith().topic(topic).qos(MqttQos.AT_LEAST_ONCE).payload(str.
getBytes()).send();
                        System.out.print("Publish in T0: "+str+"\n");
            }
                        else
                        {
                          TimeUnit.MILLISECONDS.sleep((long) num1);
                        }

     }
     else if(arr[i][0]==0 && arr[i][1]==1)
     {

            if(arr1[i][0] == 1)
            {
            TimeUnit.MILLISECONDS.sleep((long) num1);
            String str="message1";
                  String topic="T1";
                  // TODO Auto-generated method stub


   client.publishWith().topic(topic).qos(MqttQos.AT_LEAST_ONCE).payload(str.
getBytes()).send();
                        System.out.print("Publish in T1: "+str+"\n");
            }
            else
                        {
                          TimeUnit.MILLISECONDS.sleep((long) num1);
                        }
     }
```

```
else if(arr[i][0]==1 && arr[i][1]==0)
{
        if(arr1[i][0] == 1)
        {
        TimeUnit.MILLISECONDS.sleep((long) num1);
        String str="message2";
                String topic="T2";
                // TODO Auto-generated method stub


   client.publishWith().topic(topic).qos(MqttQos.AT_LEAST_ONCE).payload(str.
getBytes()).send();
                        System.out.print("Publish in T2: "+str+"\n");
        }
        else
                        {
                          TimeUnit.MILLISECONDS.sleep((long) num1);
                        }

}

else if(arr[i][0]==1 && arr[i][1]==1)
{

        if(arr1[i][0] == 1)
        {
        TimeUnit.MILLISECONDS.sleep((long) num1);
        String str="message3";
                String topic="T3";
```

```
                    // TODO Auto-generated method stub


   client.publishWith().topic(topic).qos(MqttQos.AT_LEAST_ONCE).payload(str.
getBytes()).send();
                            System.out.print("Publish in T3: "+str+"\n");
            }
            else
                            {
                              TimeUnit.MILLISECONDS.sleep((long) num1);
                            }
     }


     if(arr[i][2]==0 && arr[i][3]==0)
     {
            if(arr1[i][1] == 1)
            {
            TimeUnit.MILLISECONDS.sleep((long) num2);
            String str="message0";
                  String topic="T0";
                  // TODO Auto-generated method stub


   client.publishWith().topic(topic).qos(MqttQos.AT_LEAST_ONCE).payload(str.
getBytes()).send();
                            System.out.print("Publish in T0: "+str+"\n");
            }
            else
                            {
                              TimeUnit.MILLISECONDS.sleep((long) num2);
                            }
     }
```

```
    else if(arr[i][2]==0 && arr[i][3]==1)
    {
            if(arr1[i][1] == 1)
            {
            TimeUnit.MILLISECONDS.sleep((long) num2);
            String str="message1";
                    String topic="T1";
                    // TODO Auto-generated method stub

   client.publishWith().topic(topic).qos(MqttQos.AT_LEAST_ONCE).payload(str.
getBytes()).send();
                            System.out.print("Publish in T1: "+str+"\n");
            }
            else
                            {
                              TimeUnit.MILLISECONDS.sleep((long) num2);
                            }


    }
    else if(arr[i][2]==1 && arr[i][3]==0)
    {
            if(arr1[i][1] == 1)
            {
            TimeUnit.MILLISECONDS.sleep((long) num2);
            String str="message2";
                    String topic="T2";
                    // TODO Auto-generated method stub

   client.publishWith().topic(topic).qos(MqttQos.AT_LEAST_ONCE).payload(str.
getBytes()).send();
                            System.out.print("Publish in T2: "+str+"\n");
```

```
                }
                else
                                {
                                  TimeUnit.MILLISECONDS.sleep((long) num2);
                                }
        }
        else if(arr[i][2]==1 && arr[i][3]==1)
        {
                if(arr1[i][1] == 1)
                {
                TimeUnit.MILLISECONDS.sleep((long) num2);
                String str="message3";
                        String topic="T3";
                        // TODO Auto-generated method stub

    client.publishWith().topic(topic).qos(MqttQos.AT_LEAST_ONCE).payload(str.
getBytes()).send();
                                System.out.print("Publish in T3: "+str+"\n");
                }
                else
                                {
                                  TimeUnit.MILLISECONDS.sleep((long) num2);
                                }
        }
  }
}
else
{
  System.out.print("MQTT service connect failed");
}
```

```
  }
}
```

Код за објавувач со имплементиран скриен канал за испраќање на порака во ASCII-формат (Испраќање на податоци на 2 теми во секунда, односно 2 теми x 2 бита = 4 бита/секунда):

```
import java.util.Random;
import java.util.UUID;
import java.util.concurrent.TimeUnit;
import com.hivemq.client.mqtt.datatypes.MqttQos;
import com.hivemq.client.mqtt.mqtt5.Mqtt5BlockingClient;
import com.hivemq.client.mqtt.mqtt5.Mqtt5Client;

public class PahoDemo {
  int arr[][]= new int[608][4];
  int arr1[][]= new int[1216][2];
  int i,j, count1;
  double num1, num2, num3, num4, s;
  public PahoDemo() {}
  public static void main(String[] args) throws Exception {
    new PahoDemo().doDemo();
  }
  public void doDemo() throws Exception {
          Mqtt5Client client1 = Mqtt5Client.builder()
                  .identifier(UUID.randomUUID().toString())
                  .serverHost("18.211.71.125")
                  .serverPort(1883)
                  .buildBlocking();
          Mqtt5BlockingClient client = client1.toBlocking();
```

```
            client.connect();

        //client.publishWith().topic("test/topic").qos(MqttQos.AT_LEAST_ONCE).payload("1".getBytes()).send();
    if(client.getState().isConnected())
    {
        String str1="WHEN you are old and grey and full of sleep,\r\n" +
                    "And nodding by the fire take down this book\r\n" +
                    "And slowly read,and dream of the soft look\r\n" +
                    "Your eyes had once,and of their shadows deep\r\n" +
                    "How many loved your moments of glad grace,\r\n" +
                    "And loved your beauty with love false or true,\r\n" +
                    "But one man loved with a love\r\n" +
                    "";
        String str_bin=strToBinary(str1);
        int[] str_in=new int[str_bin.length()];

       for(int i=0;i<str_bin.length();i++)
                    str_in[i]=Character.digit(str_bin.charAt(i),10);
        int count=0;
        for(int i=0;i<608;i++)
        {
            for(int j=0;j<4;j++)
            {
                arr[i][j]=str_in[count];
                count++;
            }
        }
        int co1=0;
        for(int i=0;i<1216;i++)
```

```
{
        for(int j=0;j<2;j++)
        {
                arr1[i][j]=str_in[co1];
                System.out.print(arr1[i][j]);
                co1++;
        }
        System.out.print("\n");
}
for(i=0;i<608;i++)
{
        Random rnd=new Random();
        num1=rnd.nextDouble();
        num2=rnd.nextDouble();
        s=(num1+num2)/1000;

        num1 /=s;
        num2 /=s;
        System.out.print("num1="+num1+" num2="+num2 +"\n");
        if(arr[i][0]==0 && arr[i][1]==0)
        {
                TimeUnit.MILLISECONDS.sleep((long) num1);
                String str="message0";
                        String topic="T0";
                        // TODO Auto-generated method stub

    client.publishWith().topic(topic).qos(MqttQos.AT_LEAST_ONCE).payload(str.
getBytes()).send();
                                System.out.print("Publish in T0: "+str+"\n");
        }
```

```
                else if(arr[i][0]==0 && arr[i][1]==1)
                {
                        TimeUnit.MILLISECONDS.sleep((long) num1);
                        String str="message1";
                                String topic="T1";
                                // TODO Auto-generated method stub

        client.publishWith().topic(topic).qos(MqttQos.AT_LEAST_ONCE).payload(str.
getBytes()).send();
                                        System.out.print("Publish in T1: "+str+"\n");
                }

                else if(arr[i][0]==1 && arr[i][1]==0)
                {
                        TimeUnit.MILLISECONDS.sleep((long) num1);
                        String str="message2";
                                String topic="T2";
                                // TODO Auto-generated method stub

        client.publishWith().topic(topic).qos(MqttQos.AT_LEAST_ONCE).payload(str.
getBytes()).send();
                                        System.out.print("Publish in T2: "+str+"\n");
                }
                else if(arr[i][0]==1 && arr[i][1]==1)
                {
                        TimeUnit.MILLISECONDS.sleep((long) num1);
                        String str="message3";
                                String topic="T3";
                                // TODO Auto-generated method stub

        client.publishWith().topic(topic).qos(MqttQos.AT_LEAST_ONCE).payload(str.
getBytes()).send();
```

```
                                System.out.print("Publish in T3: "+str+"\n");
                }
                if(arr[i][2]==0 && arr[i][3]==0)
                {
                        TimeUnit.MILLISECONDS.sleep((long) num2);
                        String str="message0";
                                String topic="T0";
                                // TODO Auto-generated method stub

        client.publishWith().topic(topic).qos(MqttQos.AT_LEAST_ONCE).payload(str.
getBytes()).send();
                                System.out.print("Publish in T0: "+str+"\n");
                }
                else if(arr[i][2]==0 && arr[i][3]==1)
                {
                        TimeUnit.MILLISECONDS.sleep((long) num2);
                        String str="message1";
                                String topic="T1";
                                // TODO Auto-generated method stub

        client.publishWith().topic(topic).qos(MqttQos.AT_LEAST_ONCE).payload(str.
getBytes()).send();
                                System.out.print("Publish in T1: "+str+"\n");
                }
                else if(arr[i][2]==1 && arr[i][3]==0)
                {
                        //Timer timer=new Timer();
                        //timer.schedule(new Timer1(), 10000);
                        TimeUnit.MILLISECONDS.sleep((long) num2);
                        String str="message2";
                                String topic="T2";
```

```
                    // TODO Auto-generated method stub

        client.publishWith().topic(topic).qos(MqttQos.AT_LEAST_ONCE).payload(str.
getBytes()).send();
                                System.out.print("Publish in T2: "+str+"\n");
                }
                else if(arr[i][2]==1 && arr[i][3]==1)
                {
                        TimeUnit.MILLISECONDS.sleep((long) num2);
                        String str="message3";
                            String topic="T3";
                       // TODO Auto-generated method stub

        client.publishWith().topic(topic).qos(MqttQos.AT_LEAST_ONCE).payload(str.
getBytes()).send();
                                System.out.print("Publish in T3: "+str+"\n");
                }
           }
    }
    else
    {
        System.out.print("MQTT service connect failed");
    }
 }


 static String strToBinary(String s)
 {
        byte[] bytes = s.getBytes();
    StringBuilder binary = new StringBuilder();
    for (byte b : bytes)
    {
```

```
        int val = b;
        for (int i = 0; i < 8; i++)
        {
           binary.append((val & 128) == 0 ? 0 : 1);
           val <<= 1;
        }
       // binary.append(' ');
      }
      return binary.toString();
  }
}
```