Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

Comprehensive analysis of MQTT 5.0 susceptibility to network covert channels

Aleksandra Mileva[a], *Aleksandar Velinov*[b], *Laura Hartmann*[c], *Steffen Wendzel*[c], *Wojciech Mazurczyk*[d,*]

[a] *Department of Computer Engineering and Intelligent Systems, Faculty of Computer Science, University Goce Delcev, Štip 2000, Macedonia*

[b] *Department of Information Technologies, Faculty of Computer Science, University Goce Delcev, Štip 2000, Macedonia*

[c] *Department of Computer Science, Worms University of Applied Sciences, Worms 67549, Germany*

[d] *Institute of Computer Science, Warsaw University of Technology, Warsaw 00-665, Poland*

ARTICLE INFO

Article history:
Received 22 May 2020
Revised 5 October 2020
Accepted 18 January 2021
Available online 22 January 2021

Keywords:
MQTT
Network covert channels
Network steganography
IoT steganography
Information hiding
Network security

ABSTRACT

Message Queuing Telemetry Transport (MQTT) is a publish-subscribe protocol which is currently popular in Internet of Things (IoT) applications. Recently its 5.0 version has been introduced and ensuring that it is capable of providing services in a secure manner is of great importance. It must be noted that holistic security analysis should also evaluate protocol's susceptibility to network covert channels. That is why in this paper we present a systematic overview of potential data hiding techniques that can be applied to MQTT 5.0. We are especially focusing on network covert channels that, in order to exchange secrets, exploit characteristic features of this MQTT version. Finally, we develop proof-of-concept implementations of the chosen data hiding techniques and conduct their performance evaluation in order to assess their feasibility in practical setups.

1. Introduction

Network covert channels (CCs) are a subdiscipline of the information hiding research area which focus on investigating techniques capable to provide hidden data transfers over communication networks and their countermeasures. Many CCs have been studied for communication protocols within past three decades (Mazurczyk et al., 2016; Mileva and Panajotov, 2014; Zander et al., 2007) and it must be emphasized that such techniques are increasingly used for nefarious purposes by cyber criminals and malware developers (Cabaj et al., 2018). Note that from the security point of view the identification of new data hiding methods as well as showing how they can be detected/eliminated should be treated the same way as disclosing and patching a previously unknown vulnerability in software or hardware systems. Therefore, in the literature several existing and modern protocols have been subjected to network covert channels susceptibility analysis, e.g., Fraczek et al. (2012); Mazurczyk and Szczypiorski (2008); Velinov et al. (2019).

Message Queuing Telemetry Transport (MQTT) is a lightweight, client-server message transport protocol which is especially suitable for the machine-to-machine (M2M)/IoT connectivity. It is suitable for resource-constrained devices,

* Corresponding author.
E-mail address: wojciech.mazurczyk@pw.edu.pl (W. Mazurczyk).

https://doi.org/10.1016/j.cose.2021.102207

for low-bandwidth and high-latency environments, enterprise backends to mobile communications, and push communications. Additionally, it provides reliable communication over unreliable networks. It is binary protocol with minimal overhead.

Moreover, MQTT relies on a publish/subscribe communication model. Basically, in such a setup two types of clients exist. The *publishers* are clients that generate messages, while *subscribers* are their recipients. Note that these two types of clients never communicate directly with each other. In order to exchange information, they utilize a central point that plays the role of a server and is called a *broker*. Its responsibility is to receive the messages sent by the publishers and to forward them to the subscribers.

MQTT was originally invented in 1999, for oil pipeline monitoring through the desert, and sending data via satellite links. From this time it has been heavily used in various applications, e.g., in Facebook Messenger, Amazon IoT (a part of the Amazon Web Services), OpenStack, Microsoft Azure IoT Hub, many home automation solutions (e.g., Home Assistant, MajorDoMo, Mycontroller), just to mention a few. From the standard development perspective, the MQTT version 3.1.1 (Banks and Gupta, 2015b) became an OASIS standard in 2014 and ISO/IEC standard 20922:2016 in 2016. Then, in March 2019, the newest MQTT version, i.e., 5.0, was released and it became an OASIS standard, too (Banks et al., 2019b). Note that in principle v3.1.1 and v5.0 are not compatible with each other as the newer version has a slightly different format of messages and richer functionality when compared to v3.1.1. On the other hand, some vendors like HiveMQ provide a compatibility layer in order to ensure both versions can co-exist.[1]

Currently, the wide deployment of MQTT protocol has become another security problem. The Shadowserver Foundation, as part of the EU CEF (Connecting Europe Facility) funded project VARIoT (Vulnerability and Attack Repository for IoT), is performing IPv4 scanning on a daily basis for publicly accessible MQTT broker services enabled on port 1883/TCP. They reported that on 12th March 2020, out of 71,508 IP addresses that replied to the sent probes 41,558 broker instances allowed for anonymous access (Report, 2020). Moreover, another research performed by Avast in 2018 (Hron, 2018) showed how such unprotected MQTT servers can be used for tracking their owners in real time, for reading all the published messages, or publishing fabricated messages in some MQTT broker's topics. This proves that an outsider with an Internet access is able, for example, to obtain various sensitive information thus abusing privacy of the home residents or to seize control over somebody's smart home and all connected devices there. Even more, one can use these unprotected MQTT servers to enable covert communications for malicious purposes, e.g., to organize Command & Control (C&C) channels helping infected devices to communicate with an attacker (Velinov et al., 2019). Additionally, Mao et al. (2020) demonstrated that most of the known IoT cloud platforms that support and use MQTT have security problems, especially in the scenario of device sharing (e.g., smart locks shared among users, like hotel dwellers, Airbnb apartment renters, home visitors, etc.) and revocation, where, for example, a malicious ex-user can retain full control of the device on which his access privilege has expired.

As currently, MQTT v5.0 protocol is foreseen to take over the market soon, thus it is important now to conduct its susceptibility to network covert channels in order to have time to find and develop proper countermeasures. Note that in our previous work (Velinov et al., 2019) we have performed such analysis for MQTT v.3.1.1, however, as already mentioned due to different format of messages and additional functionality, we believe that similar investigation need to be performed also for v5.0.

That is why in this paper we investigate the susceptibility of the latest MQTT version to the previously known and novel network covert channels. From this perspective, we make the following key contributions:

- We propose 18 direct (i.e., where clandestine communication parties must be active at the same time) and five indirect (i.e., where the covert sender and the covert receiver communicate indirectly through an innocent third party) covert channels for MQTT v5.0.
- We found an indirect covert channel that represents a new hiding (timing) pattern.
- We provide proof-of-concept implementations for two indirect covert channels and included their experimental evaluation using three covert channel metrics: bandwidth, undetectability, and robustness.

The rest of the paper is structured as follows. Section 2 describes the related work in the area of network covert channels in IoT scenarios. Next, in Section 3, MQTT fundamentals are provided, including the format of control packets and new features present in v5.0. Section 4 is devoted to the new hiding pattern as well as the new direct and indirect covert channels and their properties, preceded with the view of the used system models and relations to the existing v3.1.1 covert channels. In Section 5 an experimental evaluation of two chosen indirect CCs is presented, as a proof of their feasibility and effectiveness. At the end, Section 6 concludes our work and presents potential future directions.

2. Related work

The problem of covert channels in IoT as a security and privacy threat has been recently recognized and it is starting to raise attention in the security community (Caviglione et al., 2018; Sikder et al., 2018; Wendzel et al., 2017). Despite this fact, such type of research is still not significantly explored. In the reminder of this section, we review the most relevant works in IoT-based information hiding. Moreover, we categorize the existing techniques according to the layer they operate in the well-known 3-layer IoT architecture (Lin et al., 2017) (see Fig. 1). To briefly recap, this architecture includes: *(i)* perception layer where various sensors may be used for gathering information about the surrounding environment; *(ii)* network layer which processes and transmits sensor data and enables connections to other smart things or other networking equipment; *(iii)* the application layer which defines numerous appli-

[1] https://www.hivemq.com/hivemq-4-whats-new/

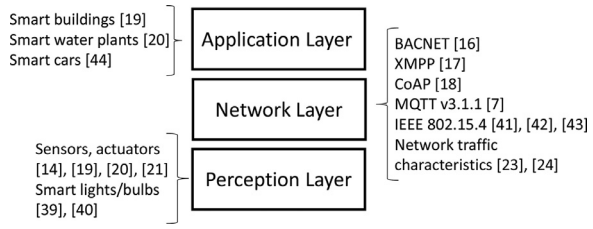

Fig. 1 – IoT 3-layer architecture (Lin et al., 2017).

cations in which the IoT can be deployed, e.g., smart homes, smart cities, etc.

For the perception layer, in Ulz et al. (2019) authors introduced three different sensor-based covert channels that provide a trade-off between the achievable covert channel bandwidth and undetectability. They present covert channels that require read- and write-access for sensor registers as well as a covert channel that transfers data by just triggering sensor readings so that the malicious behavior cannot be distinguished from typical, normal sensor usage.

Next, in Herzberg and Kfir (2019b) Herzberg and Kfir introduced a provably-covert channel for Cyber Physical Systems which relies on a corrupted actuator that is located in one zone and is able to send secrets to a sensor in a different zone, breaking the isolation. The same authors extended their work in Herzberg and Kfir (2019a) by exploring data hiding possibilities for indirect covert communication between a sensor and actuator via a benign threshold-based controller. The covert information was encoded within the output noise of the sensor in indistinguishable manner when compared to that of a benign sensor.

Ronen and Shamir (2016) use different brightness level in the smart lights as a covert LIFI communication system to exfiltrate data from a highly secure office building. Similarly, Cronin et al. (2019) suggested three different covert exfiltration techniques that deploy smart light bulbs, with modulation of their luminosity (by encoding signals as visibly undetectable brightness changes), color settings (by encoding signals as imperceptible color changes), and power utilization (by encoding signals as circuit current changes).

It is worth noting that currently published papers mostly deploy data hiding techniques related to the popular IoT protocols, i.e., the network layer. The earliest known approach for IoT-based covert channels was published for the Building Automation and Control Networks (BACnet) protocol in 2012 (Wendzel, 2012). Moreover, several storage covert channels in the Extensible Messaging and Presence Protocol (XMPP) (Patuck and Hernandez-Castro, 2013) and six storage and two timing covert channels in the Constrained Application Protocol (CoAP) (Mileva et al., 2018) were discovered. Next, in Velinov et al. (2019) the comprehensive analysis of the MQTT protocol has been performed from the information hiding perspective. In more detail, authors characterized seven direct and six indirect covert channels applicable for MQTT-based IoT environment, and for the selected data hiding methods their experimental evaluation has been presented. Several covert channels have also been suggested for the IEEE 802.15.4 standard (Martins and Guyennet, 2010; Mehta et al., 2008; Nain and Rajalakshmi, 2016), too.

Next, in Tan et al. (2018) various kinds of covert timing channels were analyzed to investigate their feasibility in the IoT environments. The inspected data hiding techniques included: packet-reordering-based, rate-switching-based, packet-loss-based, re-transmission-based, and scheduling-based covert timing channels.

Moskowitz et al. (2018) proposed a method for covert communication that utilizes transmission timing to obscure symbols. Authors also showed that IoT side channels are susceptible to network covert channels and that it is possible to create a data-in-motion data hiding technique without network protocol modifications.

At the application layer, Wendzel et al. (2017) have shown that one can hide data also outside of network protocols in cyber-physical systems (e.g., smart buildings), by slightly modifying some of its components, like sensors, controllers, actuators, etc., as well as by storing secret data in unused registers. On the other hand, in Herzberg and Kfir (2019b) Herzberg and Kfir introduced several covert channels for Cyber Physical Systems with special focus on smart water plants. Finally, Ying et al. (2019) introduced TACAN (Transmitter Authentication in Controller Area Network), a solution for the automotive CPSs that apply three different covert channels for Electronic Control Unit authentication: Inter-Arrival Time (IAT)-based, leveraging the IATs of CAN messages, offset-based, exploiting the clock offsets of CAN messages, and LSB-based concealing authentication messages into the LSBs of normal CAN data.

Taking into account above, it must be noted that the analysis of the MQTT-based information hiding susceptibility performed in Velinov et al. (2019) was conducted for version 3.1.1 which is currently the most popular MQTT variant. However, MQTT 5.0 is expected to be deployed widely in the near future. Thus, there is a pressing need to perform analogous analysis for the newer version of this protocol taking into account especially the characteristic novel functionalities that it introduces. This is what constitutes a novel contribution of this paper.

3. MQTT fundamentals

The publish-subscribe model that MQTT uses, differs from the traditional client-sever one which is currently most popularly utilized in the Internet and where the communication is performed directly between the endpoints. On contrary, the publish-subscribe model separates endpoints and introduces the central component that serves as a intermediate node and is known as a *broker*. The broker is responsible for storing session data of clients with persistent sessions but also for clients' authentication and authorization. There are two types of clients: *publishers* and *subscribers*. In principle, they may not be aware of each other. Publishers send messages to the broker that forwards them to subscribers (see Fig. 2). Note that the MQTT client can be any device (sensor device, smart light switch, controllers) that is connected to the broker.

MQTT clients always establish a connection with the broker using CONNECT message and they cannot communicate directly with other clients. The broker responds to this request

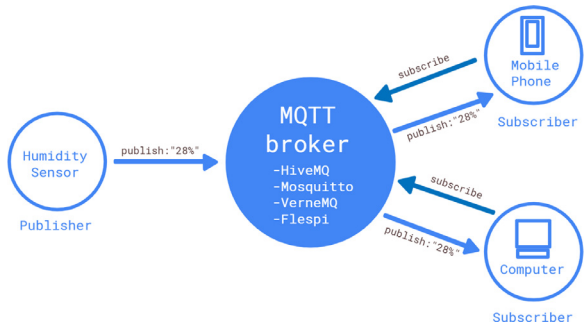

Fig. 2 – MQTT publish-subscribe model.

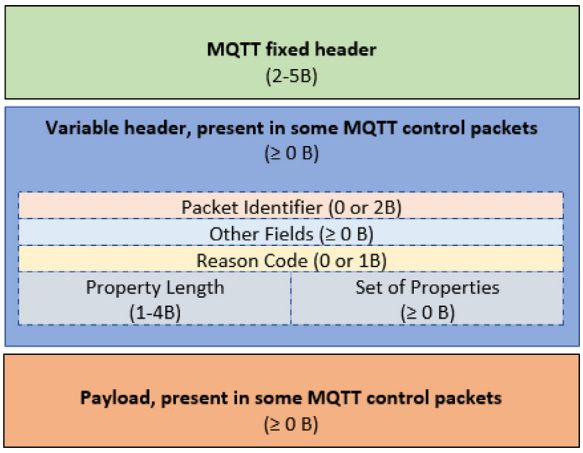

Fig. 3 – The format of the MQTT v5.0 control packet.

with CONNACK message and the return code that informs whether the connection is successful or not.

Once the connection is successfully established, clients are able to send messages or subscribe to them. To send a message, the client must specify the topic to which the message will be sent. The topic is hierarchically structured string that the broker uses to send messages to the clients who subscribed to a given topic (subscribers). Topics may have one or more topic levels that are separated by a forward slash character. An exemplary topic with 4 levels can take a form as follows:

home/LivingRoom/p1/humidity

Each topic must contain at least one character and its name is case-sensitive. In addition to the topic, the messages also have a payload which contains the data that needs to be send. The publisher can send data in a binary format, text data, JSON, or XML.

Clients are subscribing to topics of interest so that they can receive the messages sent on the chosen topics. To subscribe to a given topic, user sends a SUBSCRIBE message to the broker. Each SUBSCRIBE message contains two attributes: packet identifier and list of subscriptions. The UNSUBSCRIBE message is used to remove an existing subscription on the broker. MQTT clients use the DISCONNECT message to terminate the connection with the broker.

3.1. *MQTT v5.0 message format*

MQTT v5.0 uses 15 control packets, one more (i.e., AUTH control packet) than in MQTT v3.1.1, numbered from 1 to 15. Every control packet consists of a fixed header and an optional variable header and/or payload, with total packet size between 2B (e.g., PINGREQ packet) and 256MB.

Variable header has volatile structure in different control packets, which can be comprised from 2B Packet Identifier or other fields, single 1B Reason Code, and Properties part that consists of the mandatory Property Length and an optional set of Properties (Fig. 3). If the set of Properties is empty for the packets that need to have this part, the Property Length must be set to zero. Additionally, Will Properties field in the payload of CONNECT control packet can have a set of Properties and SUBACK/UNSUBACK packets can contain a list of Reason Codes in the payload.

One of the most important differences in the format of control packets between the two versions of the MQTT protocol is that 13 control packets in v5.0 (without PINGREQ and PINGRESP) contain a *Properties* part in the Variable header. Seven of them even contain a *Reason Code* part that was not present in the v3.1.1.

3.2. *New features in MQTT v5.0*

As already mentioned, although MQTT v5.0 inherited a lot from its predecessor, i.e., v3.1.1 both versions are, in principle, not compatible. This is caused by the different structure of messages as well as additional features that have been introduced in the new version.

The following features are new in the MQTT v5.0:

- *Improved session management* – with the optional session and message expiry intervals. In v5.0, persistent sessions can expire (in a vendor and broker independent way) and their state can be removed from the server side. The *Clean Session* flag in the CONNECT control packet is replaced by a *Clean Start* one which indicates that a new session is clear or persistent, and 32-bit *Session Expiry* interval which expresses how long (in seconds) to store the persistent session after a client disconnects itself. The message expiry interval is applied to online and queued (PUBLISH) messages. This means that an offline client with persistent session may not receive all of the messages when it reconnects, due to some of them expiring. Additionally, a server can also send DISCONNECT packets, redirect the client to another broker, or assign a client ID.
- *Improved error reporting* – with the optional *Reason Code* and *Reason String* on all ACKs. One benefit of such improvement is that DISCONNECT messages allow clients to determine why they were disconnected from the broker.
- *New extensibility mechanisms* – with a *Payload Format* indicator for the binary or UTF-8 encoded payload, an optional MIME-style *Content Type* for the payload and with an unlimited number of user properties, usually defined by the client applications. User properties allow to add data (up to 250 MB) in the header of control packets, and this data can be used to build custom protocols over MQTT.

- *Shared subscriptions* – clients can share the same subscription on the broker and the broker will deliver each message only to one chosen (per message) subscriber per group identifier (known also as client load balancing). In this way, the broker can distribute the message streams to multiple subscribers, which is good for upscaling and downscaling (on the fly) of clients.
- *Client restrictions* – resource-constrained clients can specify the maximum packet size they are prepared to receive, the maximum number of QoS (Quality of Service) 1 and QoS 2 messages that can be sent concurrently to them, and the *Will Delay* in which, if the client reconnects, it must not receive the *Will* message. In this way, clients can control their load and avoid overloads.
- *Server restrictions* – a server can define a set of features which it does not support.
- *Request/response mode of operation* – using the *Response Topic* in the PUBLISH control packet, together with a *Request-response* header and a *Correlation Data* header.
- *Enhanced authentication* – by providing a mechanism to enable challenge-response-style authentication including mutual authentication. This is achieved by two CONNECT headers, namely *Authentication Method* and *Authentication Data* and a new AUTH control packet.
- *Topic aliases* – utilization of small integers instead of topic names for reducing the size of the control packets.
- *Subscription ID* – a numeric subscription identifier can be specified in the SUBSCRIBE control packet and it can be returned within each message when it is delivered from the broker. This allows the client to determine which subscription or subscriptions caused the message to be delivered. Also, new subscription options are provided.

4. MQTT v5.0 covert channels

4.1. System model

In this paper we use the same system model as defined in our previous work (Velinov et al., 2019), i.e., with one covert sender (CS) and one or more covert receivers (CRs) and the same two submodels, i.e., Direct Covert Channels (DCC), where the broker can be the CS (DCCa) or the only CR (DCCb) and Indirect Covert Channels (ICC), where the broker is an intermediate node in the covert communication between the CS and CRs.

Furthermore, a second system model is used in this paper, in which the CS influences the network traffic through client duplication with the associated reconnection. To extract the secret message, the CRs passively analyze the network traffic and interpret it.

4.2. Relation to the existing MQTT v3.1.1 covert channels

Before presenting novel covert channels in MQTT 5.0, it must be noted that all direct and indirect covert channels found for the MQTT version 3.1.1 in our previous paper (Velinov et al., 2019) can be used for the new version as well. The summary of the applicability of these covert channels to MQTT 5.0 is presented in Table 1. Note, however, that in order to function properly in the new MQTT version, CCs require slight modifications in the following cases:

- *16-bit Client Identifier field (DCC.2)* – additionally, in v5.0 the server can set *Assigned Client Identifier* property in the CONNACK packet, so this CC can be considered as DCCa or DCCb for MQTT 5.0.
- *16-bit Keep Alive field (DCC.4)* – additionally, in v5.0 the server can set the *Server Keep Alive* field in the CONNACK packet, so this CC now belongs to both DCCa and DCCb types.
- *CC with persistent sessions (ICC.3)* – setting the *Clean Start* to 1 and *Session Expiry Interval* to 0 in the CONNECT control packet is equivalent in MQTT v3.1.1 of setting the *Clean Session* to 1 (CS creates a non-persistent session with the server). Modifying *Clean Start* to 0 and *Session Expiry Interval* to 0xFFFFFFFF is equivalent in MQTT v3.1.1 to *setting Clean Session* to 0 (CS creates a persistent session with the server). Another difference is when the server sends CONNACK packet to the CR, it must also set 0x00 (Success) Reason Code.

4.3. Hiding patterns

Hiding patterns are abstract descriptions of a covert channel's core concept. They were originally introduced in Wendzel et al. (2015) and were later extended by Mazurczyk et al. (2018, 2016); Velinov et al. (2019). Using such patterns, existing and new covert channel techniques can be categorized into one of the known hiding patterns. An overview of the latest state of hiding patterns is freely available under https://ih-patterns.blogspot.com/.

Hiding patterns are essentially split into two categories: *timing patterns* modulate different timing attributes of network traffic to signal hidden information while *storage patterns* modulate the transferred content of network traffic. Hiding patterns are described in a unique format (simplified PLML, see Wendzel et al., 2015) and form a taxonomy.

4.3.1. Timing patterns

While some timing patterns are protocol-agnostic (i.e., the content of transferred data is not relevant), others are protocol-aware, i.e., they require the understanding or functioning of transferred data. For instance, when inter-packet gaps between network packets are modulated (pattern PT1 *Inter-packet Times*), it does not matter which type of network protocol is transferred or what the semantics of the traffic are. However, if hidden data is signaled by retransmitting selected frames (pattern PT12 *Retransmission*), then the structure of these frames and, e.g., a sequence number or identifier field must be evaluated to determine *which* packet was retransmitted.

4.3.2. Storage patterns

Store patterns, on the other hand, either modify payload attributes or they modify non-payload attributes, such as protocol headers or padding fields. Hiding patterns that *modify non-payload* either alter the structure of transferred packets (e.g., by modulating the size of packets – pattern PS1 *Size Modulation*) or they preserve the structure of transmitted packets

Table 1 – Summary of applicability of existing MQTT v3.1.1 covert channels in MQTT v5.0.

CC	System sub - model	Description	MQTT v5.0	Comment
DCC.1	all	Direct or indirect CC by using PUBLISH: Application Message	Yes	As it is suggested
DCC.2	DCCb	Direct CC by using CONNECT: Client Identifier	Yes	Additionally, in v5.0 the server can set Assigned Client Identifier property in the CONNACK packet, so this CC now belongs to both DCCa and DCCb types
DCC.3	DCCb	Direct CC by using CONNECT: User Name and/or CONNECT: Password	Yes	As it is suggested
DCC.4	DCCb	Direct CC by using CONNECT: Keep Alive	Yes	Additionally, in v5.0 the server can set the *Server Keep Alive* field in the CONNACK packet, so this CC now belongs to both DCCa and DCCb types
DCC.5	DCCa, DCCb	Direct CC by using Packet Identifier in 11 Control Packets	Yes	As it is suggested
DCC.6	DCCa, DCCb	Direct CC by using PUBLISH: Topic Name	Yes	As it is suggested
DCC.7	DCCb	Direct CC by using SUBSCRIBE: Topic Filters or UNSUBSCRIBE: Topic Filters	Yes	As it is suggested
ICC.1	ICC	Indirect CC using PUBLISH: Topic Name and SUBSCRIBE: Topic Filters	Yes	As it is suggested
ICC.2	ICC	Indirect CC using topic ordering and updates presence/absence	Yes	As it is suggested
ICC.3	ICC	Indirect CC using persistent sessions	Yes	Setting *Clean Start* to 1 and *Session Expiry Interval* to 0 in the CONNECT control packet is equivalent in MQTT v3.1.1 to setting the *Clean Session* to 1 (CS creates a non-persistent session with the server). Setting *Clean Start* to 0 and *Session Expiry Interval* to 0xFFFFFFFF is equivalent in MQTT v3.1.1 to modifying *Clean Session* to 0 (CS creates a persistent session with the server). Another difference is when the server sends CONNACK packet to the CR, it must also set a 0x00 (Success) Reason Code.
ICC.4	ICC	Indirect CC using presence/absence of the Retained message	Yes	As it is suggested
ICC.5	ICC	Indirect CC using topic ordering and presence/absence of the Retained messages	Yes	As it is suggested
ICC.6	ICC	Indirect CC using information specific to the broker	Yes	As it is suggested

(e.g., by replacing some random value with an encrypted secret value – pattern PS10 *Random Value*). Regarding the storage patterns that *modify payload*, only those methods are included in the taxonomy that do not directly manipulate payload media content such as digital images or e-mail content – such methods would be part of digital media steganography instead of network steganography. Instead, there are two categories of payload-modifying hiding patterns: user-data aware and user-data agnostic patterns. *User-data aware* patterns require the understanding of transferred data (e.g., by compressing VoIP data with another codec than originally foreseen to create space for hidden data in a packet – pattern PS30 *Modify Redundancy*). *User-data agnostic* patterns ignore the meaning of transferred content, e.g., by increasing the size of payload (pattern PS20 – *Payload Field Size Modulation*) or by overwriting original content (pattern PS21 – *User-data Corruption*).

4.3.3. *Methodology*

In the remainder, we roughly follow the *unified description method* (Wendzel et al., 2016) for characterizing new covert channels on the basis of hiding patterns. The unified description method fosters a scientifically comparable representation of new hiding techniques and eases replication studies. Moreover, we describe a new pattern that we introduce for one of the indirect covert channels that we discovered in MQTT.

4.4. *New direct covert channels in MQTT v5.0*

4.4.1. *CCs with encoding secret data in fields*

MQTT v5.0 offers many new fields in the variable header or in the payload that can be utilized for direct CCs creation. We can divide these fields into two groups, according to the position of the field (in the variable header or in the payload). The first group comprises fields that can be found in the variable header of the control packets, denoted here as D1:

1. 32-bit **Message Expiry Interval** in the variable header of a PUBLISH packet or in the *Will Properties* field in the payload of a CONNECT packet. The value of this property is modified when a broker publishes a message and it contains the time that message has been waiting in the broker.
2. **Response Topic, Correlation Data** and **Content Type** in the variable header of a PUBLISH packet (each up to 65,535 bytes) or in the *Will Properties* field in the payload of a CONNECT packet.
3. 28-bit **Subscription Identifier** in the PUBLISH and SUBSCRIBE packets with the value from 1 till 268,435,455.
4. 16-bit **Topic Alias** in the PUBLISH packet.
5. 32-bit **Session Expiry Interval** in the CONNECT, CONNACK and DISCONNECT packets.
6. 16-bit **Receive Maximum** and **Topic Alias Maximum** in the CONNECT and CONNACK packets.
7. 32-bit **Maximum Packet Size** in the CONNECT and CONNACK packets.
8. 16-bit **Server Keep Alive** in the CONNACK packet.
9. **Assigned Client Identifier** in the CONNACK packet (up to 65,535 bytes).
10. **Authentication Data** in the CONNECT, CONNACK and AUTH packets (up to 65,535 bytes).
11. **Response information** in the CONNACK packet (up to 65,535 bytes).
12. **User Property** in 13 control packets (each pair of values up to $2 \cdot 65{,}535$ bytes). Only PINGREQ and PINGRESP control packets cannot contain User Property field.

Note that the D1 channels denoted as (2) and (12) can also be used as indirect covert channels.

The second group is comprised of the fields that can be found in the payload of control packets, denoted here as D2. It consists of the following fields:

1. **Response Topic, Correlation Data** and **Content Type** in the *Will Properties* field within the payload of a CONNECT packet (each up to 65,535 bytes).
2. 32-bit **Will Delay Interval** in the *Will Properties* field within the payload of a CONNECT packet.

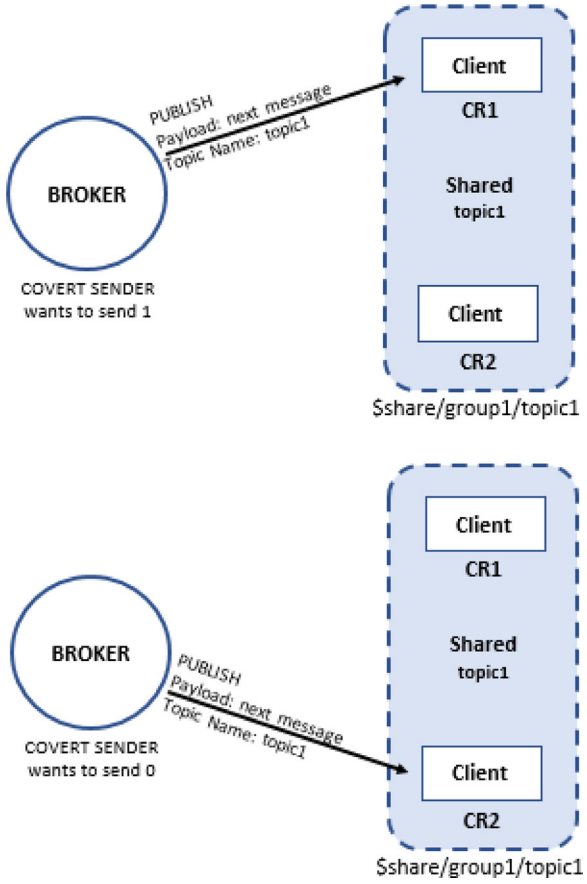

Fig. 4 – MQTT CC with the shared subscription.

Prerequisites: The only prerequisite is a client capable of exchanging messages with the broker.

Secret bits embedding and extraction: Secret bits can be directly embedded into the fields given above.

Information hiding pattern: Covert channels from the group D1 belong to the *PS10. Random value* pattern, while those from group D2 belong to the *PS31. User-data value modulation & reserved/unused* one.

4.4.2. *CC with shared subscription*

Shared subscriptions can be used in two different ways for creating covert channels with two collaborating subscribers or with only one subscriber. The binary unidirectional direct covert channel (D3a) between a broker (as a CS) and two collaborating subscribers (as CRs) can be realized in the following way.

Prerequisites Initially, two collaborating subscribers, CR1 and CR2, must have agreed upon some identifier of the shared subscription, with the specified ShareName and Topic Filter. For example, *$share/group1/topic1*. After that, they need to subscribe to this shared subscription.

Secret bits embedding and extraction If the broker as a CS wants to transmit secret bit '1', it sends the next message published in the topics specified by the Topic Filter to the CR1, and if it wants to transmit secret bit '0', it sends the message to the CR2 (Fig. 4). Because two subscribers collaborate, they always know which bit is sent.

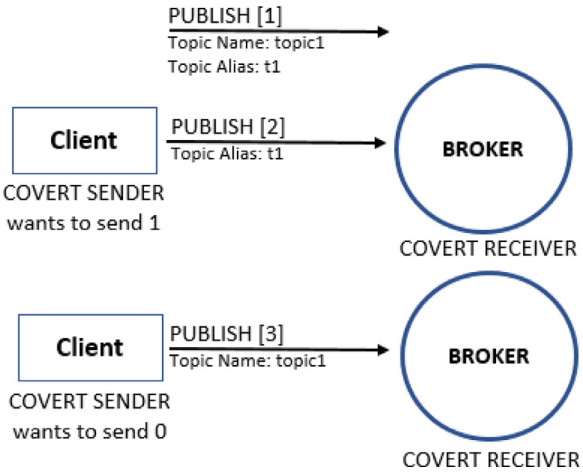

Fig. 5 – MQTT CC with topic alias.

This channel is a form of a distributed covert channel with multiple possible receivers (at least CR1 and CR2), i.e., it performs a host-based scattering (Mazurczyk et al., 2018).

Information hiding pattern PS31 (User-data Value Modulation), realized using a host-based scattering.

In the second version, the binary unidirectional direct covert channel (D3b) between a broker (as a CS) and one subscriber (as CR) can be realized in the following way.

Prerequisites In the beginning, after communication parties agreed upon the ShareName and Topic Filter of the shared subscription, the client CR must subscribe to that shared subscription with two different Client Identifiers, CID1 and CID2.

Secret bits embedding and extraction If the broker as a CS wants to transmit secret bit '1', it sends the next message published in the topics specified by the Topic Filter to the CID1, and if it wants to transmit secret bit '0', it sends the message to the CID2. The CR is aware of the identifiers and bits that they denote thus it is able to extract secret data.

Information hiding pattern PS31 (User-data Value Modulation).

4.4.3. *CC with topic alias*

A covert channel based on topic aliases can be created.

Prerequisites For this binary direct covert channel (D4) between a publisher (as a CS) and a broker (as a CR) or between a broker (as a CS) and one or more subscribers (as CRs), first the CS needs to send a PUBLISH packet with a non-zero length Topic Name and a Topic Alias, by which the receiver will set the specified Topic Alias mapping to that Topic Name (Fig. 5, step 1).

Secret bits embedding and extraction If the CS wants to transmit secret bit '1', it publishes the message by using Topic Alias and a zero length Topic Name for identifying the topic (Fig. 5, step 2), and if it wants to send secret bit '0', it publishes the message by using Topic Name for identifying the topic (Fig. 5, step 3).

Information hiding pattern The covert channel described above represents the *PS11. Value modulation* pattern.

4.4.4. *CC with session expiry interval*

One can create an unidirectional binary direct covert channel (D5) between a publisher (as a CS) and a broker (as a CR) using Session Expiry Interval in the CONNECT packet.

Prerequisites The CS should use some small amount of time for the Session Expiry Interval field and then it should disconnect.

Secret bits embedding and extraction If the CS reconnects before the expiration of the Session Expiry Interval on the broker's side, a binary '1' is sent to the broker. However, if the CS reconnects after the expiration of the Session Expiry Interval it denotes that the binary '0' has been transmitted. *Information hiding pattern* This is essentially a covert channel that requires the use of two different patterns. First, a "control channel" (simplified form of a covert channel internal control protocol) is realized by configuring both CS and broker with the Session Expiry Interval. This channel uses the PS31 pattern (*User-data Value Modulation and Reserved/Unused*) as 1 of *n* values for the Session Expiry Interval must be selected. Afterwards, pattern PT2 (*Message Timing*) is used, since the timing of the CS reconnecting to the broker determines the secret value.

4.4.5. *CC with the presence/absence of Reason code*

MQTT v5.0 provides improved error reporting by optional use of Reason Code and/or Reason String in all ACKs, AUTH and DISCONNECT packets. The normal Reason Code for success is 0. The CONNACK, DISCONNECT, PUBACK, PUBREC, PUBREL, PUBCOMP, and AUTH packets have a single Reason Code as part of the Variable Header, while the SUBACK and UNSUBACK packets contain a list of one or more Reason Codes in the payload. While for CONNACK the presence of 0 Reason Code for success is mandatory, for other control packets this is optional in the variable header. Absence of the Reason Code field is equivalent to Reason Code of 0. So, this duality can be used for creation of a new unidirectional binary covert channel (D6) from the broker (serving as CS) to the client (CR).

Prerequisites The best choice is to use ACKs for the PUBLISH control packet. This means that the client needs to use QoS 1 or QoS 2 for publishing a given message. First, the client publishes some arbitrary messages.

Secret bits embedding and extraction If the broker adds a Reason Code of 0 in ACKs (PUBACK packet for QoS 1, or PUBREC and PUBCOMP packets for QoS 2), a binary '1' is sent, while if the Reason Code is absent from the acknowledgement, a binary '0' is transmitted.

Information hiding pattern This covert channel represents the *PS11. Value modulation* pattern.

4.5. *New indirect covert channels in MQTT v5.0*

4.5.1. *CC with order of properties*

The bidirectional direct covert channel between a client and a broker, or indirect covert channel between a publisher and subscribers (I1) can be created by properties ordering. For the direct version, any control packet that contains the set of properties can be used, while for the indirect version, the logical choice is the PUBLISH control packet.

Prerequisites At the beginning, two collaborating parties must agree upon the order of a given set of properties

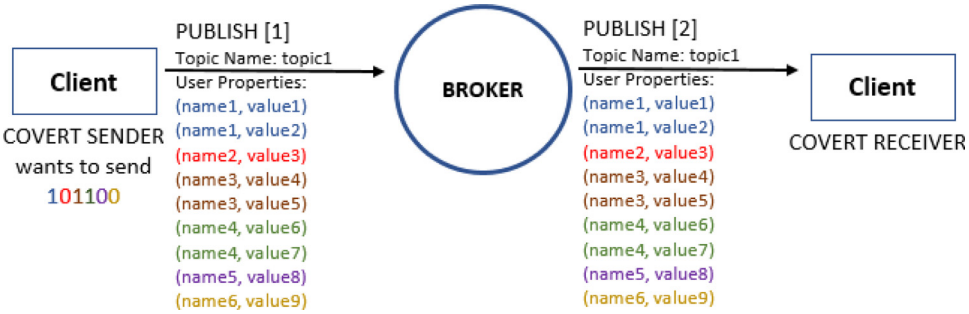

Fig. 6 – CC with user property duplication.

$(P_1, P_2, \ldots, P_p)$. For the PUBLISH packet, there are eight different properties that can be found in the variable header, from which six can be passed by the broker to the subscribers. Moreover, five of them are sent unaltered: Payload Format Indicator, Content Type, Response Topic, Correlation Data, and User Property, while one, i.e., Message Expiry Interval, is sent with the modified value. Additionally, the User Property can be sent multiple times, the only restriction is the total length of the control packet.

Secret bits embedding and extraction For binary channel, if the CS wants to transmit secret bit '1', it sends P_1 property before P_2, and for the secret bit '0', it transmits P_2 property before P_1 property. In result, using such ordering, one can send p bits per control packet.

Information hiding pattern According to the pattern-based classification, this covert channel is representing *PS2. Sequence Modulation* pattern (if the sequence is interpreted) as well as *PS2.a Position* pattern (if only the position of one element is interpreted).

4.5.2. *CC with user property duplication*

The indirect covert channel between a publisher and subscribers (I2) can be created by using (name, value) pairs multiple times with the same name in one PUBLISH packet. The broker must send all User Properties unaltered when forwarding a message to the subscribers.

Prerequisites Covert receivers need to be subscribed to the same topic in which the covert sender will publish the messages.

Secret bits embedding and extraction The CS encodes the secret data in the following way. For transmitting a binary '1', it will use two consecutive (name, value) pairs with the same name, and for sending binary '0', it will utilize only one (name, value) pair in the set of User Properties. This process is illustrated in Fig. 6.

Information hiding pattern This covert channel represents *PS3. Add Redundancy* pattern.

4.5.3. *CC with request/response pattern*

As already mentioned, MQTT v5.0 offers an imitation of the request-response behavior which is a basic rule for the HTTP client-server communication. The difference is that instead of direct client-server communication in MQTT we have an indirect communication via broker, with one of the clients serving as a server. This can be implemented in the MQTT protocol in the following way:

- One of the clients (C1) will have a role of the client and will be subscribed to some response topic (e.g., `/response/C1`). One can configure access control lists on the broker so that only C1 can subscribe to the given response topic.
- The other client (S) will have a role of the server. It could host a database, central logging service, etc. It will subscribe to some topic, e.g., `/topicS`.
- C1 can send a request message to the S by publishing in the topic `/topicS` and specifying its response topic in the PUBLISH control packet.
- S will then send a response message to C1, by publishing in the response topic `/response/C1`. Additionally, a Correlation Data can be used for connecting the response with the original request.

This feature can be used for creating an unidirectional indirect binary covert channel (I3) from the C1 to S.

Prerequisites C1 needs to subscribe to two different response topics RT1 and RT2, while S has to subscribe to its `/topicS`.

Secret bits embedding and extraction The C1 as a CS encodes the secret data in the following way. For sending a binary '1', it will publish in the topic `/topicS` with a specified response topic RT1, while for transmitting a binary '0' it will publish in the topic `/topicS` with the specified response topic RT2.

Information hiding pattern This covert channel represents the *PS11. Value modulation* pattern.

4.5.4. *CC with different topics*

One can create an n-bit bidirectional indirect covert channel (I4) that is applicable also to the older versions of MQTT (v3.1.1), in the following way:

Prerequisites For sending m bits, covert communication participants need to agree on 2^m different topics $T_0, T_1, \ldots, T_{2^m-1}$ that will correspond to the numbers $0, 1, \ldots, M$. All covert participants need to subscribe to all these topics.

Secret bits embedding and extraction The CS encodes the secret data in the following way. For sending the number M represented in a binary form with m bits, the CS publishes in the topic T_M. All CRs will obtain the message update from the topic T_M, so they can conclude the secret message M.

Information hiding pattern PS31. User-data Value Modulation and Reserved/Unused pattern (since one out of m possible values within the user-data is used to represent the hidden information).

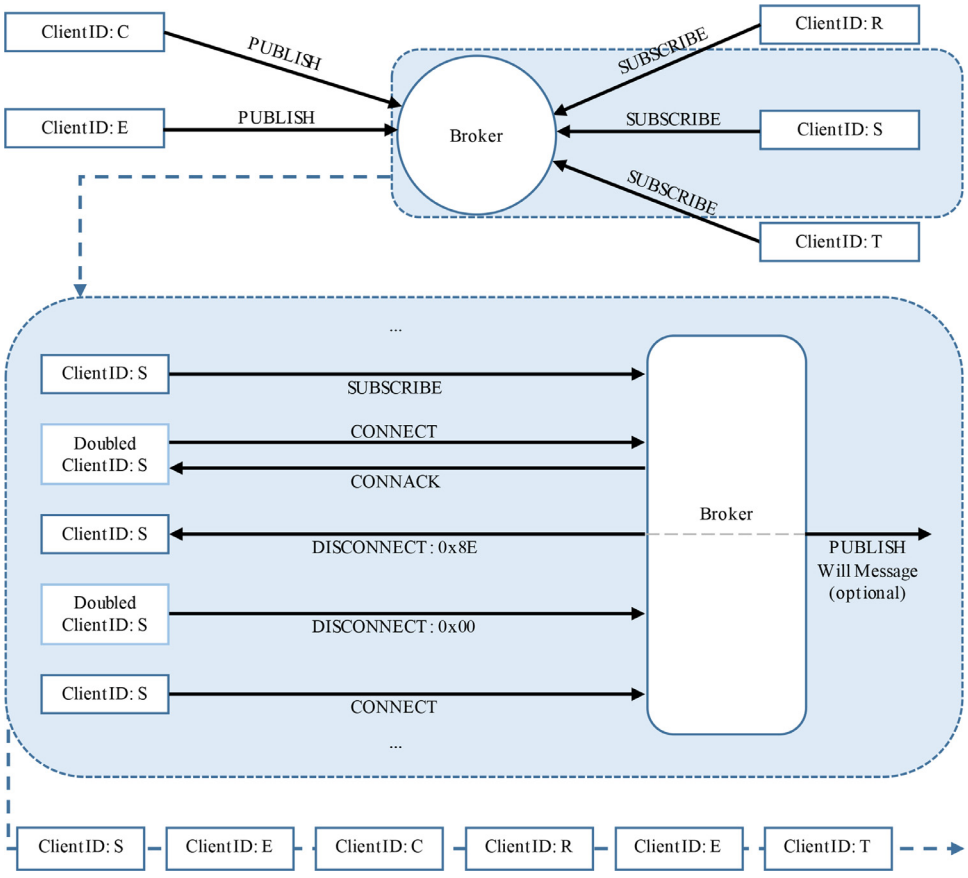

Fig. 7 – ICC.I5 with artificial reconnections.

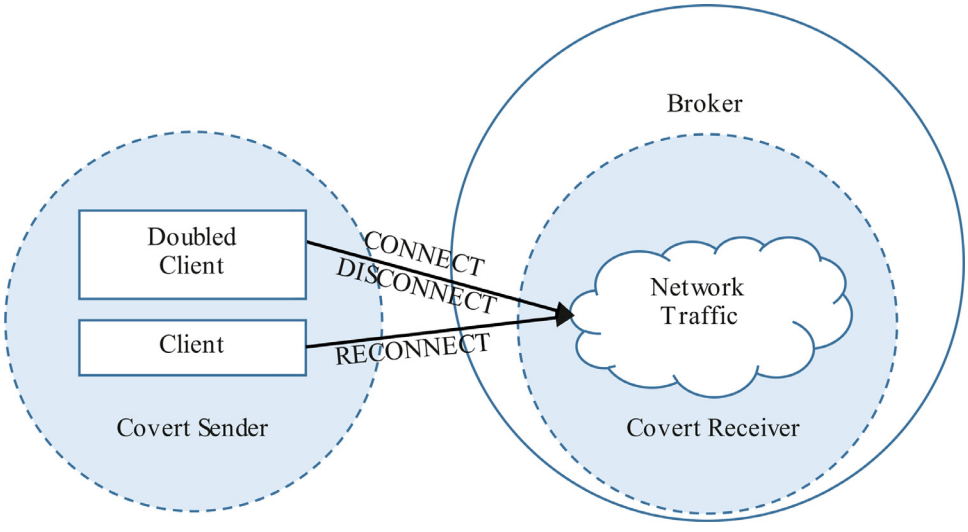

Fig. 8 – System model for I5 with artificial reconnections.

4.5.5. *Covert channel with artificial reconnections*

Another indirect unidirectional covert channel (I5) can be created using the reconnection action from a client where the covert sender duplicates another one's Client Identifier to force a reconnection between a client and a broker in a specific order (see Fig. 7). The covert receiver can read the message by eavesdropping the network traffic and decode it (see Fig. 8). For this covert channel, every client i represents a hidden symbol S_i.

Prerequisites For the usage of this covert channel, the broker cannot use authentication mechanism (which as mentioned in the Introduction is quite common right now). In case of

an open broker, two or more secret covert parties can connect to it. Another prerequisite is that the clients of the broker need to programmatically be configured with an (async) reconnect. For instance, Mosquitto, HiveMQ or Paho brokers are able to handle such automatic reconnections. Originally, they were configured to reconnect when a connection is half-open or lost. With this configuration, the client starts to reconnect to the broker, which returns a DISCONNECT to the first instance of the client's connection after opening the new session. The DISCONNECT implies the "Session taken over" Disconnect Reason Code 0x8E.

CS and CR agree on an encoding in advance, where every client i is linked to a secret symbol S_i.

Secret bits embedding and extraction If the covert sender connects and disconnects with an existing Client Identifier, it forces a reconnection of the original client. For each forced reconnection, all characters of the Client Identifier are sent to the broker, where the covert receiver monitors the network traffic and can interpret the secret message consisting of characters of the Client's Identifier. However, the Client Identifier itself does not represent the hidden message but allows to determine the client that just reconnected. Given a list of disconnects, the hidden message is composed by concatenation of the linked symbols in the order of their appearance (as this is a timing channel), i.e., $S_1||S_2||\ldots||S_n$.

Information hiding pattern According to the pattern-based classification, this is a *new* hiding pattern that we call *PT15. Artificial Reconnections*, see Section 4.6.

4.6. New hiding pattern PT15

Illustration The pattern used by CC I5, i.e., *PT15. Artificial Reconnections*, employs artificial (forced) reconnections to transfer secret messages. The covert sender influences connections of third-party nodes in a way that their connections to either a central element (e.g., an MQTT broker or a server) or a peer (in a peer-to-peer network) are terminated and then established again (i.e., a reconnect is performed). The covert receiver must be capable of monitoring these reconnects, e.g., either by compromising the central/peer element or in a passive network observer situation, like a MitM location. Encoding works by assigning secret values to third-party, so that a reconnect of a particular node represents the transfer of the secret symbol assigned to that node. Another scenario for this pattern can also be a chatroom or a gaming server with a large number of clients that reconnect automatically after being disconnected. The concept of this pattern is illustrated in Fig. 9.

Context In general, new patterns can be either derived from existing patterns (e.g., PS20 is derived from PS1) or, if they add entirely new ideas, they can be added to the particular position within the taxonomy. The process for adding new patterns is described in Wendzel et al. (2016). In our case, the new pattern cannot be derived from an existing pattern.

This pattern refers to Network Covert Timing Channels and the Protocol-aware branch because the functioning of a protocol's reconnects have to be understood and interpreted in a time-dependent manner. The final position of the pattern in the pattern-based taxonomy is shown in Fig. 10.

Implementation The implementation of this pattern can be performed in the following way. The covert sender is monitoring all nodes in a network/distributed system. It then duplicates the nodes in a chosen order to force reconnections. Alternatively, other methods may lead to forced reconnections – these can also be applied. The covert receiver monitors the network traffic and decodes the secret message sent.

Another approach for coding this channel would be to force only *one* client to reconnect initially, pause for a certain time interval, and then force a second reconnect. The transmitted character is then defined by the time that passed between the two reconnections. I.e., each time interval between two forced reconnections would represent a secret symbol. For instance, character 'x' might be encoded by reconnecting once, waiting for n seconds, and forcing another reconnect. Please note that this approach does *not* reflect the inter-packet times pattern, however, it is similar. In contrast to the inter-packet times pattern, the time between reconnects can contain multiple packets and the inter-packet gaps between these packets are not of direct interest here.

4.7. Bandwidth of the new covert channels

For estimating the maximal number of secret bits that can be transferred per second (the bandwidth) we will mainly rely on the maximal available number of secret bits per control packet and the number of control packets of particular type or types (given in the definition of the covert channel) sent per second. Summary of the results are provided in Table 2.

For D1.1, D1.5, D1.7,and D2.2, there is a maximum of 32 secret bits per control packet, thus, if there are n control packets of particular types per second, the resulting bandwidth will be $32 \cdot n$ bps.

D1.2 (D2.1) allows up to $3 \cdot 65,535B = 1,572,840$ bits per PUBLISH (CONNECT) control packet to be used for covert message. Thus, if there are n PUBLISH (CONNECT) packets per second, the resulting bandwidth will be $1,572,840 \cdot n$ bps.

The Subscription Identifier in D1.3 can have maximal value of 268,435,455 with its 28 bits encoded as Variable Byte Integer. Thus, if there are n control packets of particular types per second, the resulting bandwidth will be $28 \cdot n$ bps.

For D1.4 and D1.8 up to 16 secret bits per control packet are available, so if there are n control packets of particular types per second, the resulting bandwidth will be $16 \cdot n$ bps.

Similarly, for D1.6 up to $2 \cdot 16 = 32$ secret bits per control packet can be utilized, i.e., if there are n control packets of particular types per second, the resulting bandwidth will be $32 \cdot n$ bps.

Covert channels D1.9-D1.11, offer up to $65,535B = 524,280$ bits per control packet for hiding data, so if there are n control packets of particular types per second, the resulting bandwidth will be $524,280 \cdot n$ bps.

D1.12 is a little bit different from the rest of the CCs in D1 group, because there is an unspecified number of User Property pairs with the only limit of the maximal size of a control packet (256MB). Each User property (name, value) pair can have a maximal size up to $2 \cdot 65,535B$ or 1,048,560 bits. Thus, if there are u different User Property pairs per packet, and if there are n control packets of particular types per second, the resulting bandwidth will be $1,048,560 \cdot u \cdot n$ bps.

D3, D4 and I3 covert channels transfer 1 bit per PUBLISH control packet, so the resulting bandwidth will be n bps, if n

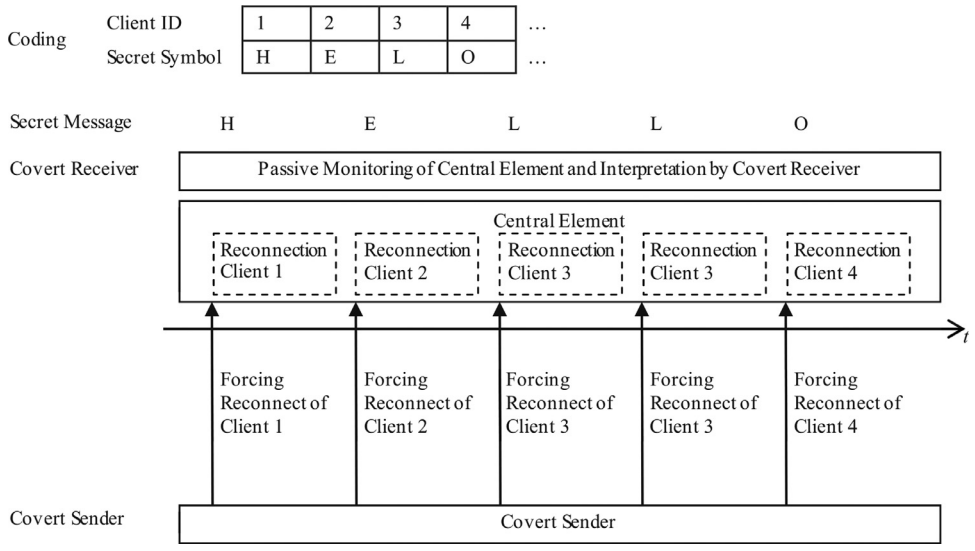

Fig. 9 – Generic illustration of the pattern *PT15. Artificial Reconnections*.

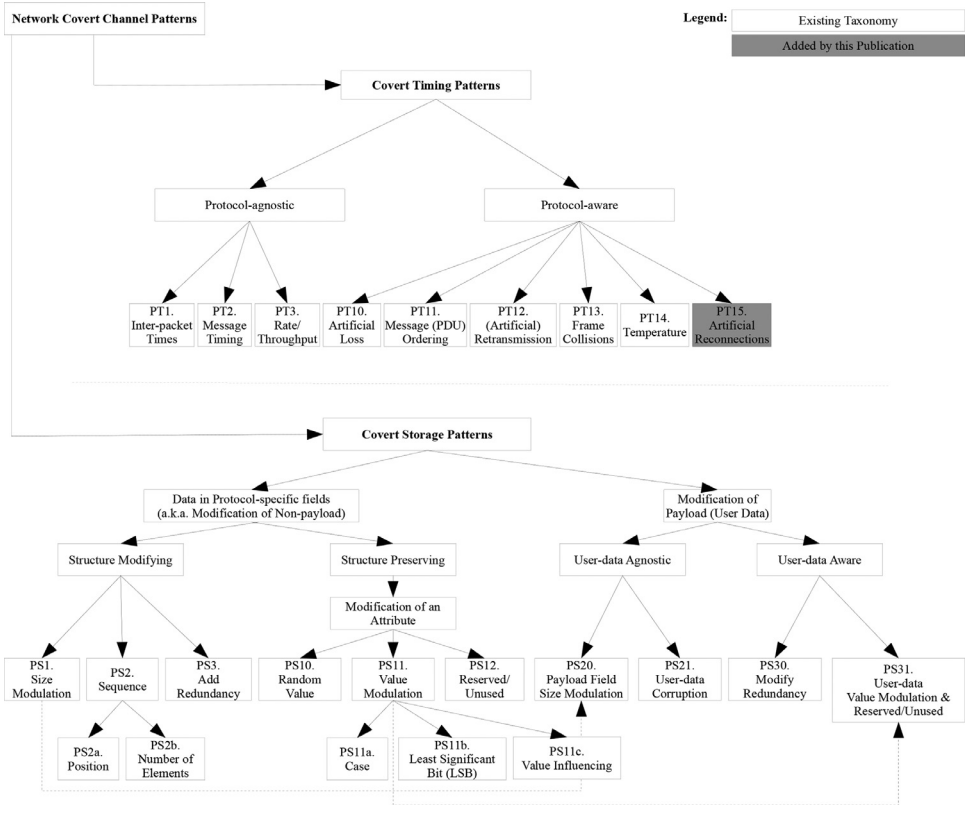

Fig. 10 – Extended version of the patterns taxonomy.

PUBLISH packets are sent per second. Similarly, the resulting bandwidth for D5 (D6) will be n bps if n CONNECT (ACKs) packets are sent each second.

For estimating the bandwidth of I1, we have p ordered properties per PUBLISH control packet. Therefore, the resulting bandwidth will be $p \cdot n$ bps in one direction, if n PUBLISH packets (in one direction) are sent each second.

For I2, if we assume that there are dp different (name, value) pairs in the set of PUBLISH User Properties, and if there are n PUBLISH packets (in one direction) sent per second, the estimated bandwidth will be $dp \cdot n$ bps per each direction.

If they are 2^m agreed different topics for I4 and if there are n PUBLISH packets (in one direction) sent per second, the estimated bandwidth for I4 will be $m \cdot n$ bps per direction.

Table 2 – Summary of the introduced covert channels.

CC	System sub - model	Pattern	Bandwidth (bps)	Type	MQTT v5.0 feature
D1.1	DCCa, DCCb	PS10	$32 \cdot n$	Unidirectional	Improved session management
D1.2	all	PS10	$1,572,840 \cdot n$	Unidirectional	Request/response mode of operation and new extensibility mechanisms
D1.3	DCCa, DCCb	PS10	$28 \cdot n$	Unidirectional	Subscription ID
D1.4	DCCa, DCCb	PS10	$16 \cdot n$	Unidirectional	Topic aliases
D1.5	DCCa, DCCb	PS10	$32 \cdot n$	Unidirectional	Improved session management
D1.6	DCCa, DCCb	PS10	$32 \cdot n$	Unidirectional	Client and/or server restrictions
D1.7	DCCa, DCCb	PS10	$32 \cdot n$	Unidirectional	Client and/or server restrictions
D1.8	DCCa	PS10	$16 \cdot n$	Unidirectional	Server restrictions
D1.9	DCCa	PS10	$524,280 \cdot n$	Unidirectional	Server restrictions
D1.10	DCCa, DCCb	PS10	$524,280 \cdot n$	Unidirectional	Enhanced authentication
D1.11	DCCa	PS10	$524,280 \cdot n$	Unidirectional	Request/response mode of operation
D1.12	all	PS10	$1,048,560 \cdot u \cdot n$	Unidirectional	New extensibility mechanisms
D2.1	DCCb	PS31	$1,572,840 \cdot n$	Unidirectional	Client restrictions
D2.2	DCCb	PS31	$32 \cdot n$	Unidirectional	Client restrictions
D3	DCCa	PS31	n	Unidirectional	Shared subscription
D4	DCCa, DCCb	PS11	n	Bidirectional	Topic aliases
D5	DCCb	PS31 and PT2	n	Unidirectional	Improved session management
D6	DCCa	PS11	n	Unidirectional	Improved error reporting
I1	all	PS2	$p \cdot n$	Bidirectional	New extensibility mechanisms
I2	ICC	PS3	$dp \cdot n$	Bidirectional	New extensibility mechanisms
I3	ICC	PS11	n	Unidirectional	Request/response mode of operation
I4	ICC	PS31	$m \cdot n$	Bidirectional	Protocol version independent
I5	ICC	PT15	$16 \cdot n$	Unidirectional	Protocol version independent

Covert channel I5 transfers one character (or 16 bits, because Unicode character set is used) per automatic reconnection, by sending CONNECT control packet for existing Client Identifier. So if there are n CONNECT control packets sent per second, the resulting bandwidth will be $16 \cdot n$ bps.

4.8. *Robustness of the new covert channels*

Robustness of the CC is the ability to protect the covert information from interferences introduced by the network, like different network conditions (e.g., packet losses, delay) or by third parties. For example, losses of packets carrying bits of the covert message in any of the newly introduced CCs, will negatively impact the secret message decoding, because some secret bits will be missing.

Because all newly CCs, except I5, are not timing channels, any delay introduced by the network will not affect them, until the delay is constant for all packets. But, when the delay rapidly changes over time, this can introduce the difference in the order of receiving the packets and the order of sending the packets. This means that the secret bits will be mixed up, and in the end, secret message would not be read by the CR. But this also can be prevented by introducing client-based sequential numbering of the packets carrying the secret bits. For the I5, any kind of the network delay in reconnection can impact negatively secret message decoding. Additionally, the robustness of this channel can be destroyed by sending CONNECT control packets from an adversary with the same Client Identifier as the CS, always before the expiration of the Session Expiry Interval on the broker's side.

The robustness of the direct CCs depends from the situation if the CS and CRs are on the same or on the different network. In the second case, an adversary can perform Man-in-the-Middle (MitM) attack, by impersonating himself as the broker or the client, or both, and by doing fabrication or modification of control packets.

Indirect covert channels I1, I2 and I4 depend on the PUBLISH packets for sending secret messages. One way to influence their robustness is to intentionally publish messages in the same topic(s) by any third party, because the broker does not identify the source of the message update, when sending it to the subscribers. Still, these CCs can resist to this attack by introducing some kind of client-based identification of the message source, for instance, by specific use of some User Property or some other Property. For example, the setting of User Property to start with some two (name, value) pairs, previously arranged by the participants in the clandestine communication, can be used for identification purposes. The same can be used for increasing undetectability by identifying the messages with secret bits in the pool of all messages sent from the CS. Another way is to deploy access control for a specific topic, so only selected clients can publish in that topic.

Because of the nature of the server's `/topicS` from the I3 CC, anybody is able to publish in it. So a third party can publish in the topic `/topicS` with a randomly chosen response topic from the set of two, `RT1` and `RT2`, and the same Client Identifier as the CS. In result, one can insert fabricated secret bits in I3. This is only possible, if the third party somehow learns the arranged response topics, the server topic and the Client Identifier of the covert sender.

One simple way for any third party to attack the robustness of the I5 CC, is to send CONNECT control packets with randomly chosen Client Identifiers from the list of currently used Client Identifiers. Even the MAC address of the CS can be spoofed if the CR searches for it. This will be interpreted as sending fabricated symbols in I5.

4.9. *Undetectability and potential countermeasures*

Undetectability is a metric for inability of third parties to distinguish covert traffic from a legitimate one. Some potential countermeasures can be used for detecting, limiting or preventing these new covert channels. These countermeasures can be applied only if the same type of channel is used more than once (or in more than one control packet).

Covert channels D1.1, D1.3, D1.4, D1.5, and D1.7 can be potentially detected by monitoring settings for a suspected client of the Message Expiry Intervals, Subscription Identifiers, Topic Aliases, Session Expiry Intervals, and Maximum Packet Sizes. The occurrences of these settings from the same Client Identifier or the same IP address or MAC address should be investigated. Moreover, one can monitor non-existing Subscription Identifiers in the PUBLISH control packets for detecting D1.3.

Also, D1.6, D1.8 and D2.2 can be potentially detected by investigating Receive Maximum, Topic Alias Maximum, Server Keep Alive, and Will Delay Interval values. Especially, their occurrences from the same Client Identifier, same IP or MAC address should be investigated.

Response Topic and Correlation Data from D1.2 and Response information from D1.11 are related to the Request/Response mode of operation, so if somebody is using them in this context, it can be a sign for potential CC utilization. Many different Response Topics from the same IP address can also be used as a potential detecting method. For D1.2, Content Type property in the PUBLISH packet normally is application-specific, thus, if someone is familiar with the application context, he can easily detect manipulation of this property. The same holds for the D2.1 covert channel.

D1.9 can be potentially detected by monitoring the assigned Client Identifier in the CONNACK packet and by seeking for many different values given to the same IP address or to the same MAC address.

The content of the Authentication Data field that is used in the covert channel D1.10 is defined by previously agreed authentication method. It can contain, for example, a context token if Kerberos (Authentication Method=”GS2-KRB5”) is used. These tokens usually are randomly generated, so this channel can be difficult to detect in this scenario. But in other cases, when its content is expected to be something non-random, any other value will rise a suspicion.

User Property presence is application-specific and normally it is used to provide means of transferring application layer name-value tags, thus, if someone is familiar with the application context, he can easily detect manipulation of this property and discover the presence of D1.12.

The second version of the D3, D3b, can be easily detected by counting the number of different Client Identifiers from the same IP or MAC addresses at the same broker, in the same time. However, the first version of the D3 channel, D3a, is almost impossible to detect. This is because there is no rule how the broker will choose the representative for the group with shared subscription to whom it will send a given message. The only way is to seek for collaborative communication between the clients of the group and to try to detect the potential CC utilization.

One possible countermeasure for detecting D4 CC is to record mappings between the Topic Names and a Topic Aliases in various communications in order to discover unusual behavior, e.g., publishing in a given topic sometimes by using its Topic Name and sometimes by using its Topic Alias.

One way to detect the D5, is to keep a record of Session Expiry Intervals in the CONNECT packet. The presence of many small values of Session Expiry Intervals can be a sign of this covert channel as small values are required to transfer binary zeros.

Covert channel D6 can be possibly detected by monitoring the presence/ absence of the Reason Code of 0 in ACKs. If there is a many ACKs with presence of the Reason Code of 0, and also many ACK without it in some period of time, this can be a sign of CC in use.

I1 channel can be detected by monitoring the order of the properties in the PUBLISH packets to establish significant discrepancies in it. This covert channel can be prevented even if the broker actively influences the MQTT packets by always rearranging present order properties in a previously specified manner.

A possible countermeasure for detection of the indirect covert channel I2 is to count occurrences of consecutive (name, value) pairs with the same name. It cannot be prevented, because the broker has to send all User Properties unaltered and must maintain their order when forwarding the Application Message.

The channel I3 is very difficult to detect because a situation where a client with the role of a server offers two or more different services to other clients using the Request/Response mode of operation is quite common. However, if there is a knowledge that the particular client (as a server) offers only one service, observed deployment of two different Response Topics for the same Client Identifier or the same IP or MAC address can be treated as anomaly and prove the existence of the covert channel.

One possible method for detection of I4 is to measure the times between the consecutive publishing from the same Client Identifier (or the same IP or MAC addresses) to the given topic. Significant discrepancies when compared with the previous legitimate recordings can be an indicator of the possible covert channel.

One possible way to detect I5, is to keep a record of (Client Identifier, MAC address) pairs and to look for at least two different pairs with the same Client Identifier or for many different pairs with the same MAC address (if the CS does not use MAC spoofing). A second option is to keep track of the amount of disconnections and reconnections with the reason code 0x8E. If this number becomes large, this can be a sign of possible covert channel existence. In addition, too many reconnections can be considered as a Denial of Service (DoS) attack.

Fig. 11 – Experimental testbed for I4 covert channel.

5. Experimental evaluation

In order to prove the effectiveness and feasibility of the proposed covert channels we have implemented I4 and I5 covert channels because their novelty and significance are higher than in case of the other channels. Moreover, for them legitimate traffic recordings can be artificially created (in the absence of real datasets). In result, in this section we present experimental evaluation of I4 and I5 undetectability, bandwidth, and robustness.

We left out the other three indirect CCs from the evaluation because the concept of User Properties and Request/Response mode of operation are introduced in version 5.0, and we were unable to find any real examples of their application, thus we could not create traffic recordings that can correspond to some legitimate use. Maybe in the future, when these features will be commonly used one can try to create a proof-of-concept implementation of these CCs.

We also decided not to experimentally evaluate the direct CCs as there are already a lot of examples of similar methods available in the literature (e.g., Cabuk et al., 2004; Houmansadr and Borisov, 2011; Liang et al., 2018; Murdoch and Lewis, 2005; Rowland, 1997).

5.1. *Covert channel I4*

5.1.1. *Detectability of covert channel I4*

To evaluate this CC, we used three EC2 instances of Amazon Web Services (AWS) to conduct our experiments, see Fig. 11. On one of the EC2 instances we installed HiveMQ broker v4.3.1. This version of the broker has support for MQTT 5.0. Using Java programming language, we implemented the covert sender (publisher) on another EC2 instance and the covert receiver (subscriber) on the third EC2 instance. For this purpose we used the HiveMQ Java client library v1.1.4 which also supports MQTT 5.0. Finally, we utilized Wireshark v3.2.1 to record the MQTT traffic.

For our experiments, we used $m = 2$ and $2^2 = 4$ different topics (T0, T1, T2 and T3). As already done in Velinov et al. (2019), we performed a so-called *countermeasure variation* (see Wendzel et al., 2018) to investigate whether the so-called *compressibility score* (κ) could be used to detect CC I4. To calculate κ, we recorded the first approx. 7500 MQTT packets of a connection and extracted the topics that appeared within the connection. To achieve this, we used different *window sizes*, i.e., the number of topics to consider within each flow. Next, we enumerated each flow's topics (the first topic became number 0, the second number 1, etc.) and concatenated the enumerated numbers of the topics to a string. Finally, we calculated the *compressiblilty score* κ by dividing original string length by the compressed length of the string. As a secondary metric, we calculated the number of topic changes between succeeding packets within the first 1000 MQTT packets for a flow. To compare legitimate and covert traffic, we investigate three different scenarios for our experimental testbed:

- *Scenario 1:* Only legitimate traffic is used,
- *Scenario 2:* CC with secret messages in plain ASCII,
- *Scenario 3:* CC with secret messages encrypted with AES (Advanced Encryption Standard).

For each scenario, we generated traffic samples. In the first scenario, we obtained legitimate traffic recordings, where the publisher sends message updates by randomly deciding every second whether to publish or not for each of the 4 topics separately and with random time (in milliseconds) between each publishing event.

In the second scenario, we created an I4 covert channel, and we sent secret messages in plain ASCII format (encoded by the presence of particular topics). We created three variants of this channel: at each second we published in: *(i)* 1 out of 4, *(ii)* 2 out of 4 and *(iii)* 3 out of 4 topics, which corresponds to transmitting 2, 4 or 6 bps, respectively. The choice to publish in 2 topics out of 4 is motivated by the largest number of combinations for two topics (6 out of 16 possible) from which it follows that the probability of the event to publish in 2 topics per second is the highest, compared to other possible events. Note, that additionally we performed the experiments with 1 and 3 topics (each with 4 possible combinations) per second, just to observe what will happen in such case. The time between consecutive publishing of messages was random (in milliseconds).

In the third scenario, we created again an I4 covert channel where the secret messages were encrypted beforehand with the AES. Again, we created the same three variants of this channel, i.e., publishing in 1 out of 4, 2 out of 4 and 3 out of 4 topics per second with random time between consecutive publishing events, which corresponds to sending 2, 4 or 6 bps, respectively). This CC was used to show how encryption influences the undetectability compared with unencrypted traffic.

Results The obtained compressibility scores are highly similar for both, the AES-encoded covert channel and the legitimate traffic: for both channel types, the κ value was around 4.0 for a window size (i.e., the number of topics for a flow to consider for compressibility purposes) of 1750,[2] see Fig. 12. This

[2] This window size performed best in our experiments.

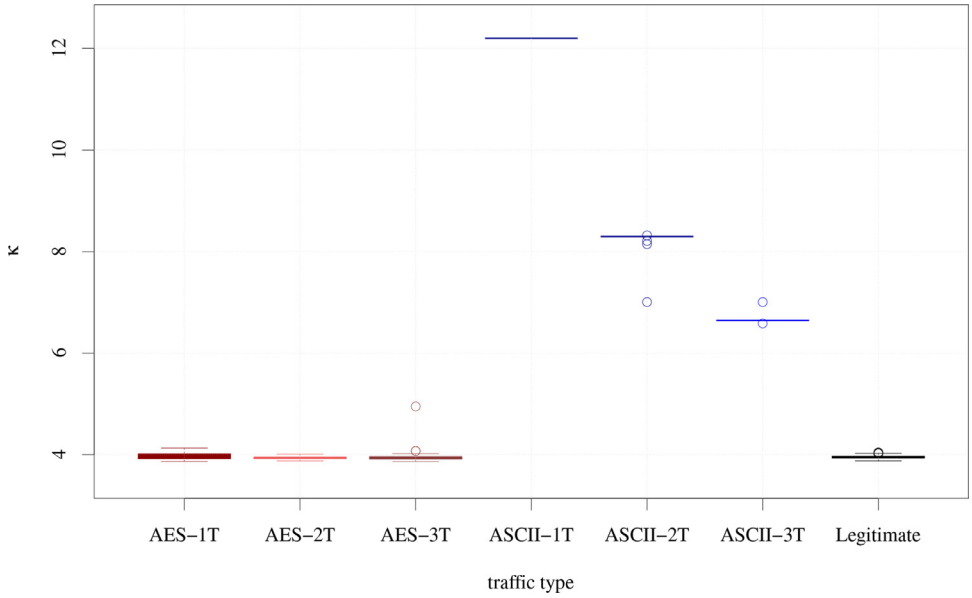

Fig. 12 – κ scores for I4 – AES-encoding vs. ASCII-encoding.

is rooted in the fact that both channels essentially perform a random selection of the topic to be sent next. Thus, the AES channel cannot be detected in our experimental setup.

However, the ASCII channel's compressibility scores differ from those of the AES-encoded covert channel's and the legitimate channel's (see Fig. 12 for window size = 1750). The reason for this is that ASCII data contains less information per bit, so it can be compressed easily, i.e., leading to higher κ values. A window size of 1250 to 2500 combined with a suitable threshold resulted in a detectability of 100% when an ASCII channel is required to be distinguished from an AES or legitimate channel (e.g., for the window size 1750 the optimal threshold is $\kappa = 5.5$, while the threshold slightly increases with higher window sizes).[3] However, the longer the window sizes, the fewer flows can be considered as they need to contain enough considerable packets.

The second metric (the number of topic changes within a connection for windows of 500 to 3000 packages) resulted in no useful distinction. The number of topic changes in our legitimate traffic was strongly overlapping with the number of topic changes for both covert channel setups.

5.1.2. *Bandwidth of covert channel I4*

We performed experiments with three different bandwidths, 2, 4 and 6 bps, which corresponds to publishing in 1, 2 and 3 topics, respectfully. We wanted to observe what happens with undetectability if we choose the most possible event (publishing in 2 topics) compared to the less likely ones (publishing in 1 or 3 topics).

The first 20 s of one of the 100 performed experiments for I4 CC with 4 topics is presented in Fig. 13. During these 20 s, 80 secret bits are transmitted from the 3.6th till the 23.6th second. The figure indicates the points in time when publishing in T0-T3 occurs.

[3] Short traffic flows that would only fill tiny windows < 400 using ASCII-encoding cannot be detected since they do not contain enough redundancy to achieve distinguishable compressibility scores.

Table 3 – The number (and percentage) of flipped bits in transferred 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%)
Avg.	**0.67** (0.84%)	**1.33** (1.66%)	**2.00** (2.50%)	**4.67** (5.84%)	**10.00** (12.50%)	**14.67** (18.34%)

5.1.3. *Robustness of covert channel I4*

The I4 CC is not a timing channel thus in a case of any constant network delay, the secret bits will be received in the correct order, because all the covert messages will be delivered with the same delay. The following Table 3 presents the numbers of flipped bits at the receiver side (bit-errors) if we introduce different delays on each 2nd second. We experimented with six different delays (of 50 ms, 100 ms, 200 ms, 300 ms, 400 ms, and 500 ms) and with publishing in 1, 2 and 3 topics per second. Transmission errors occur because the order of the receiving message updates is changed in some cases. If one message update arrives later than its consecutive message three cases are possible. If the consecutive message updates belong to the same topic, bits are without errors. If the consecutive topics are T0 and T3, all four bits will suffer from errors. For any other combination of topics, only 2 bits will be not successfully transmitted.

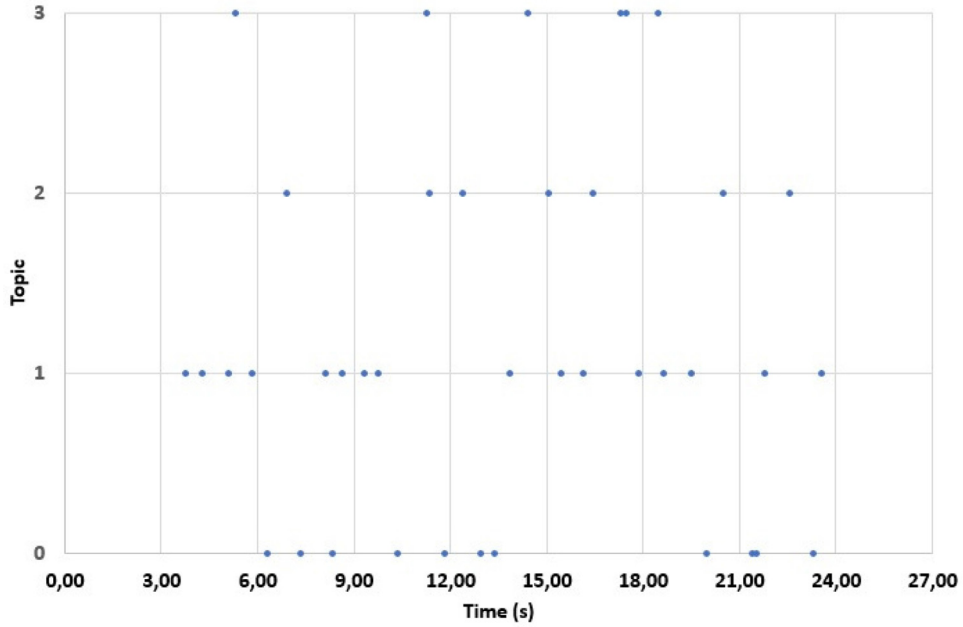

Fig. 13 – I4 CC with 4 topics, sending 80 secret bits in 20 s.

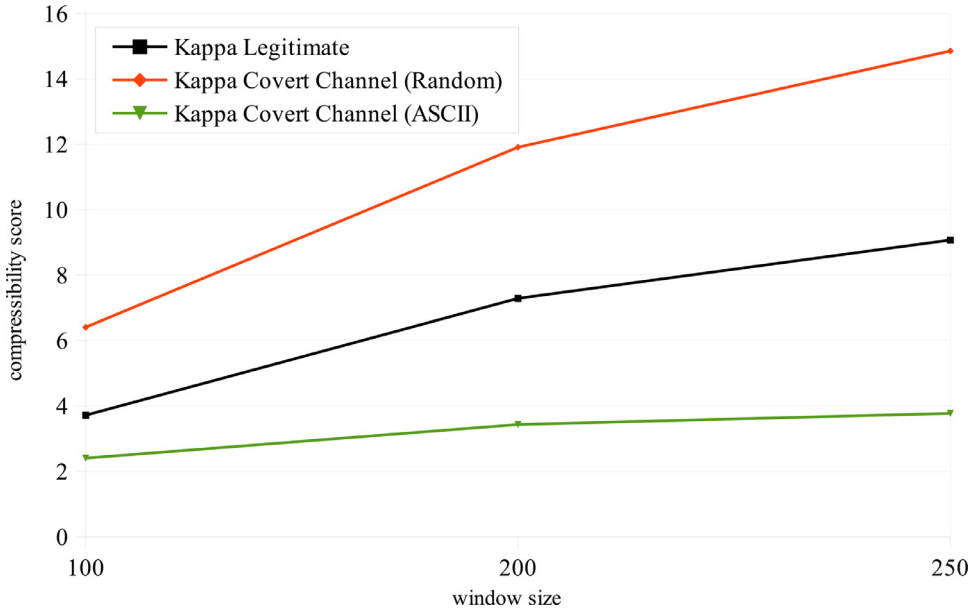

Fig. 14 – κ scores for I5 (randomized data and ASCII data).

On average, the experimental results indicate that for 50 ms of introduced delay, 0.84% of all bits are received with errors, for 100 ms it is 1.66%, for 200 ms it is 2.50%, for 300 ms it is 5.84%, for 400 ms it is 12.50%, and finally for 500 ms 18.34% of the bits are received with errors. So, this small experiment proves that by increasing the delay, the level of bit errors also increases.

The nature of the CC I4 is such that with the loss of each packet, 2 bits for the case with 4 topics are lost. Packet losses are occurring only during the time when the receiver is disconnected from the broker, because in other cases, TCP provides packet re-transmissions. So, for one packet loss per 20 published message updates (5%) we will have 2 lost secret bits per 40 transmitted secret bits (5%).

5.2. *Covert channel I5*

5.2.1. *Detectability of covert channel I5*

As in the case of channel I4, we performed a countermeasure variation on the basis of the compressibility score to detect covert channel I5. Therefore, we concatenated the enu-

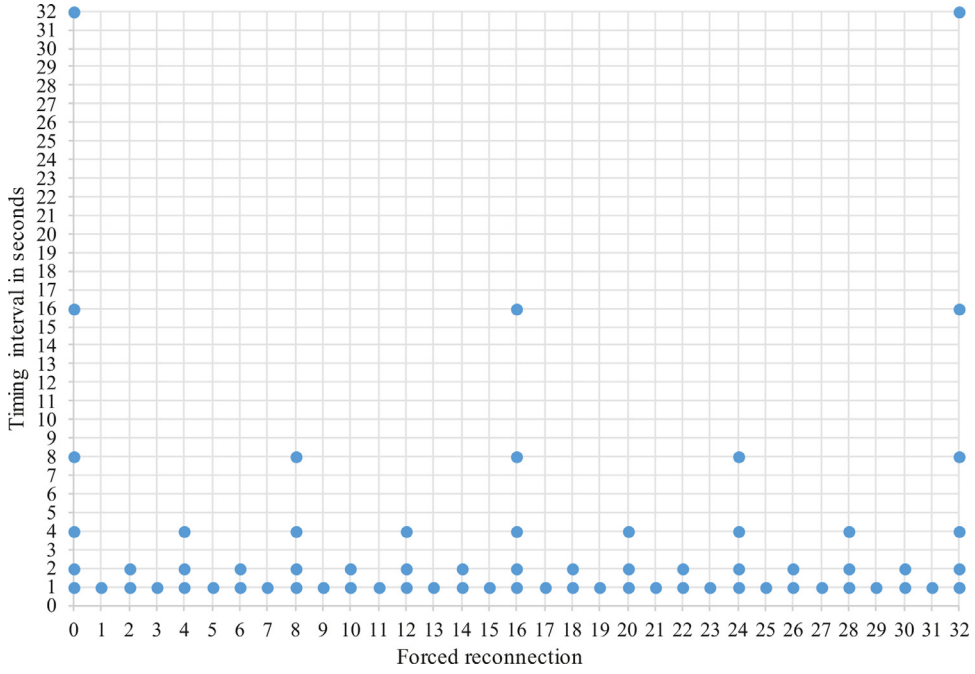

Fig. 15 – Experimental interval structure.

Table 4 – The number (and percentage) of lost messages (sent in a persistent session all 10 s) in transferred 180 characters of a secret message.

	1 s	2 s	4 s	8 s	16 s	32 s
Pub	0/18	0/36	0/72	0/144	0/288	0/576
Sub	6/18 (33.33%)	7/36 (19.44%)	9/72 (12.5%)	8/144 (5.56%)	10/288 (3.47%)	13/576 (2.26%)

merated client-ID values of connections for each particular node in a row (window sizes: 100, 200 and 250 client-ID values). If the client-ID changed, we inserted an "!" between the old and the new value. For instance, if the client-IDs "client A" and "client B" would be used in the following order "client A", "client A", "client B", "client A", "client B", then the resulting string (after enumerating the client-ID values) would be S ="00!1!0!1". Thus, connections with many client-ID changes result in longer strings.

As in the case of I4, we transferred encrypted (random) as well as ASCII content via the covert channel while the legitimate traffic used always the same client-ID values (since these values typically do not change per node).

Results As can be observed in Fig. 14, the compressibility scores of the covert channel traffic and legitimate traffic were clearly distinguishable for both, random (encrypted) covert channel traffic and ASCII covert channel traffic. The results where independent of the consumed bandwidth of the covert channels as the particular window sizes could be filled within a few seconds or even days. The window size of 250 client-ID values performed best. Overall, we conclude that this covert channel is easily detectable.

5.2.2. *Bandwidth of covert channel I5*

To examine the bandwidth of this channel, experiments were performed which relied on sending one character (16 bits) per forced reconnection at intervals of 1, 2, 4, 8, 16 and 32 s. Also, for testing purposes, bits instead of characters (i.e., 1 bit per reconnection) were sent at the same intervals. The analysis of I5 was performed once with QoS = 0 (*Quality of Service*; non-persistent session) and once with QoS = 1 (persistent session). The publisher therefore sends a PUBLISH message every 10 s. Fig. 15 illustrates the different timing intervals including their forced reconnections exemplifying the first 32 s.

5.2.3. *Robustness of covert channel I5*

The experiments show that with QoS = 0 messages are lost once reconnections occur. In case of a forced reconnection each second, only about 2/3 of published messages arrive from publisher to broker, from broker to subscriber the rate of the originally sent messages is then only about 1/3, where the half of the really transmitted messages by the publisher is arriving at the subscriber. If the intervals are increased, then at an interval of 32 s 98.2% (publisher to broker) and 96.8% (broker to subscriber) or coupled 98.6% of all published messages arrive at the subscriber. With QoS = 1 and the correspondingly persistent session, all messages sent by the publisher arrive at the broker. The messages forwarded to the subscriber are at a time interval of 1 s at about 2/3, at an interval of 32 s at 97.7%.

In every case, the messages from the CS received by the CR are at 100%. Table 4 shows that with an increasing CS transmission time the CC can be found less effectively than if it causes highly frequented reconnections. The lost messages and their percentages may vary, but they decrease with longer intervals anyway.

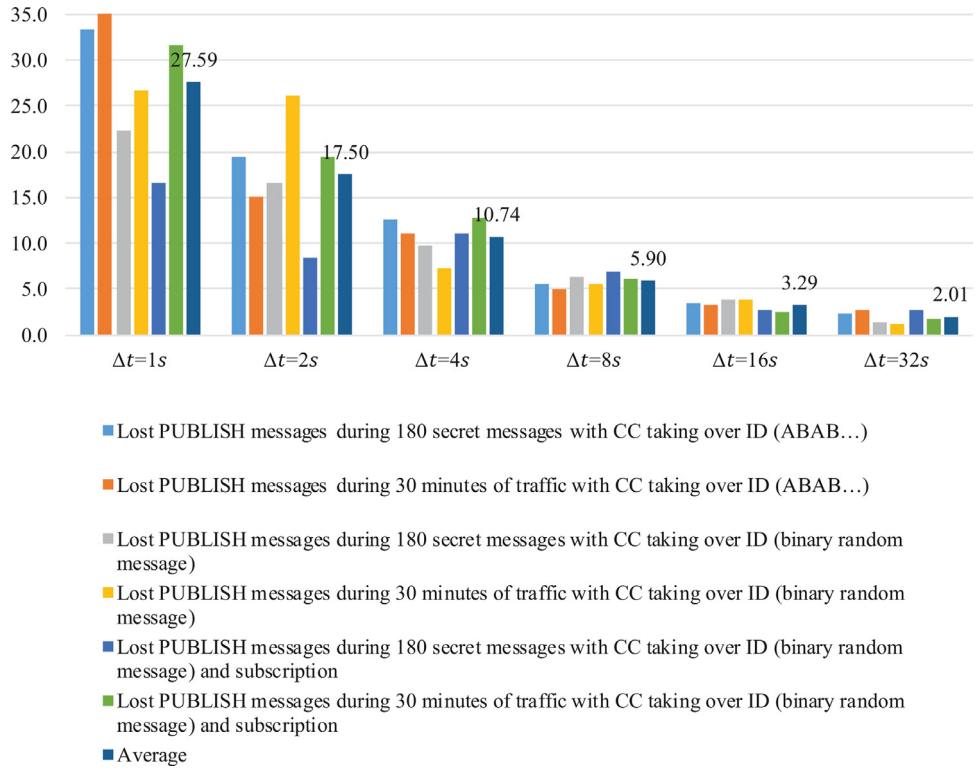

Fig. 16 – Differing rates due to higher Δt (broker to subscriber, QoS = 1, in %).

Fig. 16 illustrates different types of experiments (random character like in Table 4 (CC with CONNECT), random binary (CC with CONNECT) and random binary with CC connects and subscribes) and their lost PUBLISH messages split into the first 180 secret signs (characters or bits), the secret signs over a period of 30 min and the average of all of the recordings.

The trend emphasizes that the higher Δt the lower the lost messages level. However, the CS also has the option of taking over only the publisher IDs, which would reduce the rate of lost messages to 0% at QoS = 1.

In case of real reconnections caused by network disturbances of the clients, these can be easily filtered out and do not disturb the secret message for the CR by checking the source IP. Finally, it can also be concluded that the shorter, more varied in the characters or less highly frequented a secret message is sent, the less effectively the CC can be found.

6. Conclusion

In this paper, we have analyzed MQTT 5.0 regarding its exploitability from network covert channels perspective. Based on the conducted study we were able to demonstrate that novel covert channels exist that were not feasible for previous MQTT versions as they rely on MQTT 5.0 specific features. Moreover, we introduced a novel timing channel hiding pattern that in order to transfer secrets utilizes reconnections. We have structured our analysis using a unified description method and evaluated the bandwidth, robustness and undetectability of selected covert channels and also discussed potential countermeasures.

As our future work, we plan to further design and evaluate countermeasures for MQTT 5.0, especially we focus on developing an active warden capable of normalizing/limiting the covert channels proposed in this paper.

Credit author statement

All authors contributed equally to the research described in this paper.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Part of this research was funded by the European Union from the European Regional Development Fund (EFRE) and the State of Rhineland-Palatinate (MWWK), Germany. Funding content: P1-SZ2-7 F&E : Wissens- und Technologietransfer (WTT), Application number: 84003751. Wojciech Mazurczyk was funded by POB Research Centre Cybersecurity and Data Science of

Warsaw University of Technology within the Excellence Initiative Program-Research University (ID-UB).

REFERENCES

Zander S, Armitage G, Branch P. A survey of covert channels and countermeasures in computer network protocols. IEEE Commun. Surv. Tutor. 2007;9(3):44–57. doi:10.1109/COMST.2007.4317620.

Banks A., Briggs E., Borgendale K., Gupta R.. MQTT version 5.0. OASIS Standard2019b;URL: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html.

Banks A., Gupta R.. MQTT version 3.1.1. Plus Errata 012015b;URL: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html.

Cabaj K, Caviglione L, Mazurczyk W, Wendzel S, Woodward A, Zander S. The new threats of information hiding: the road ahead. IEEE IT Prof. 2018;20(3):31–9. doi:10.1109/MITP.2018.032501746.

Cabuk S, Brodley CE, Shields C. IP covert timing channels: design and detection. In: Proc of the 11th ACM Conference on Computer and Communications Security; 2004. p. 178–87. doi:10.1145/1030083.1030108.

Caviglione L, Merlo A, Migliardi M. Covert channels in IoT deployments through data hiding techniques. In: Proc of the 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA 2018); 2018. p. 559–63. doi:10.1109/WAINA.2018.00144.

Cronin P, Gouert C, Mouris D, Tsoutsos NG, Yang C. Covert data exfiltration using light and power channels. In: 2019 IEEE 37th International Conference on Computer Design (ICCD), Abu Dhabi, United Arab Emirates; 2019. p. 301–4. doi:10.1109/ICCD46524.2019.00045.

Fraczek W, Mazurczyk W, Szczypiorski K. Hiding information in stream control transmission protocol. Comput. Commun. 2012;35(2):159–69. doi:10.1016/j.comcom.2011.08.009.

Herzberg A, Kfir Y. The chatty-sensor: a provably-covert channel in cyber physical systems. In: Proc of the 35th Annual Computer Security Applications Conference (ACSAC '19), San Juan, Puerto Rico; 2019a. p. 638–49. doi:10.1145/3359789.3359794.

Herzberg A, Kfir Y. The leaky actuator: a provably-covert channel in cyber physical systems. In: Proc of the ACM Workshop on Cyber-Physical Systems Security & Privacy, London, United Kingdom; 2019b. p. 87–98. doi:10.1145/3338499.3357358.

Houmansadr A, Borisov N. COCO: coding-based covert timing channels for network flows, 6958; 2011. p. 314–28. doi:10.1007/978-3-642-24178-9_22.

Liang C, Tan Y, Zhang X, Wang X, Zheng J, Zhang Q. Building packet length covert channel over mobile voIP traffics. J. Netw. Comput. Appl. 2018;118:144–53. doi:10.1016/j.jnca.2018.06.012.

Lin J, Yu W, Zhang N, Yang X, Zhang H, Zhao W. A survey on internet of things: architecture, enabling technologies, security and privacy, and applications. IEEE Internet Things J. 2017;4(5):1125–42. doi:10.1109/JIOT.2017.2683200.

Mao Y, Zhao D, Wang X, Zhao S, Zhang Y. Burglars IoT paradise: understanding and mitigating security risks of general messaging protocols on IoT clouds, 1; 2020. p. 465–81. doi:10.1109/SP40000.2020.00051.

Martins D, Guyennet H. Attacks with steganography in PHY and MAC layers of 802.15.4 protocol. In: Proc of the Fifth International Conference on Systems and Networks Communications; 2010. p. 31–6. doi:10.1109/ICSNC.2010.11.

Mazurczyk W, Szczypiorski K. Steganography of voIP streams. In: Meersman R, Tari Z, editors. In: Proc. of The 3rd International Symposium on Information Security (IS'08), Monterrey, Mexico, November 10–11. Springer-Verlag Berlin Heidelberg; 2008. p. 1001–18. doi:10.1007/978-3-540-88873-4_6. OTM 2008, Part II - LNCS 5332

Mazurczyk W, Wendzel S, Cabaj K. In: Proc Second International Workshop on Criminal Use of Information Hiding (CUING 2018) at ARES. Towards deriving insights into data hiding methods using pattern-based approach. ACM; 2018. doi:10.1145/3230833.3233261. 10:1-10:10

Mazurczyk W, Wendzel S, Zander S, Houmansadr A, Szczypiorski K. In: IEEE Press Series on Information and Communication Networks Security. Information hiding in communication networks: Fundamentals, mechanisms, applications, and countermeasures. IEEE Press-Wiley; 2016. April ISBN-10: 1118861698

Mehta AM, Lanzisera S, Pister KSJ. Steganography in 802.15.4 wireless communication. In: Proc of the 2nd International Symposium on Advanced Networks and Telecommunication Systems; 2008. p. 1–3. doi:10.1109/ANTS.2008.4937785.

Mileva A, Panajotov B. Covert channels in the TCP/IP protocol stack – ext. ver. Central Eur. J. Comput. Sci. 2014;4(2):45–66. doi:10.2478/s13537-014-0205-6.

Mileva A, Velinov A, Stojanov D. New covert channels in internet of things. In: Proc of the 12th International Conference on Emerging Security Information, Systems and Technologies (SECURWARE 2018), Venice, Italy, September 16-20; 2018. p. 30–6.

Moskowitz IS, Russell S, Jalaian B. Steganographic internet of things: graph topology timing channels. In: Proc of Workshops of the The Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7; 2018. p. 252–9.

Murdoch SJ, Lewis S. Embedding covert channels into TCP/IP, 3727; 2005. doi:10.1007/11558859_19. 247–226

Nain AK, Rajalakshmi P. A reliable covert channel over IEEE 802.15.4 using steganography. In: 2016 IEEE 3rd World Forum on Internet of Things (WF-IoT), Reston, VA; 2016. p. 711–16. doi:10.1109/WF-IoT.2016.7845486.

Hron M.. Are smart homes vulnerable to hacking?; 2018; Online Available: https://blog.avast.com/mqtt-vulnerabilities-hacking-smarthomes.

Patuck R., Hernandez-Castro J.. Steganography using the extensible messaging and presence protocol (XMPP). Comput. Res. Repository. 2013. arXiv, ID: 13100524.

Report T.S.F.O.M.. Expanding the hunt for vulnerable IoT devices. 2020. [Online], March 15, Available: https://www.shadowserver.org/news/open-mqtt-report-expanding-the-hunt-for-vulnerable-iot-devices/.

Ronen E, Shamir A. Extended functionality attacks on IoT devices: the case of smart lights. In: IEEE European Symposium on Security and Privacy (EuroS&P), Saarbrucken; 2016. p. 3–12. doi:10.1109/EuroSP.2016.13.

Rowland C. In: First Monday, Peer Reviewed Journal on the Internet. Covert channels in the TCP/IP protocol suite; 1997.

Sikder A.K., Petracca G., Aksu H., Jaeger T., Uluagac: A.S.. A survey on sensor-based threats to internet-of-things (IoT) devices and applications. Comput. Res. Repository. arXiv, ID:1802020412018;.

Tan Y, Zhang X, Sharif K, Liang C, Zhang Q, Li Y. Covert timing channels for IoT over mobile networks. IEEE Wirel. Commun. 2018;25(6):38–44. doi:10.1109/MWC.2017.1800062.

Ulz T, Feldbacher M, Pieber T, Steger C. Sensing danger: exploiting sensors to build covert channels. In: Proc of the 5th International Conference on Information Systems Security and Privacy (ICISSP 2019), Prague, Czech Republic, Feb; 2019. p. 100–13. doi:10.5220/0007353801000113.

Velinov A, Mileva A, Wendzel S, Mazurczyk W. Covert channels in the MQTT-based internet of things. IEEE Access 2019;7:161899–915. doi:10.1109/ACCESS.2019.2951425.

Wendzel S. Covert and side channels in buildings and the prototype of a building-aware active warden, 2012; 2012. p. 6753–8. doi:10.1109/ICC.2012.6364876.
Wendzel S, Link F, Eller D, Mazurczyk W. Detection of size modulation covert channels using countermeasure variation. J. Univ. Comput. Sci. 2018;25(11):1396–416. doi:10.3217/jucs-025-11-1396.
Wendzel S, Mazurczyk W, Haas G. Steganography for cyber-physical systems. J. Cyber Secur. Mobility 2017;6(2):105–26. doi:10.13052/jcsm2245-1439.621.
Wendzel S, Mazurczyk W, Zander S. Unified description for network information hiding methods. J. Univ. Comput. Sci. 2016;22(11):1456–86. doi:10.3217/jucs-022-11-1456.
Wendzel S, Mazurczyk W, Haas G. Don't you touch my nuts: Information hiding in cyber-physical systems. In: Proc of the IEEE SPW 2017, San Francisco, USA; 2017. p. 29–34. doi:10.1109/SPW.2017.40.
Wendzel S, Zander S, Fechner B, Herdin C. Pattern-based survey and categorization of network covert channels. ACM Comput. Surv. 2015;47(3) 50:1–50:26. doi:10.1145/2684195.
Ying X, Bernieri G, Conti M, Poovendran R. TACAN: Transmitter authentication through covert channels in controller area networks. In: Proc of the 10th ACM/IEEE International Conference on Cyber-Physical Systems; 2019. p. 23–34. doi:10.1145/3302509.3313783.

Aleksandra Mileva received the B.Sc., M.Sc., and Ph.D. degrees from Ss. Cyril and Methodius University in Skopje, Macedonia. She is currently an Associate Professor with the Faculty of Computer Science, University Goce Delcev, Štip, Macedonia, where she is also the Head of the Laboratory of Computer Security and Computer Forensics. Her research interests include computer and network security, digital steganography, the IoT protocols and security, cryptography, computer forensics, and quasi-groups theory. Since 2019, she has been a member of the EURASIP Data Forensics and Security TAC. She was with the management committee of two COST actions IC1201: BETTY and IC1306: Cryptography for Secure Digital Interaction.

Aleksandar Velinov received the M.Sc. degree in computer science from the Faculty of Computer Science, University Goce Delcev, Štip, Macedonia, in 2016, where he is currently pursuing the Ph.D. degree. He is also a Teaching Assistant with the Faculty of Computer Science, University Goce Delcev. His research interests include the Internet of Things (IoT), machine-to-machine (M2M), digital steganography, computer and network security, big data, big data analysis, learning analytics, cloud computing, and mobile technologies. He is involved in the following projects: Blended Learning and Developing and Testing and Installing E-learning System for African Member States.

Laura Hartmann received the Mobile Computing M.Sc. degree from the University of Applied Sciences Worms, Germany, in 2016. At the University of Hagen, Germany, she is currently pursuing her Ph.D. degree in computer science. She is a member of the Cyber Security Research Group of the University of Applied Sciences in Worms. She works on the funded project MADISA (Machine Learning for Attack Detection Using Data of Industrial Control Systems). Her research interests include anomaly detection, network security, machine learning, steganography, the Internet of Things and industrial control systems.

Steffen Wendzel received the Ph.D. degree in computer science from the University of Hagen, Germany, in 2013. From 2013 to 2016, he led a smart building security research team at Fraunhofer FKIE, Germany. He joined the Worms University of Applied Sciences as a professor of information security and computer networks, in 2016. Since 2017, he has been the deputy scientific head of the universitys Centre for Technology and Transfer. He wrote six books and published more than 140 papers. His papers appeared in journals, such as FGCS, CSUR, IEEE S&P Comm. ACM, Scientometrics, Annals of Telecommunications, and J.UCS. His research interests include information hiding and the Internet-of-Things security. He is a member of the Editorial Board of the Journal of Universal Computer Science (JUCS) and the Journal of Cyber Security and Mobility (JCSM) as well as a Steering Committee Member of CUING. Steffen (co-)organized several workshops and conferences and served as a Guest Editor for the IEEE TII, IEEE S&P, Elsevier FGCS, Wiley SCN, JUCS, and other journals.

Wojciech Mazurczyk received the B.Sc., M.Sc., Ph.D. (Hons.), and D.Sc. (Habilitation) degrees in telecommunications from the Warsaw University of Technology (WUT), Warsaw, Poland, in 2003, 2004, 2009, and 2014, respectively. He is currently a University Professor with the Institute of Computer Science, WUT. He is also a Researcher with the Parallelism and VLSI Group, Faculty of Mathematics and Computer Science, FernUniversitaet, Germany. His research interests include bioinspired cybersecurity and networking, information hiding, and network security. He is involved in the Technical Program Committee of many international conferences and also serves as a Reviewer for major international magazines and journals. Since 2016, he has been the Editor-in-Chief of an open access Journal of Cyber Security and Mobility. Since 2018, he has been serving as an Associate Editor for the IEEE Transactions on Information Forensics and Security and as a Mobile Communications and Networks Series Editor for the IEEE Communications Magazine.