

On The Possibilities of Cybercrime in IoT Devices

¹Yosafat Marselino Agus, ²Muhammad Dzakwan Falih, ³Gandeva Bayu Satrya

¹ School of Electrical Engineering, Telkom University, Republic of Indonesia
 ^{2,3}School of Applied Science, Telkom University, Republic of Indonesia
 ¹ninoagus@protonmail.com, ²dzakwan@ieee.org, ³gbs@telkomuniversity.ac.id

Abstract

Article Info Volume 83 Page Number: 8231 - 8238 Publication Issue: March - April 2020

Article History Article Received: 24 July 2019 Revised: 12 September 2019 Accepted: 15 February 2020 Publication: 09 April 2020 Internet of Things (IoT) has been augmenting the emerging technologies and certainly been varying our daily life. The adoption of this technology is strengthened by the growth of connecting devices as shown in recent literature. However, responsibility related to secure communication also needs to increase as the number of connections grows. For instance, cybercrime might happen if simple topology and protocol are not implemented on IoT applications, or the communications from sensors to the Internet are weakly defined. This research reviews the vulnerability of existing topology and configuration on IoT. A secure communication is proposed between sensor nodes and the Internet. Further, this research demonstrates the feasibility of recommended protocol communication for several IoT devices through real testbed for smart home.

Index Terms; IoT, protocol, sensor nodes, secured communi-cation, vulnerability.

I. INTRODUCTION

Communication between smartphone devices or sending commands to home devices is no longer a fantasy with the development of future wireless communication technology. Data communication from machine to machine or human to machine using Internet connection is called as Internet of Things (IoT). IoT is one of the most important IT tendencies and it is getting more appealing in home or factory automa-tion. Figure. 1 illustrates an IoT communication architecture consists of sensors, smart objects, smart devices, gateways, back-end data centers, and services [1].

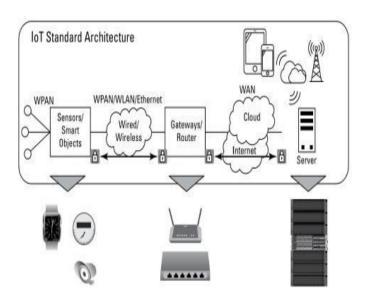


Figure. 1: IoT system architecture

Data security and privacy certainly become the main topics for any organization that is connected to the Internet. On February 2018, the Interpol National Central Bureau held a meeting in Vienna where 43 cybercrime investigators and digital forensics experts were invited from 23 countries to investigate cyberattack simulations on IoT devices [2]. They



have given four guidance on how to secure connected devices from cybercrime. The risks are far greater on IoT because of much more personal and important data collected from abundant IoT devices that could be used for vicious intentions. Thus, providing a suitable security control or communication protocol for consumers has become a necessity for IoT device vendors.

IoT needs to incorporate various sensor nodes or smart objects, computer or server and wireless connection. which are using different protocols. Based TCP/IP communication on protocol stack, several communication protocols have been proposed on each layer. For this research, it will focus on application layer. Observing last recent research update related to application's protocol for IoT, five outstanding protocols are HTTP [3], XMPP [4], MQTT [5], AMQP [6], and CoAP

a) However, MQTT is the most candidates for standard IoT communication protocols due to it reduces protocol overheads and provides reliable delivery of messages [8]–[10].

Researches in recent years focusing on IoT security prob-lems aimed for different aspect. Commencing with the survey by [11]] that proposed security requirements specific to IoT systems by taking into consideration network security, identity management, privacy, trust, and resilience from well-known standardization technology. Another approach from [12] de-signed and implemented a reliable message transmission sys-tem based on MQTT Protocol for IoT devices. However,

[13] proposed a simple security framework for MQTT (for short, AugMQTT) by incorporating the AugPAKE protocol that gives security against passive attacks, active attacks, and off-line dictionary attacks. In the era of urban computing and IoT systems, [14] identified three prominent approaches to provide seamless and lightweight IoTinteractions, high-lighting a representative, standardized protocol for each of these approaches. Acknowledging those literature, there are still many available research areas that can be explored in order to provide optimized topology, secure scheme, or modest algorithm for IoT architecture.

Initiating with the deployment MQTT protocol for IoT adopted from [10], this research shows that vulnerability i.e. stealing data information can give a serious hit. That reason brought this research to optimization of MQTT protocol implementation in IoT environment. By using that topology, the pentesting for deep investigations were successfully con-ducted and developed secure end to end communication from sensor nodes to back-end data centers. The implementation and testing were carried out using three Raspberry Pi 3B as IoT devices, Mosquitto as MQTT broker, Eclipse Paho library on Python for IoT, and Wireshark for protocol analyzerThe contributions of this research are as follows:

Proposing a secure topology by using MQTT over web-socket where the unregistered IoT devices might not connect.

Providing an optimized MQTT protocol configuration based on the testbed by adding lightweight cryptography from physical layer to application layer.

Presenting a guidance for secure IoT device topology by considering smart hospital, smart home, smart factory, or other enhanced IoT technology serving vital information.

The section II reviews existing works in IoT security. While section III explains the proposed scheme for secure end to end communication among IoT devices. Then, the details of unsecure and secure experiments are provided in section IV. Finally, section V gives the conclusions of this research.



II. RELATED WORKS

Singh et al. proposed a secure version of MQTT and MQTT-SN protocols (SMQTT and SMQTT-SN) in which the security feature is augmented to the existing MQTT protocol based on Key/Ciphertext Policy-Attribute Based Encryption (KP/CP-ABE) using lightweight Elliptic Curve Cryptography [15]. According to the theoretical and experimental analysis, the proposed schemes showed better performance than the scheme described in [3] which was based on KP/CP-ABE. However, those proposed schemes were just simulation-based and has not yet been implemented on a real IoT platform.

Dragomir et al. have done surveys in the last several years to prevent attackers from obtaining control over the IoT devices

1) Many solutions have been proposed by IoT security researches in the last years, but most of them were not standardized or interoperable. The research evaluated security requirements specific to IoT systems considering the network security, identity management, privacy, trust, and resilience. The results were summarized by functionality and security capabilities and specified by standardization bodies e.g., IETF, IEEE, or industry alliances.

Andy et al. discussed several reasons of many IoT system that does not implement adequate security mechanism. The research also demonstrated and analyzed how to easily attack this protocol using several attack scenarios [16]. Considering Shodan as the problem identification, it showed that 24998 brokers who have connection code of "0" were easier to be attacked because this kind of broker did not use any client authentication mechanism. To the best of authors' knowledge, this paper just concluded with a suggestion that MQTT pro-tocol must be implemented e.g., TLS, ECC, or other recent secured protocol on IoT devices.

Grgic´ et al. [17] proposed a web-based IoT solution aimed for monitoring, tracking and analyzing data in agriculture area. The research was done by installing and connecting sensors to devices (e.g. Arduino Uno, Raspberry Pi) which main task was to send data (i.e. temperature and moisture values) to a central server over MQTT. The results showed that MQTT protocol has several characteristics i.e., low overhead, asynchronous communication, low complexity and low power.

Niruntasukrat et al. [18] presented design and implemen-tation of an authorization mechanism by using MQTT for IoT. The design was based on the OAuth 1.0a, an open authorization standard for web applications. Some redesign and modification have been made to the base framework to make it fit within the MQTT environment. However, the authorization mechanism demanded two sets of credentials for the device to access the MQTT broker. Through experiments and real services, the proposed authorization process worked as intended, and the incurred overhead did not affect user's experiences.

III. PROPOSED FRAMEWORK

For the sake of completeness in securing communication between the IoT devices, this research implements MQTT as the prime candidate from recent existing literature. Figure. 2a shows the deployment of conventional MQTT implementation for IoT topology or called as unsecured transaction framework

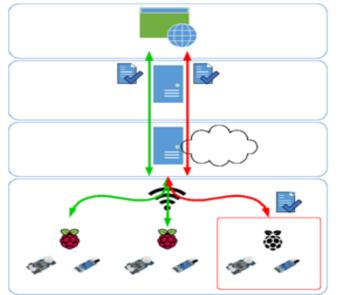
[10]. The next section provides a deep investigation for this topology. To the best of authors' knowledge, most of the simulation and evaluation of MQTT just gave assurance on the protocol level from sensor nodes to MQTT broker. But from adversary's point of view (unregistered sensor nodes), the data sensor over web-socket still can be diagnosed with plain text. For concealing this issue, this research suggests adopting auto-keying between publishers and subscribers to reject the unregistered sensor nodes. Proposed IoT topology for securing end to end communication from sensor nodes to web platform



(HTTPS) or called secured transaction framework is depicted in Figure. 2b.

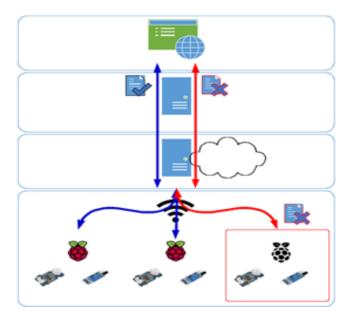
A. Unsecured Transaction Framework

The implementation was deployed using two unit of Rasp-berry Pi 3B as registered devices and one unit as an un-registered device. The Wi-Fi connection had medium access to MOTT Publisher. The configuration for MQTT broker, MQTT subscriber, and database server had been built by following the MQTT library or documentation as can be seen in Figure. 2a. This topology also provided HTTPS web server for pooling the data's sensor. The pen-testing this topology are discussed in the following section. MQTT is a topic-based publish/subscribe protocol. Every message is published on a designated topic,



and every subscription has a topic filter that can include wildcards. So, the authorization is in terms of publishing/subscribing and topic designating.

The most common problem in implementing IoT system based on MQTT protocol is the poorly configured devices. The default configuration of MQTT devices running on Mosquito service and Paho MQTT library on Python enables the data with unencrypted format to be transmitted over the Internet. By imitating the format data and the topic, some adversaries may spoof the broker's and the subscriber's sides. Faking the data or crippling the server is possible in this scenario. Upon this hypothesis, further analysis will be addressed in the next section.



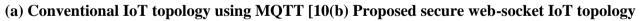


Figure. 2: Comparison IoT architecture with optimized configuration by using MQTT protocol

B. Secured Transaction Framework

Slightly, there is no difference between the proposed topol-ogy and the conventional IoT topology. As shown in Fig-ure. 2b, this research recommends lightweight cryptography algorithm for IoT devices to camouflage the messages from sensor node to web platform or to prevent the possibility of the publisher's data stealing (unregistered Raspberry Pi device). The implementation of costumed Fernet library at the publisher's, broker's, subscriber's, and web platform's sides, the secure topology were enforced to be more secure.

Fernet cryptography guarantees that a message, that was encrypted by using it, cannot be manipulated or read without the key. Fernet is categorized a symmetric authenticated cryp-tography. This research employs 128-bit AES and HMAC for the authentication and modifies the Fernet to be more robust in key rotation. Thus, the IoT devices were authenticated and only the authorized devices with 8234



the key can publish and subscribe to a restricted topic. This research hypothesizes that the proposed topology could give protection from fake publishers, fake devices, and invalid uses of data.

Algorithm 1 Autokeying Publisher Encryption

1	INITIALISE publisher_II	D;
2	INITIALISE dataSensors_sen	it;
3	INITIALISE MQTT_connect	ion;
4	passcode = HMAC "encoded_	
5	final_key = derive (passcode)	1 0
	while true do	,
7	ENCODE dataSensors_sen	t
8	° —	
:	PUBLISH dαtaSensørs_še,ñ	t . encrypted data
9	end while	••
Al	gorithm 2 Autokeying Subscri	ber Decryption
		• •
1	INITIALISE subscriber_1	ID;
2	INITIALISE MQTT_connect	ion;
3	INITIALISE final_key;	
4	•	4: final_key =
ge	nerated_key_publisher;	from
-	blishers	
	while true do	
6	SUBSCRIBE dataSensørs	še ñ°t
7	• · · · · · · · · · · · · · · · · · · ·	_ 3
9	7:DECODE	8.
d	αtαSensørs še,ñt	usingfinal_key
	end while	c = c

1) Publisher Encryption Algorithm: The publisher's iden-tity send that data (dataSensors_sent) is known. The publisher will forward the data to the MQTT broker by using MQTT connection. Upon sending a secret key to one connection, the secret key will do HMAC for each string of passwords that will be used on the subscriber's side as a guarantee of information integrity. The same key generation method can be used by all publishers, so the subscriber can decode all of the incoming information with only one key. The next process with the key is to encrypt the data $(d\alpha t\alpha S \epsilon ns \phi rs se_{\tilde{n}}^{\circ}t)$ sent by the publisher as presented in Algorithm 1.

2) Subscriber Decryption Algorithm: It is known that the identity of which subscriber will receive the encrypted data from the publisher. Indeed, the connection between the sub-scriber and the MQTT broker has been established and the key of the publisher (final_key) has been generated as presented in Algorithm 2. Then, the incoming key will be decrypted by using the key. If the format of incoming data could not be recognized, then the subscriber will reject the received data from publisher. Moreover, the subscriber will get encrypted data (dataSɛnsørs_še,ñ°t) which will then be decrypted to be displayed on the web platform

IV. SECURITY ANALYSIS

A. Unsecured Transaction Experiment

Figures. 3 and 4 shows the results of network analysis with topology as in section III is successfully extracted. Certainly, MQTT has been implemented properly which can be seen from the Wireshark output below. Data from sensor node will be sent to the web platform (HTTPS). Although the broker and MQTT broker are secure, the data sent via web-socket still can be seen (plain text). Therefore, it can be concluded that it is not enough for guaranteeing the data's integrity only by implementing the publisher node, MQTT broker, and subscriber node. Next, the solution to this problem will be explained.

lo.	Т	ime		So	urce				Destina	tion	1	Protocol	Length	Info		
	986 1	5.286	6203	19	2.168	.1.8		-	192.1	68.1.20		TCP	74	[TCP Retra	nsmissi	on] 52390 →
	1306 2															
1	1322 2	3.5330	7625	3 19	2.168	.1.20			192.1	68.1.8		ГСР	74	1883 + 523	90 [SYN	, ACK] Seq=
1	1323 2	3.533	3957	5 19	2.168	.1.8			192.1	68.1.20	1	TCP	66	52390 + 18	83 [ACK] Seq=1 Ack
1	1327 2	3.533	8873	1 19	2.168	.1.8			192.1	68.1.20	1	AQTT	114	Connect Co	mmand	
1	1334 2	3.713	1657	19	2.168	.1.20			192.1	68.1.8	1	TCP	66	1883 ÷ 523	90 [ACK] Seq=1 Ack
1	1335 2	3.713	5250	2 19	2.168	.1.20			192.1	68.1.8	1	IQTT	70	Connect Ac	k	
1	1336 2	3.713	9023	2 19	2.168	.1.8			192.1	68.1.20	1	TCP	66	52390 → 18	83 [ACK] Seq=49 Ac
1	1338 2	3.714	4239) 19	2.168	.1.8			192.1	68.1.20	1	IQTT	90	Publish Me	ssage [topic/baru]
3	1339 2	3.714	1314	5 19	2.168	.1.8			192.1	68.1.20	1	IQTT	68	Disconnect	Req	
1	1354 2	3.980	9265	1 19	2.168	.1.20			192.1	68.1.8	1	ГСР	66	1883 - 523	90 [ACK] Seq=5 Ack
3	1355 2	3.980	9931	3 19	2.168	.1.20			192.1	68.1.8	2	ГСР	66	1883 - 523	90 [FIN	, ACK] Seq=
Eth Inf Tra MQ	hernet ternet ansmi: Teler	t Prot ssion metry	Src: ocol Contr Trans	Tp-L Vers ol P port	inkT_0 ion 4, rotocc Proto	Src: Src: 1, Sr col,	28 (192 c Po Publ	18:a .168 rt: 1	6:f7:0 .1.8, 52390, Messa	Be:8c:28 Dst: 19 , Dst Po ge	8), Dst: R 02.168.1.2 ort: 1883,	aspbern 0 Seq: 4	_01:82	2:7e (b8:27 c: 5, Len: ce delivery	24	
> Eth > Int > Tra > MQ >	hernet ternet ansmi: Teler Heade Msg L Topic	t II, t Prot ssion metry er Fla Len: 2 t Leng	Src: ocol Contr Trans gs: 0 2 th: 1	Tp-L Vers ol P port x30,	inkT_0 ion 4, rotocc Proto	Src: Src: 1, Sr col,	28 (192 c Po Publ	18:a .168 rt: 1	6:f7:0 .1.8, 52390, Messa	Be:8c:28 Dst: 19 , Dst Po ge	8), Dst: R 02.168.1.2 ort: 1883,	aspbern 0 Seq: 4	_01:82	2:7e (b8:27 c: 5, Len:	24	
> Eth > Inf > Tra > MQ >	hernet ansmis Teler Heade Msg L Topic Topic	t II, t Prot ssion metry er Fla Len: 2 : Leng : top	Src: ocol Contr Trans gs: 0 2 th: 1 ic/ba	Tp-L Vers ol P port x30, 0 ru	inkT_0 ion 4, rotoco Proto Messa	e:8c: Src: bl, Src bcol, ge Ty	28 (192 c Po Publ	18:a .168 rt: 1	6:f7:0 .1.8, 52390, Messa	Be:8c:28 Dst: 19 , Dst Po ge	8), Dst: R 02.168.1.2 ort: 1883,	aspbern 0 Seq: 4	_01:82	2:7e (b8:27 c: 5, Len:	24	
> Eth > Inf > Tra > MQ >	hernet ansmis Teler Heade Msg L Topic Topic	t II, t Prot ssion metry er Fla Len: 2 t Leng	Src: ocol Contr Trans gs: 0 2 th: 1 ic/ba	Tp-L Vers ol P port x30, 0 ru	inkT_0 ion 4, rotoco Proto Messa	e:8c: Src: bl, Src bcol, ge Ty	28 (192 c Po Publ	18:a .168 rt: 1	6:f7:0 .1.8, 52390, Messa	Be:8c:28 Dst: 19 , Dst Po ge	8), Dst: R 02.168.1.2 ort: 1883,	aspbern 0 Seq: 4	_01:82	2:7e (b8:27 c: 5, Len:	24	
> Eth > Inf > Tra > MQ >	hernet ansmis Teler Heade Msg L Topic Topic	t II, t Prot ssion metry er Fla Len: 2 : Leng : top	Src: ocol Contr Trans gs: 0 2 th: 1 ic/ba	Tp-L Vers ol P port x30, 0 ru	inkT_0 ion 4, rotoco Proto Messa	e:8c: Src: bl, Src bcol, ge Ty	28 (192 c Po Publ	18:a .168 rt: 1	6:f7:0 .1.8, 52390, Messa	Be:8c:28 Dst: 19 , Dst Po ge	8), Dst: R 02.168.1.2 ort: 1883,	aspbern 0 Seq: 4	_01:82	2:7e (b8:27 c: 5, Len:	24	
> Eth > Inf > Tra > MQ >	hernet ansmis Teler Heade Msg L Topic Topic	t II, t Prot ssion metry er Fla Len: 2 : Leng : top	Src: ocol Contr Trans gs: 0 2 th: 1 ic/ba	Tp-L Vers ol P port x30, 0 ru	inkT_0 ion 4, rotoco Proto Messa	e:8c: Src: bl, Src bcol, ge Ty	28 (192 c Po Publ	18:a .168 rt: 1	6:f7:0 .1.8, 52390, Messa	Be:8c:28 Dst: 19 , Dst Po ge	8), Dst: R 02.168.1.2 ort: 1883,	aspbern 0 Seq: 4	_01:82	2:7e (b8:27 c: 5, Len:	24	
> Eth > Int > Tra - MQ >	hernet ternet ansmi: Teler Heade Msg L Topic Topic Messa	t II, t Prot ssion metry er Fla Len: 2 : Leng :: top age: 7	Src: ocol Contr Trans gs: 0 2 th: 1 ic/ba 36573 01 82	Tp-L Vers ol P port x30, 0 ru 616e: 7e	inkT_@ ion 4, rotocc Proto Messa 206c61	e:8c: Src: ol, Sr col, ge Ty 696e f7 @	28 (192 c Po Publ pe: 1	18:ad .168 rt: ! ish / Publi	6:f7:(.1.8, 52390, Messag ish Me	0e:8c:28 Dst: 19 , Dst Po ge 2ssage, 45 00	<pre>3), Dst: R 92.168.1.2 ort: 1883, Qo5 Level</pre>	aspbern Ø Seq: 4 : At mo	01:82	2:7e (b8:27 c: 5, Len:	24	
<pre>> Eth > Int > Tra > MQ > ></pre>	hernet ternet ansmi: Teler Heade Msg L Topic Topic Messa b8 2 00 4	t II, t Prot ssion metry er Fla Len: 2 : Leng :: top age: 7 27 eb 4c 52	Src: ocol Contr Trans gs: 0 2 th: 1 ic/ba 36573 01 82 18 40	Tp-L Vers ol P port x30, 0 ru 616e: 7e : 00	inkT_0 ion 4, rotocc Proto Messa 206c61 18 a6 40 06	e:8c: Src: ol, Sr col, ge Ty 696e f7 0 65 2	28 (192 c Po Publ pe: 1 pe: 1	18:a .168 rt: 1 ish / Publi	6:f7:(.1.8, 52390, Messa ish Me 08 00 01 08	00:8c:28 Dst: 19 , Dst Po ge 2ssage, 45 00 c0 a8	8), Dst: R 92.168.1.2 ort: 1883, QoS Level	aspbern Ø Seq: 4 : At mo	01:82 19, Ack st onc E-	2:7e (b8:27 c: 5, Len:	24	
> Eth > Int > Tra > MQ > >	hernet ternet ansmi: Teler Heade Msg L Topic Topic Messa 00 4 01 1	t II, t Prot ssion metry er Fla en: 2 : Leng :: top age: 7 27 eb 4c 52 14 cc	Src: ocol Contr Trans gs: 0 2 th: 1 ic/ba 36573 01 82 18 40 a6 07	Tp-L Vers ol P port x30, 0 ru 616e 7 e 00 5 b	inkT_0 ion 4, rotocc Proto Messa 206c61 18 a6 10 06 19 f5	696e f7 @ 65 2 78 4	28 (192 c Po Publ pe: 1 pe: 1 c Po Publ pe: 1 53	18:ad .168 rt: 1 ish / Publi	6:f7:(.1.8, 52390, Messa ish Me 08 00 01 08 07 08	0e:8c:28 Dst: 19 , Dst Po ge essage, 45 00 c0 a8 80 18	<pre>8), Dst: R 12.168.1.2 rrt: 1883, QoS Level -LR @ @ </pre>	aspberr Ø Seq: 4 : At mo	01:82 19, Ack st onc E- 	2:7e (b8:27 c: 5, Len:	24	
<pre>> Eth > Int > Tra > MQ > ></pre>	hernet ternet ansmis Teler Heade Msg L Topic Topic Messa 00 4 01 1 01 1	t II, t Prot ssion metry er Fla en: 2 : Leng :: top age: 7 27 eb 4c 52 14 cc f6 15	Src: ocol Contr Trans gs: 0 2 th: 1 ic/ba 36573 01 82 18 40 a6 07 a8 00	Tp-L Vers ol P port x30, 0 ru 616e: 7e 5b 00	inkT_0 ion 4, rotoco Proto Messa 206c61 18 a6 10 06 19 f5 31 01	696e f7 0 65 2 78 4 08 0	28 (192 cc Po Publ pe: 1 pe: 1 1 53 a cc	18:a4 .168 rt: 1 ish / Publi 28 (a8 (c2 (4c 2	6:f7:0 .1.8, 52390, Messag ish Me 08 00 01 08 07 08 38 2f	00:8c:28 Dst: 19 , Dst Po ge 2ssage, 45 00 c0 a8	8), Dst: R 92.168.1.2 ort: 1883, QoS Level	aspberr θ Seq: 4 : At mo e' ×AS	01:82	2:7e (b8:27 c: 5, Len:	24	



Figure. 3: Analyzing for the conventional topology

Wireshark · Follow TCP Stream (tcp.stream eq 7) · MQTT_raspi.pcapng

....MQTT...<..mosq/Yighgki6QYp610mTo6. mosquitto ...0..

Figure. 4: Captured packets from the conventional topology

B. Secured Transaction Experiment

As seen in Figures. 5 and 6, the proposed IoT topology is more secure in terms of data privacy. The dataSensor_sent from the publisher was already authenticated and encrypted by using base-64 format. Though Fernet cryptography is symmetric encryption method, this problem was addressed by using generate_key to keep rotating the secret key for each data released by the publisher.

To the best of authors' knowledge, it can also be seen in Figure. 6 that the data are no longer recognized as a plain text but a cipher text which can also eliminate the possibility of attacks in data integrity category.

No.		1 4 - 333	Destination		
140.	Time	Source	Descination	Protocol Le	ngth Info
	1065 24.26940832	5 192.168.1.18	192.168.1.16	TCP	74 50937 - 1883 [SYN] Seq=0 Win
	1066 24.26946142	6 192.168.1.16	192.168.1.18	TCP	74 1883 + 50937 [SYN, ACK] Seq=
	1100 25.27182465	3 192 168 1 16	192.168.1.18	TCP	74 [TCP Retransmission] 1883 -
					180 [TCP Previous segment not ca
1 6	1102 25.34752802	7 192.168.1.16	192.168.1.18	TCP	78 [TCP Window Update] 1883 + 5
	1107 25:46459721	4 192.168.1.18	192.168.1.16	TCP	80 [TCP Out-Of-Order] 50937 - 1
	1108 25.46463007	7 192.168.1.16	192.168.1.18	TCP	66 1883 + 50937 [ACK] Seq=1 Ack
	1113 25.46472317	5 192.168.1.16	192.168.1.18	PIQTT	70 Connect Ack
	1114 25.46482856	9 192.168.1.16	192.168.1.18	TCP	66 1883 + 50937 [FIN, ACK] Seq.
	1118 25.66302034	1 192.168.1.18	192.168.1.16		60 50937 + 1883 [RST] Seq=130 W
	1124 25.78810305	8 192.168.1.18	192.168.1.16		60 50937 + 1883 [RST] Seq=130 W
Tr			ort: 50937, Dst Port:	1883, Seq: 15,	Ack: 1, Len: 114
	Header Flags: 0 0011 = 00. = 0 = Msg Len: 112 Topic Length: 1 Topic: House/To	Message Type: Publi DUP Flag: Not set QoS Level: At most Retain: Not set L0 mp	Publish Message, QoS sh Message (3) once delivery (Fire a	nd Forget) (0)	once delivery (Fire and Forget)
~	Header Flags: 0 0011 = 0 = 00. = 0 Msg Len: 112 Topic Length: 1 Topic: House/Tr Message: 67414	bx30, Message Type: Message Type: Publi DUP Flag: Not set QoS Level: At most Retain: Not set Not set Not set Not set	Publish Message, QoS sh Message (3) once delivery (Fire a 18754c4a7746546f386338	nd Forget) (0) 59_	once delivery (Fire and Forget)
0000	Header Flags: 0 0011 e 00. = 00 = Msg Len: 112 Topic Length: 1 Topic: House/To Message: 674142 98 22 ef 8e d	b:30, Message Type: Message Type: Publi DUP Flag: Not set Qo5 Level: At most Retain: Not set 10 mp 141414142644f5a7a704 4 7b 1a 0b 42 bd ff	Publish Message, QoS sh Message (3) once delivery (Fire a 18754c4a7746546f386338 5 3c 08 00 45 00 -*-	nd Forget) (0) 59_	once delivery (Fire and Forget)
0000	Header Flags: 0 0011 = 00. = 00 = Msg Len: 112 Topic Length: 1 Topic: House/Tr Message: 67414	bx30, Message Type: Message Type: Publi DUP Flag: Not set QoS Level: At most Retain: Not set 10 mp 14114142644f5a7a784 5 7b 1a 0b 42 bd f6 9 00 40 06 68 14 ct	Publish Message, QoS sh Message (3) once delivery (Fire an 18754c4a7746546f386338 5 3 c 08 00 45 00 -** 0 a8 01 12 c 0 a8 -**	nd Forget) (0) 59_ (- B < E	once delivery (Fire and Forget)
0000	Header Flags: (0011 = 00. = 00. = Msg Len: 112 Topic Length: 1 Topic: House/T. Message: 67414 98 22 ef 8e d 00 a6 ce ca 4 01 10 c6 f9 0	hx30, Hessage Type: Publi Message Type: Publi DUP Flag: Not set QoS Level: At most Retain: Not set 10 mp 141414142644f5a7a7044 17 b 1a 0b 42 bd f6 10 00 40 66 e8 14 cC 10 59 aa 10 59 aa	Publish Message, QoS sh Message (3) once delivery (Fire an 18754c4a7746546f386338 5 3c 08 00 45 00 -** a a8 01 12 c0 a8 -** b be 44 08 09 19 -**	nd Forget) (0) 59_ (- B < E @ @ - [- Y	once delivery (Fire and Forget)
0000	Header Flags: 0011 = 00. = 00. = Msg Len: 112 Topic Length: Topic: House/T Message: 67414 98 22 ef 8e d 00 a6 ce ca 4 01 10 c6 f9 0 00 ef 22 37 00	bx30, Hessage Type: Message Type: Publi CUP Flag: Not set QoS Level: At most Retain: Not set 00 mp 141414142644f5a7a704 5 7b 1a 0b 42 bd ff 0 00 40 06 e 81 4 ct 7 5b 1a el b0 59 aŭ 0 00 10 10 80 aŭ	Publish Message, QoS sh Message (3) once delivery (Fire al 18754c4a7746546f386338 5 ac 08 00 45 00 - * a 80 11 2 c 0 a8 - * 5 be d4 00 80 19 c 5 46 64 fb 86 - *	nd Forget) (0) 59 	once delivery (Fire and Forget)
0000 0010 0020 0030 0040	Header Flags: 0 001 = 00. = 00. = Msg Len: 112 Topic Length: 1 Topic: House/Tr Message: 67414 98 22 ef &ed 00 a6 ce ca 44 01 10 c6 f9 0 00 e5 22 37 0 02 48 80 70 0	hx30, Hessage Type: Publi DUP Flag: Not set QoS Level: At most Retain: Not set 10 mp 141414142644f5a7a704 4 7b 1a 0b 42 bd ff 8 00 40 66 e8 14 ct 5 %b1 ae 1b 59 af 9 00 e1 01 68 0a 11 00 43 06 75 75 73 05	Publish Hessage, QoS sh Pessage (3) once delivery (Fire at 18754c4a7746546f386338 5 3c 08 00 45 00 * a 88 01 12 c0 a8 b 64 d0 80 19 c1 c5 46 c4 fb 86 19 c7 56 65 66 70 ****	nd Forget) (0) 59 -(- B. < E. @.0 -(- Y 7 F 7 F	once delivery (Fire and Forget)
0000 0010 0020 0030 0040 0050	Header Flags: 6 0011	hx30, Hessage Type: Message Type: Publi QDF Flag: Not set QoS Level: At most Retain: Not set 10 mp 441414142644f5a7a704 5 7b 1a 0b 42 bd ff 9 00 40 06 e8 14 cd 7 5b 1a e1 b0 59 ai 9 00 e1 01 08 0a 12 3 0a 43 6f 75.73 05	Publish Hessage, QoS sh Message (3) once delivery (Fire a 18754c4a7746546f386338 5 to 08 09 45 00 8 48 01 12 co 48 5 to 48 00 19 1 c 3 46 e4 16 06 2 4 5 4 5 6 6 3 7 5 4 6 5 6 6 7 10 70 48 75 3 4 2 10 74 4 75 4 4 4	nd Forget) (0) 59 	once delivery (Fire and Forget)
00000 0010 0030 0040 0050 0050	Header Flags: 0011	hx30, Hessage Type: Publi DOP Flag: Not set QoS Level: At most Retain: Not set 10 mp 141414142644f5a7a704 4 7b 1a 0b 42 bd ff 9 00 40 66 e8 14 cc 9 00 40 66 e8 14 cc 9 00 101 68 0a 14 9 00 101 68 0a 14 9 00 101 68 0a 14 9 00 41 67 77 73 05 141 42 64 45 5a 72	Publish Hessage, QoS sh Pessage (3) once delivery (Fire al 18754c4a7746546f3866388 5 3c 08 00 45 00 a 80 01 12 c0 a8 15 be 4d 40 80 19 c 5 46 c4 fb 86 19 c 75 45 c5 60 70 c 70 43 75 42 c4 50 77 2 c6 c3 51 77 77 c7 c7 77 c7 77 c7 77 c7 77 c	nd Forget) (0) 59	once delivery (Fire and Forget)
0010 0010 0010 0010 0010 0010 0010 0050 0050	Hender Flags: (0011	$ \begin{array}{llllllllllllllllllllllllllllllllllll$	Publish Hessage, QoS sh Message (3) once delivery (Fire a 18754c4a7746546f386338 5 1c 08 00 45 00 8 40 112 c 08 1 be d4 00 80 19 1 c 36 64 4f 86 2 c 46 64 6f 86 3 7f 54 65 64 76 1 58 77 27 6e 39 1 58 77 72 6e 39 1 58 77 75 77 75 75 75 75 75 75 75 75 75 75	nd Forget) (0) 59_ -{- B<-E @-B	once delivery (Fire and Forget)
0010 0010 0010 0010 0010 0010 0010 0050 0050 0050 0050	Header Flags: (0011	Noill, Hessage Type: Message Type: Publi DUP Flag: Not set QOS Level: At most Hetain: Not set U0 The set H0 The set H0 The set H0 The set H14141412644f5a7a704 A 7b 1a 0b 42 bd ff 9 00 01 01 68 0a 10 59 af 9 00 01 01 68 0a 10 59 af 9 00 01 01 68 0a 10 77 33 05 1 41 42 64 4f 5a 77 30 61 61 31 39 07 71 33 5 an 65 54 46 6d 37 13 5 36 57 43 72	Publish Hessage, QoS sh Pessage (3) once delivery (Fire at 18754c4a7746546f386538 5 3c 08 00 45 00 a 88 01 12 c0 at b c4 d0 80 19 c 54 66 4f b8 60 19 c 75 46 c5 60 70 70 43 77 54 c2 45 15 17 72 66 37 c1 53 77 54 c2 45 15 17 77 55 00 74 c1 54 75 42 c4 51 77 75 50 27 44 c5 46 20 51 c5 46 20 50 51 c5 50 50 50 50 c5 50 50 50 c5 50 50 50 c5 50 50 50 c5 50 50 c	nd Forget) (0) 59_ (- B c - E @ @ [- Y - F p - Ho use / Teep AMAG 022, pHul 7 olic SY 6330arr0 00120T F 0531000	once delivery (Fire and Forget)
0000 0010 0030 0040	Hender Flags: : 0011	No.80, Herssage Type: Message Type: Publi DUP Flag: Not set DUP Flag: Not set Go Level: At most Retain: Not set H0 Plau D0 At most Retain: Not set H0 Data At most MP Attaint4142644f5a7a7044 17 bh a 0b 42 bd ff D0 40 00 00 at 14 cd So 414 cd D 00 40 00 00 at 14 cd So 414 cd D 00 41 00 00 at 14 cd So 414 cd D 00 41 00 00 at 14 cd So 414 cd D 00 42 00 00 00 at 14 cd So 414 cd D 00 42 00 00 at 14 cd So 414 cd D 00 41 00 00 at 14 cd So 414 cd D 00 42 00 10 cd So 414 cd D 00 41 00 10 cd So 414 cd So 415 42 64 75 73 60 cd So 415 cd So 415 42 64 73 73 73 64 cd So 414 cd So 415 43 67 37 30 74 37 73 So 414 cd So 51 55 44 30 33 74 33 72 So 43 30 33	Publish Message, QoS sh Message (3) once delivery (Fire a Message (3) be do not show the show the show the show the show the show the show the show the show the show the show the show the show the show the show the show the show the show	nd Forget) (0) 59	once delivery (Fire and Forget)

Figure. 5: Analyzing for the secured topology



Figure. 6: Captured packets from the secured topology

C. Discussion of Experiment

The proposed secure web-socket topology was indeed proved to be twice bigger than the conventional topology by examining data length during the communication between publisher and subscriber. Upon the product specification of Raspberry Pi 3 Model B, the cryptography computation of encrypted data for 180 bytes (as depicted in Figure. 5) still can be addressed by Quad Core 1.2GHz Broadcom BCM2837 64bit CPU. To the best of authors' knowledge, this proposed topology should be implemented for future secure end to end communication of IoT devices.

Based on the two experiment categories, the basic con-figuration or features provided by vendors should be given more attention during the deployment of IoT devices. With the increasing number of sensor nodes that will serve the consumers, it is very important to ensure the security of the network e.g., optimizing the configuration, pen-testing the networks, and implementing third party authentication. This research proposes autokeying either encryption or decryption in ensuring integrity of the passing data (as shown in Figure. 6). In accordance with the previous explanation, it is proven that the proposed IoT topology is more secure compared to conventional IoT topology which only uses MQTT protocol.

V. CONCLUSION

In this research, the feasibility of MQTT were implemented and analyzed to enable secure communication for IoT devices with real testbed. However, it also showed the drawback of MQTT



protocol when the publisher sends the topic to throughout subscriber web platform. As recommendation, this research proposed a secure web-socket IoT topology between publisher and subscriber combined with MQTT protocol for IoT devices. Moreover, this research also suggested the use of encryption and decryption to enhance security of end to end communication in the implementation of smart hospital, smart home, smart factory, etc. Further security analysis of secure web-socket for IoT devices under different attack scenarios might be considered. This tesbed research was based on the internal threat, future research needs to scrutinize the external threat and use different vendor IoT devices.

VI. ACKNOWLEDGMENT

This research was collaborated between School of Ap-plied Science and School of Electrical Engineering, Telkom University. This research was also funded by PPM, Telkom University.

REFERENCES

- [1].Lawrence Miller. Iot security for dummies, carrie a. johnson, ed, 2016.
- [2].Interpol. Digital security challenge. https://tinyurl.com/yyjz6g2r, Febru-ary 2018. Accessed: 2019-07-07.
- [3].Roy Fielding, Jim Gettys, Jeffrey Mogul, HenrikFrystyk, Larry Mas-inter, Paul Leach, and Tim Berners-Lee. Hypertext transfer protocol- http/1.1, 1999.
- [4].Peter Saint-Andre et al. Rfc 3920: Extensible messaging and presenceprotocol (xmpp): Core. Internet Engineering Task Force, 2004.
- [5]. Andrew Banks and Rahul Gupta. Mqtt version 3.1. 1. OASIS standard, 29:89, 2014.
- [6]. John O'Hara. Toward a commodity enterprise middleware. Queue, 5(4):48–55, May 2007.
- [7].Z. Shelby, A. P. Castellani, and C. Bormann. Coap: An application protocol for billions of tiny internet nodes. IEEE Internet Computing, 16(02):62–67, mar 2012.

- [8]. T. Yokotani and Y. Sasaki. Comparison with http and mqtt on required network resources for iot. In 2016 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC), pages 1–6, Sep. 2016.
- [9]. N. Naik. Choice of effective messaging protocols for iot systems: Mqtt, coap, amqp and http. In 2017 IEEE International Systems Engineering Symposium (ISSE), pages 1–7, Oct 2017.
- [10]. Gaston C Hillar. MQTT Essentials-A Lightweight IoT Protocol. Packt Publishing Ltd, Birmingham, UK, 2017.
- [11]. D. Dragomir, L. Gheorghe, S. Costea, and A. Radovici. A survey on secure communication protocols for iot systems. In 2016 International Workshop on Secure Internet of Things (SIoT), pages 47–62, Sep. 2016.
- [12]. Hyun Cheon Hwang, JiSu Park, and Jin GonShon. Design and implementation of a reliable message transmission system based on mqtt protocol in iot. Wireless Personal Communications, 91(4):1765–1777, Dec 2016.
- [13]. S. Shin, K. Kobara, Chia-Chuan Chuang, and Weicheng Huang. A secu-rity framework for mqtt. In 2016 IEEE Conference on Communications and Network Security (CNS), pages 432–436, Oct 2016.
- [14]. K. Fysarakis, I. Askoxylakis, O. Soultatos, I. Papaefstathiou, C. Man-ifavas, and V. Katos. Which iot protocol? com-paring standardized approaches over a common m2m application. In 2016 IEEE Global Communications Conference (GLOBECOM), pages 1–7, Dec 2016.
- [15]. M. Singh, M. A. Rajan, V. L. Shivraj, and P. Balamuralidhar. Secure mqtt for internet of things (iot). In 2015 Fifth International Conference on Communication Systems and Network Technologies, pages 746–751, April 2015.
- [16]. S. Andy, B. Rahardjo, and B. Hanindhito. Attack scenarios and security analysis of 8237



mqttcommunication protocol in iot system. In 2017 4th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI), pages 1–6, Sep. 2017.

- [17]. KrešimirGrgic,' Ivan Špeh, and Ivan Hedi⁻. A web-based iot solution for monitoring data using mqtt protocol. In 2016 International Conference on Smart Systems and Technologies (SST), pages 249–253. IEEE, 2016.
- [18]. AimaschanaNiruntasukrat, ChaveeIssariyapat, PanitaPongpaibool, KoonlachatMeesublak, PramrudeeAiumsupucgul, and AnunPanya. Authorization mechanism for mqtt-based internet of things. In 2016 IEEE International Conference on Communications Workshops (ICC), pages 290–295. IEEE, 2016.