Vol.2020-MBL-94 No.42
Vol.2020-UBI-65 No.42
2020/3/3

A Study of Secure Communication Protocols

Zongxin Wang[1,a)] Kazuya Sakai[1,b)]

Abstract: MQTT is widely used as a light-weight communication protocol for the Internet of Things (IoTs). However, MQTT has a security issue in its strings handling. To be specific, should a packet contain malicious codes or invalidate characters, attackers can exploit the discrepancy between validating and non-validating nodes to keep nodes offline. In addition, this can be used to launch DoS attacks and/or other security attacks. To address this issue, we propose a secure communication protocols, called S-MQTT, using light-weight elliptic curve-based cryptography. The proposed scheme can not only detect the invalidating codes, but also protect data privacy. The experimental results demonstrate the security and efficiency of the proposed scheme.

Keywords: MQTT, ECC, IoT

1. Introduction

We are currently witnessing everything being connected to each other, and the Internet of the Things (IoT) market is rapidly growing. Computing devices with sensing and communication capabilities, called IoT devices, generate data, with which a number of computing services, such as environmental monitoring [1], smart home [2], smart transportation [3], and so on, have been developed.

Message Queuing and Telemetry Transport (MQTT) is widely used as a publish/subscribe-based protocol which is simple, light, and energy efficient. With the development of MQTT, it has been applied to lots of scenario. For example, some mobile monitoring system adopts MQTT protocol to meet the requirements of health monitoring and management [4]. Besides this, The MQTT system has been used to gain high-quality and reliable data to study physics as parameters of environment condition [5]. However, it also has some problems, for an example that MQTT protocol only provides authentication for security mechanism but does not encrypt the data in transit [7]. Besides, security problems are concerned due to focusing on service-oriented data sharing and processing rather than point-to-point data collection [8]. For the popular clients, lots of them support SSL/TLS protocol, howerver, some scholar take consideration that the cost of SSL/TLS protocol is relatively big and this run counters to the idea of MQTT. In the current study, the lightweight elliptic curve cryptography has been used in MQTT. The main advantage of the elliptic curve cryptography (ECC) is the performance advantage [9]. ECC compared with RSA, at the same calculating time, using much smaller bits of key. To address security problem of the Variable Header containing username and password flag, MQTT protocol based on lightweight attribute based encryption over elliptic curves has been proposed [10]. Though this method solves the problem of data security, the disallowed codes or characters which are receiving from the receiver may close the network connection. If a broker does not implement checks for disallowed code points or characters and clients do (or vice versa), a malicious client could exploit this discrepancy to disconnect other clients by sending invalidly encoded strings [11].

Hence in this direction, we propose a simplified publish and subscribe-based IoT protocol based on lightweight elliptic curve cryptography, secure MQTT, called S-MQTT. In different encoding environment, due to irreversible process of encoding and decoding, the encrypted string codes will not be exactly same after decryption during the transmission process. Specifically, the sender sends the encrypted data by kind of encoded mode, and the receiver receives the data and uses key to decrypt it. If the receiver whose encoded mode is different from the sender, the decrypted data will not be same as sending data. Through this, we can detect the invalidating codes or characters. The merit of using lightweight elliptic curve cryptography is because the key size of the ECC algorithm is short so it takes up less storage space, lower CPU overhead and occupies less bandwidth that ECC owns better performance. Besides this, ECC algorithm is more suitable to wireless environment. With the development of IoT, the vast majority IoT devices will be deployed in wireless environment. The ECC encryption algorithm provides a better customer experience for wireless environment. According to this secure protocol, the invalid symbol security detection issues of heterogeneous network of MQTT could be solved. The proposed secure protocol is feasible, efficient and stable.

The main contribution of the paper is to: (i). We study the feasibility of using lightweight elliptic curve to enable security of S-MQTT. (ii). We design the S-MQTT protocol. (iii). We evaluate the performance analysis of S-MQTT protocol.

The rest of paper is organized as follows. Section 2 reviews the preliminary. In Section 3, S-MQTT Control Packet is proposed.

[1] Tokyo Metropolitan University, 6-6 Asahigaoka, Hino, Tokyo, 191-0065, Japan
[a)] wang-zongxin@ed.tmu.ac.jp
[b)] ksakai@tmu.ac.jp

Section 4 provides security analyses for the correctness of our S-MQTT Control Packet, and Section 5 evaluates the performance of the proposed scheme. Section 6 concludes this paper.

2. Preliminary

The Internet of Things (IoT) usually refers to a world-wide network of interconnected heterogeneous objects (sensors, actuators, smart devices, smart objects, RFID, embedded computers, etc) uniquely addressable, based on standard communication protocols. And with the development of 5G, IoT technology will be more focused on [6].

Message Queuing Telemetry Transport (MQTT) written by IBM, is an open OASIS, ISO standard and client server publish/subscribe messaging transport protocol. It is light weight, open, simple, and designed to be easy to implement. These characteristics make it ideal for use in many situations, including constrained environments such as for communication in Machine to Machine (M2M) and Internet of Things (IoT) contexts [12].

Publish - subscribe system is a messaging pattern and it is comprised of information producers who publish and information consumers who subscribe to information. It provides a simple and effective method for disseminating data while maintaining a clean decoupling of data sources and data sinks. In the publish - subscribe system, information producers submit data as publications to the system and information consumers indicate their interests by submitting subscriptions. A subscription has a notification set, which is a set of potential publications that would match the subscription. On receiving a publication, the broker determines the subset of matching subscriptions and notifies the appropriate subscribes [13].

Elliptic curve cryptography (ECC) is an approach to asymmetric cryptography based on the algebraic structure of elliptic curves over finite fields. The merit of ECC is that smaller key can ensure relatively higher level security, and the speed of ECC is faster than non-EC cryptography based on the similar environment [14].

3. Proposed S-MQTT Control Packet

In this section, we discuss the overview in section 3.1. Section 3.2 takes about the packet format of original MQTT protocol and proposed S-MQTT packet format. Section 3.3 refers to the action that each entity shall take.

3.1 Overview

OASIS has now published the official MQTT v5.0 standard. This protocol runs over TCP/IP, or over other network protocols that provide ordered, lossless, bidirectional connections.One of its feature is that use of the publish/subscribe message pattern which provides one-to-many message distribution and decoupling of applications. An MQTT Control Packet consists of up to three parts, Fixed Header, Variable Header and Payload.

3.2 Packet Format

An MQTT Control Packet contains Fixed Header and Remaining Length. Among Remaining Length, it includes Variable Header and Payload and the bytes size of them are not fixed, as shown below (Table 1).

Table 1 MQTT Fixed Header Format

MQTT Fixed Header								
Bit	7	6	5	4	3	2	1	0
Byte 1	Control Packet type				Flag specific			
Byte 2...	Remaining Length							

Remaining Length +1~4 bytes	
Variable Header 0~N bytes	Payload 0~X bytes

Because MQTT protocol is a lightweight protocol, the MQTT Control Packet are encoded as UTF-8 strings. The reason is that UTF-8 is an efficient encoding of Unicode characters that optimizes the encoding of ASCII characters in support of text-based communications [12].

According to the standard, the character data in UTF-8 encoded string MUST be well-formed UTF-8 as defined by the Unicode specification and restated in RFC 3629. If the client or server receives an MQTT Control Packet containing ill-formed UTF-8 it is a Malformed Packet and will close the network connection.
In order to solve the similar security problem about illegal codes or characters resulting in disconnecting network or even worse, we propose S-MQTT protocol which is secure part of original MQTT protocol. We redefine the reserved MQTT Control Packet type " 0000 " as DETECT. Because after a Network Connection is established by a client to a server, the first packet sent from the client to the server MUST be a CONNECT packet, the DETECT packet should be sent as the second packet. The purpose of DETECT is detecting illegal codes or characters and doing minimum encryption. The proposed DETECT Fixed Header is described as below (Table 2).

Table 2 Proposed DETECT packet Fixed Head

DETECT								
Bit	7	6	5	4	3	2	1	0
Byte1	Control Packet type(0)				Flag	ECC Validation		Passing Path
	0	0	0	0	X	X	X	X
Byte2..	Remaining Length							

In order to be consistent with the original protocol, we defined the value, direction of flow and description.

3.2.1 Passing Path

Position : byte 1, bit 0
Passing path is defining Direction of flow of control packet message. The passing path is shown below (Table 3).
When passing path is set to 0, it indicates that the server sends data to client (server to client).
When passing path is set to 1, it indicates that the client sends data to server (client to server).

Table 3 Passing Path

Passing Path	
Bit0	Description
0	server sends data to client
1	client sends data to server

3.2.2 ECC and Validation

ECC and Validation is used for detecting invalidated codes and satisfying the data privacy. The ECC and Validation is shown below (Table 4).
Position : byte 1, bits 2-1
First of all, the server chooses an elliptic curve and we can know the public key and private key.
In step 1 of ECC and Validation, the client uses the public key to encrypt the data [M] then send timestamp and ciphertext to server.
In step 2 of ECC and Validation, the server saves the timestamp first, then uses private key to decrypt the ciphertext and get plaintext.
In step 3 of ECC and Validation, the server uses the private keys, adopts digital signature to signature the data [M] which have been received and sends the signature data and timestamp to the client.
In step 4 of ECC and Validation, the client uses the public key to validate the data [M] and send compared message to server.

Table 4 ECC and Validation

ECC Validation			
ECC and Validation	Bit 2	Bit 1	Description
0	0	0	*Enc.* and send data
1	0	1	*Dec.* and save data
2	1	0	*Sig.* decrypted data
3	1	1	*Ver.* and send message

3.2.3 Flag

Flag is used for containing several parameters specifying the behavior of the MQTT connection. The flag is shown below (Table 5).
Position : byte 1, bits 3
When the Bit 2 is set to 0, we will ignore the flag (Bit 3) and always set it to 0.
When the Bit 2 is set to 1, we can not ignore the flag.
When Bit 3 equals to 0, it means the garbled after decoding.
When Bit 3 equals to 1, it means the original code after decoding.

3.3 The action that each entity shall take

In our assumptions, the length of sending data in publish-subscribe system meets the size of sending encoded string. For simplicity, it is assumed that the client and server can communicate and the hash function $Hash(x)$ is assumed to be collision resistant. Besides this, we also assume that the client and server can hold the public key safely and the server can hold the private key safely. We will use client to send plaintext message to server by broker.

The notations used in this paper are listed in Table 6.

The client will send plaintext message (M) and we could not know the sending characters will be satisfied with UTF-8 encoded code or not. The plaintext message (M) will be encrypted by lightweight elliptical curve cryptography and it becomes ciphertext message (C). The client will send timestamp and $Hash(M)$ to server, as the same time, when it sends the ciphertext message to server. Via broker, the server will receive the ciphertext message, $Hash(M)$ and receive timestamp. Immediately later, the server uses its private key to decrypt the ciphertext message (C), we can name it plaintext message 2 (M') temporarily and the server will calculate the $Hash(M')$, after this it will compare the $Hash(M)$ and $Hash(M')$. And now, the server will know whether the twice plaintext message are same or not and the server can make right judgment. Subsequently, the server will use digital signature for $Hash(M')$. Then the server will send the digital signature data (σ) and timestamp to client by broker. After this, the client uses its public key to verify the received signature data (σ) and calculates $Hash(M)$ to compare original data and validated data whether same or not. Besides this, it also will compare the difference between these two timestamps. We can set the time difference as a fixed value, since the timestamp is real time, when the difference is too big we can recognize that the data have tampered with broker or there is an excessive network latency, for safety reasons, we will disconnect this access. If the verified results are as same as before, it will continue to run publish phase following the original MQTT protocol, else if, it will disconnect this access and wait for next access.
Among the process, there are some things should be noted.
1). We assume that one of client and server does not meet UTF-8 code encoded. When the sending characters are not UTF-8 codes encoded, through the above operation, the validated results will be different from the original data.
2). About the timestamp, because of real time, after numerous experiments, it is high time that we can set the difference time as a stable value.
3). About the broker, in the original MQTT protocol, the malicious broker can send invalidated codes to server. However, it does not use encryption technology, the sent data can not be recognized by server normally and the data from client can not be tampered. So, in our simulation, we do not take broker into account. Fig. 1 shows the run flowchart of S-MQTT.

It is similar to the original protocol that there are three parts in

Table 5 Flag

Flag	
Bit0	Description
0	Compare the UTF-8 Library and appear mojibake
1	Compare the UTF-8 Library and not appear mojibake

Table 6 Definition of Notations

Symbol	Meaning
M,M'	plaintext message
C	ciphertext message
pk	public key
sk	private key
σ	message of digital signature data
$Enc(.), Dec(.)$	the encryption and decryption functions
$Sig(.)$	the digital signature function
$Ver(,)$	the validation function about signature
$Hash(.)$	the hash function

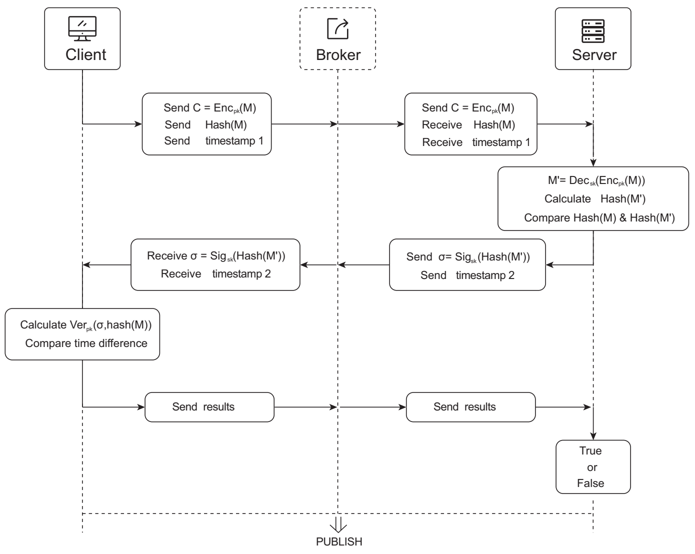


Fig. 1 The flowchart of S-MQTT protocol

the S-MQTT protocol: (i) The client publishes the data. (ii) The broker forwards the data. (iii) The server receives the data. And there are three phases in the proposed protocol. In Setup Phase selecting elliptic curve and key management are done. During encrypt and decrypt phase, it will make data privacy and safety judgments. In publish phase, it will determine to whether publish the data following the original protocol or not.

(i). Setup Phase

- Selecting an elliptic curve, server and client hold the public key simultaneously. And only server holds the private key.

(ii). Encrypt and Decrypt Phase

- The client encrypts data using public keys and then sends encrypted data and timestamp 1 to the server.
- The server receives the encrypted data and timestamp 1, after that it uses private key to decrypt the data which have been encrypted. After this, the server stores the encrypted data and timestamp 1.
- The server uses its private key to do digital signature on the encrypted data, then sending the signature data and timestamp 2 to the client.
- The client uses the public key to validate the data from server, then comparing the data which have sent from client by itself and the data which have been signature from the server, besides this it also will compare the timestamp from the two sides.
- The client makes time difference between timestamps, as an auxiliary function, distinguishing sending and receiving time and judging if the time difference is too long the data has been tampered.
- The client will come to conclusion, comparing the data is as same as before or not, and send the results to server. So the server will also get conclusion as same as client.

(iii). Publish Phase

- If the decrypted message is as same as plaintext, it will follow the original MQTT protocol.
- If the decrypted message is different from plaintext, it will disconnect this access and wait for new access.

4. The Security Guarantee of MQTT

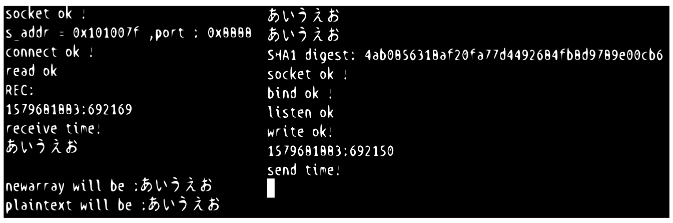
In this section, we first describe the security requirements for MQTT control packets. Then we discuss the security about receiving the invalidating characters from malicious client. At last, we talk about the qualitative security analysis.

4.1 Security Requirements

The MQTT Control Packets are encoded as UTF-8 strings and the maximum size of a UTF-8 encoded string is 65,535 bytes. If the client or server receives an MQTT Control Packet containing ill-formed UTF-8, it will lead to closing the network or even worse, such as DoS attacks. To solve this point, we propose adopting client sends the encyrpted data via broker, after server decrypting the data then forwarding the digital signature data via broker. By using this, in different coding environments, sending and receiving data may be different, so not only it can detect the invalidating codes, but also can protect data privacy.

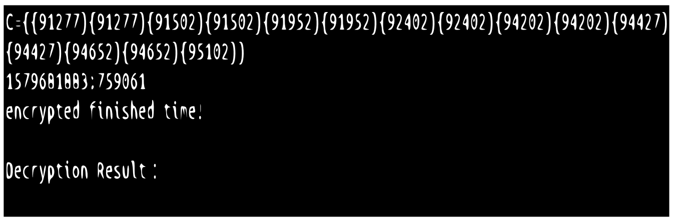
4.2 Security about Invalidating Characters

We assume one of the client or server obeys the UTF-8 encoding. It may be assumed here that the client meets the standards and the server dose not follow the standards. The client sends the encrypted UTF-8 code data to server via broker. For example, the client will send CJK characters, such as “ aiueo ” (shown as Fig. 2), to server. Because of UTF-8 code, standards compliant, the server can receive and decrypt it completely and correctly. Because CJK characters in UTF-8 occupy 3 Octet sizes, in other words, they occupy 3 bytes. For just sending CJK characters, it will occupy 15 points on the elliptic curve, however, because the server could not distinguish these characters, they can not been encoded and displayed (shown as Fig. 3). And these characters can only be presented at the curve as point. Due to irreversible process of encoding and decoding, the server can not store the original data and forward digital signed data correctly.



```
socket ok !                             あいうえお
s_addr = 0x101007f ,port : 0x8888       あいうえお
connect ok !                            SHA1 digest: 4ab0856318af20fa77d4492684fb8d9789e00cb6
read ok                                 socket ok !
REC:                                    bind ok !
1579681883:692169                       listen ok
receive time!                           write ok!
あいうえお                               1579681883:692150
                                        send time!
newarray will be :あいうえお              █
plaintext will be :あいうえお
```

Fig. 2 Sending UTF-8 code



```
C={(91277)(91277)(91502)(91502)(91952)(91952)(92402)(92402)(94202)(94202)(94427)
(94427)(94652)(94652)(95102)}
1579681883:759061
encrypted finished time!

Decryption Result :
```

Fig. 3 Receiving UTF-8 code

On the other hand, if both server and client do not obey the standard, because both of them let invalid character pass, the issue is avoided. And if both server and client follow the standards, because of meeting the standards, the issue is also avoided.

4.3 Qualitative Security Analysis

In this section, we analyze how S-MQTT achieves the security requirements.

Data Security—Privacy of data can be ensured because the data has been encrypted by lightweight elliptic curve cryptography.

Detection of invalidating characters—Because encoding and decoding process is irreversible, through encryption and decryption, in different encoding environment, the resulting data may be different from the original data so that can detect invalidating characters.

DoS attack resistance—In this attack, about original protocol, an adversary client can keep all clients offline with a single malicious message. But in the S-MQTT protocol, the server can make judgment by comparing the value of two hash functions. So it can defend the DoS attack caused by invalidated codes.

So, in our proposed S-MQTT protocol, if one of the server and client is malicious, and another one is honest, we can detect it and report error. If both of them are malicious or both of them are honest, we could not take them into consideration, because in the original protocol, they can communicate normally.

5. Performance Analysis

We evaluate and compare the performance of S-MQTT based on our proposed scheme. In order to meet accuracy of experiment, for the different parameters, we do numerous experiments on encoding delay, encrypting delay and decrypting delay. Test Table for evaluating the performance of the S-MQTT protocol are described in Table 7.

Table 7 System details

Hardware	Intel Core i7-7500U CPU @ 2.70GHz
Memory	2 GB
Operating System	Windows 10, 64bit, Ubuntu 14.04
Gcc Version	4.8.4 (Ubuntu 4.8.4)

In the experiment, we send message size from 2~200 bytes and receive the message size from 8~1322 bytes.

Performance of encoding algorithms of our proposed scheme is depicted in Fig. 4 when the sending data is too small, we do numerous simulation about it and it shows large fluctuations; When the sending data is getting more larger, curve tends to be stable. And the other two figures are also like this. This figure depicts the delay that sending data encodes to the elliptic curve.

Performance of encrypting delay of our proposed scheme is depicted in Fig. 5 and this figure shows the delay that the encryption time of encoded messages.

Performance of decrypting delay of our proposed scheme is depicted in Fig. 6 and this figure shows the delay that the decrypting time of encoded messages.

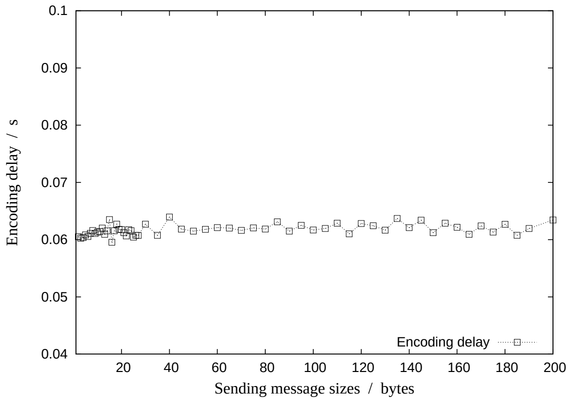


Fig. 4 Encoding delay

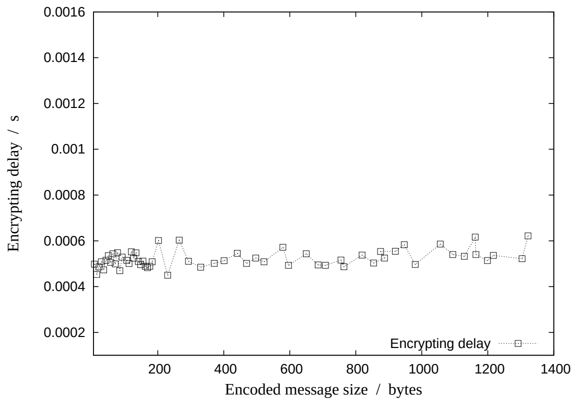


Fig. 5 Encrypting delay

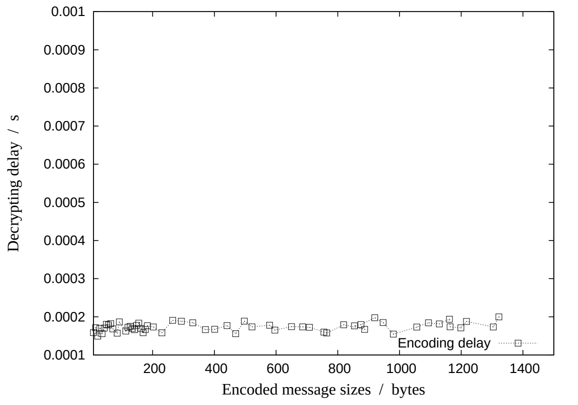


Fig. 6 Decrypting delay

Based on these experiment data, the encoding delay takes up approximately 0.064 second. And the sum of duration about encrypting delay and decrypting delay is about 0.00064 second. The simulation meets our requirements.

6. Conclusion

The original MQTT protocol as a lightweight publish / subscribe IoT protocol more or less has some problems. In our proposed S-MQTT protocol, by using lightweight elliptic curve cryptography, in different encoding environment, not only can we protect data privacy, but also can avoid the control packet which contains malicious codes and characters effectively. Besides, we also can prevent a series of problem caused by malicious client, such as DoS attack. Experiments results based on the simulation demonstrates the security and efficiency of our proposed scheme.

References

[1] Lazarescu, Mihai T: Design of a WSN platform for long-term environmental monitoring for IoT applications, *IEEE Trans. Emerg. Sel. Topics Circuits Syst.*, Vol.3, No.1, pp.45–54 (2013).

[2] Sivaraman, Vijay and Gharakheili, Hassan Habibi and Vishwanath, Arun and Boreli, Roksana and Mehani, Olivier: Network-level security and privacy control for smart-home IoT devices, *IEEE Trans. WiMob.*, pp.163–167 (2015).

[3] Xie, Xiao-Feng and Wang, Zun-Jing: Integrated in-vehicle decision support system for driving at signalized intersections: A prototype of smart IoT in transportation., No. 17-00671, (2017).

[4] Yi, Ding and Binwen, Fan and Xiaoming, Kong and Qianqian, Ma: Design and implementation of mobile health monitoring system based on MQTT protocol, *IEEE Trans. IMCEC.*, pp.1679–1682 (2016).

[5] Atmoko, RA and Riantini, R and Hasin, MK: IoT real time data acquisition using MQTT protocol, *IOP Publishing Trans.* , Vol.853, No.1, pp.012003 (2017).

[6] Fortino, Giancarlo and Trunfio, Paolo: Internet of things based on smart objects: Technology, middleware and applications, *Springer Trans.* , (2014).

[7] Andy, Syaiful and Rahardjo, Budi and Hanindhito, Bagus: Attack scenarios and security analysis of MQTT communication protocol in IoT system, *IEEE Trans. EECSI.*, pp.1–6 (2017).

[8] Chen, Jiachen and Li, Sugang and Yu, Haoyang and Zhang, Yanyong and Raychaudhuri, Dipankar and Ravindran, Ravishankar and Gao, Hongju and Dong, Lijun and Wang, Guoqiang and Liu, Hang: Exploiting ICN for realizing service-oriented communication in IoT, *IEEE Trans. Communications Magazine.*, Vol.54, No.12, pp.24–30 (2016).

[9] Lauter, Kristin: The advantages of elliptic curve cryptography for wireless security, *IEEE Trans. Wireless Commun.*, Vol.11, No.1, pp.62–67 (2004).

[10] Singh, Meena and Rajan, MA and Shivraj, VL and Balamuralidhar, P: Secure mqtt for internet of things (iot), *IEEE Trans. Communication Systems and Network Technologies.*, pp.746–751 (2015).

[11] Maggi, Federico and Vosseler, Rainer and Quarta, Davide: The fragility of industrial IoT's data backbone, *Trend Micro Inc Trans.*, https://www.trendmicro.com/vinfo/us/security/news/internet-of-things/mqtt-and-coap-security-and-privacy-issues-in-iot-and-iiot-communication-protocols, (2018).

[12] Andrew Banks, Ed Briggs, Ken Borgendale, and Rahul Gupta: MQTT Version 5.0, *OASIS Standard Trans.*, (2019).

[13] Fidler, E and Jacobsen, HA and Li, G and Mankovski, S: Publish/Subscribe System, *IOS Press Trans. Feature Interactions in Telecommunications and Software Systems VIII.*, pp.12 (2005).

[14] Tirthani, Neha and Ganesan, R: Data Security in Cloud Architecture Based on Diffie Hellman and Elliptical Curve Cryptography, *IACR Trans. Cryptology ePrint Archive.*, Vol.2014, pp.49 (2014).