Lightweight Security Mechanism over MQTT Protocol for IoT Devices

Sanaz Amanlou[1], Khairul Azmi Abu Bakar[2]
Center for Cyber Security, Faculty of Information Science & Technology
Universiti Kebangsaan Malaysia, Bangi, Malaysia

***Abstract*—Security is one of the main concerns with regard to the Internet of Things (IoT) networks. Since most IoT devices are restricted in resource and power consumption, it is not easy to implement robust security mechanisms. There are different methods to secure network communications; however, they are not applicable to IoT devices. In addition, most authentication methods use certificates in which signing and verifying certificates need more computation and power. The main objective of this paper is to propose a lightweight authentication and encryption mechanism for IoT constrained devices. This mechanism uses ECDHE-PSK which is the Transport Layer Security (TLS) authentication algorithm over Message Queuing Telemetry Transport (MQTT) Protocol. This authentication algorithm provides a Perfect Forward Secrecy (PFS) feature that makes an improvement in security. It is the first time that this TLS authentication algorithm is implemented and evaluated over the MQTT protocol for IoT devices. To evaluate resource consumption of the proposed security mechanism, it was compared with the default security mechanism of the MQTT protocol and the ECDHE-ECDSA that is a certificate-based authentication algorithm. They were evaluated in terms of CPU utilization, execution time, bandwidth, and power consumption. The results show that the proposed security mechanism outperforms the ECDHE-ECDSA in all tests.**

Keywords*—Internet of Things (IoT); MQTT; Pre-Shared Keys (PSK); elliptic curve cryptography; Diffie-Hellman Ephemeral (DHE); Digital Signature Algorithm (DSA); Perfect Forward Secrecy (PFS); authentication; power consumption; wireless sensors

I. Introduction

The Internet of Things (IoT) has been developed significantly and is moving towards maturity. It can be viewed as "a global network which provides the communication between human-to-human, human-to-things, and things-to-things by making a unique identity for each object" [1]. Multiple objects which include embedded sensors, wireless communication, processors, can be linked together to make the network of IoT. The Internet of Things is a mixture of two terms. The first term is the Internet, which connects billions of users, devices, personal systems, and even business organizations. The second term is Thing, which refers to intelligent objects [2].

There are many different environments [3] where IoT objects interact automatically with their surroundings and the Internet automatically and independently, as a result, a lot of security and privacy concerns have arisen. IoT devices have become crucial to many enterprises and even cities and preserving exchanged data becomes most important. Also, objects in IoT networks are seriously resource-constrained with limited computational ability, memory, and power, so it is very difficult to implement heavy operations on them which are required by ciphering algorithms. They need a lightweight security mechanism with low resource consumption. Traditional authentication and encryption methods have a huge overhead on IoT devices. Therefore, this study focuses on the evaluation of a lightweight security mechanism that secures communication in IoT networks as well as reduces resource consumption of IoT limited devices.

Based on these requirements in IoT networks, a communication stack is needed to provide a low-power, secure, and lightweight protocol. IoT communication stack includes different types of protocols [4] [5], this paper was tried to improve the security of the MQTT application layer protocol since it built on top of TCP protocol, has low power usage and lightweight overhead than other IoT protocols [6]. In addition, because IoT networks depend on TPC/IP, in this study, Transport Layer Security (TLS) [7] is selected which is the reliable protocol and supports most of the security cipher suites. Then the TLS evaluated over MQTT protocol when using the ECDHE-PSK authentication algorithm. The results are then compared with the ECDHE- ECDSA and also with the default security mechanism of the MQTT protocol.

The rest of the paper is categorized as follows: Section II reviews previous works related to IoT security also an overview of the MQTT protocol. Section III provides details of the proposed security mechanism. Section IV describes performance evaluation and discussion on the results. Lastly, Section V concludes the paper and offers a suggestion for future work.

II. Related Work

A. Message Queuing Telemetry Transport (MQTT) Protocol

MQTT is an open protocol, standardized by OASIS and became an ISO standard (ISO/IEC 20922:2016) [8]. This protocol is being adopted widely and used extensively by most big companies such as Amazon and Facebook to exchange data between resource-constrained devices. MQTT supports publish/subscribe architecture [9] over TCP [10] protocol. It has two components that are the client as a publisher or subscriber and the broker. The publisher publishes messages and the client subscribes to certain topics that are relevant to them and by that, receives every message published under those topics. Each message has a topic and clients can

subscribe to several topics. These topics categorized in a hierarchical system [11] similar to file paths in a computer; e.g. "home/living room/air condition/status".

MQTT has a low overhead, meaning that it sends a very small amount of extra data and the actual content of the message. The MQTT header is incredibly small (only 2 bytes) [12] in comparison with other protocols like HTTP or CoAP [13].

Among different IoT protocols the Quality of Service (QoS) [14] features of MQTT make it unique which guaranteed delivery of messages and assurance of data distribution between two parties.

MQTT has default security features [15], For instance, for authenticating purposes, MQTT offers a simple authentication method via username and password for connecting a client to the broker. Authentication options are sent in plaintext without any encryption. Therefore, developers can implement their own security mechanisms on the network and transport layer. Security can also be provided at lower layers. For instance, MQTT uses TLS on the Transport layer to protect communication between parties.

B. The Transport Layer Security (TLS) and its cipher suits

TLS protocol [16] is a widely used secure-channel protocol that allows secure end-to-end communication between two devices. It utilizes cryptography for data protection and device authenticity. This protocol contains two main protocols: the TLS handshake protocol and the TLS record protocol. The handshake protocol allows the client and server to authenticate each other and to negotiate an encryption algorithm, MAC algorithms, and session keys for data encryption in the TLS record [17]. The TLS Record Protocol secures the connection after the TLS Handshake Protocol established using symmetric cryptography such as RC4 or AES [18] and hash functions like SHA-1 [19]. The operation is determined by the cipher suite which applied and its name represents the involved algorithms. For example, the TLS-ECDHE-ECDSA-CHACHA20-POLY1305-SHA256 cipher suite uses ECDHE as the key-exchange algorithm, ECDSA as the authentication algorithm, CHACHA20-POLY1305 as the stream cipher, and SHA256 as the hash algorithm to preserve message integrity of the handshake process.

There are some alternatives for securing end-to-end communications in networks with resource-constrained devices. For instance, using a protocol like Datagram TLS (DTLS) [20] was suggested in the past. It was developed to secure UDP-based protocols such as CoAP. It was originally developed to protect web application communication and its executions have heavy overhead [21] in IoT networks. Further, DTLS implementation is complex since it works on top of the UDP protocol. Since DTLS using UDP as a transport protocol, it cannot ensure reliability as much as TLS can, although it employs a "sequence number" field for verification. Furthermore, DTLS is not resistant against Denial of Service (DoS) attacks [22]. As a result, the use of (TLS) is becoming compulsory for most of the communications since it provides a better security level.

C. Elliptic Curve Diffie Hellman Ephemeral (ECDHE) Key exchange algorithm

ECDHE is an algorithm used for key exchange which lets two entities to make a shared secret. In fact, it is an adaption of the Diffie-Hellman (DH) [23] key exchange protocol that employs elliptic curve cryptography to minimize the key length and improve performance. Elliptic curves are used widely in various key exchange methods which include the DH key agreement. ECC [24] provides a security level similar to RSA but with smaller key sizes [25] that result in fast calculations and less power consumption for IoT devices.

In the ECDHE algorithm, each party must generate a key pair (include a public key and a private key). Public keys are Ephemeral therefore a unique session key made for each session and provide the Perfect Forward Secrecy (PFS) [26] feature. This feature guarantees that the session keys will not be compromised by making a unique session key for each session. Previous key exchange algorithms like RSA [27] and ECDH [28] cannot provide the (PFS) feature because of their static public keys. There are some papers that compared RSA and ECDH-based cryptography together. According to measurement results in [29], the overall execution time of TLS-ECDH is faster than TLS-RSA. In [30] the power consumption and performance of RSA, DH, and ECDH key exchange algorithms are compared and the results showed the ECDH algorithm is better than others. Although ECDH outperforms RSA and DH algorithms, it cannot provide forward secrecy feature that is possible by using ECDHE algorithm.

D. Elliptic Curve Digital Signature Algorithm (ECDSA) Authentication algorithm

The U.S. National Institute of Standards and Technology (NIST) developed the Digital Signature Algorithm (DSA) [31] for use in their Digital Signature Standard (DSS) in 1991. In a Digital Signature, the signer's Private Key is used to sign, and the signatory's Public Key is used to verify the signature by the recipient. The ECDSA [32] is an elliptic curve variant of the DSA algorithm that produces cryptographically digital signatures by using the Elliptic curve. This algorithm applies (160/256 bits) which is much smaller rather than (1024/2048 bits) in DSA and RSA [33]. In [34] they considered ECDSA performance compared with RSA and concluded how employing various ECC curves and also RSA key sizes impact energy consumption in IoT nodes. The results showed that ECDSA was better than RSA to secure IoT communications with resource-constrained devices since RSA has large key sizes. Although ECDSA eliminates the issues of RSA and DSA algorithms, its resource consumption is still high for IoT resource-constrained devices due to signing and verifying the certificate.

III. Proposed Security Mechanism and Experiments

Security in IoT networks is the main concern for developers. Since most IoT nodes are limited in resource and power, they require a security mechanism that fits their limitations. This research provides a lightweight security mechanism for IoT resource-constrained devices over the MQTT protocol.

Previous authentication methods (like RSA and ECDSA) use a certificate to authenticate devices. Although using a certificate provides a high level of security, signing and verifying certificate uses a high computational process that increases CPU and power consumption. Therefore, instead of using certificate-based authentication algorithms, the proposed mechanism uses the PSK authentication algorithm that uses a pre-shared key to authenticate other parties along with the ECDHE key exchange algorithm. The symbols used to describe different processes in the proposed mechanism are illustrated in Table I.

The ECDHE is used for the key exchange process in the proposed security mechanism. In this process, to use ECC, each party must agree on the domain parameters (p, a, b, G, n, h) that define the elliptic curve. Also, the client and server should have a key pair for elliptic curve cryptography, including a private key d (a random integer) and a public key Q ($Q = dG$). Therefore, the private key d_C and the public key $Q_C = d_C G$ are for client and the keys d_S and $Q_S = d_S G$ are for server. Then, the client and the server exchange their public keys. They calculate the secret key with using their own private key and other party's public key. The client computes $S = d_C Q_S$ and the server computes $S = d_S Q_S$. The shared secret key (S) is equal for both parties since $S = d_C Q_S = d_C d_S G = d_S d_C G = d_S Q_C$ [29].

ECDHE algorithm does not provide authentication on its own, because the key is different every time and any of the parties cannot be confident that the key is from the intended party. Therefore, the Pre Shared Key (PSK) [35] authentication algorithm is used along with ECDHE in order to authenticate both parties.

The PSK authentication algorithm applies a string of characters (64 hexadecimal digits) which is used as an authentication key (shared secret) and shared previously between the client and the server to text encryption. When the secret key is shared between them, they authenticate each other through the four-step procedure Shared Key authentication algorithm [36]. The benefit of the PSK algorithm is to avoid heavy public key calculation to authenticate. The only problem of it is that if the attacker can obtain the shared secret key, previous and future sessions would be compromised. When the proposed mechanism uses PSK along with ECDHE key exchange algorithm, it provides the Perfect Forward Secrecy (PFS) feature that protects past sessions against future compromises by providing a distinct key for each session. Even if the attacker accesses this shared secret somehow, it would only compromise that specific session. Previous or future sessions would not be compromised.

A. Test Environment Architecture

To set up the architecture of the test environment, different suitable hardware was selected.

As can be seen in Fig. 1, there are two Raspberry Pi devices which act as MQTT publisher and subscriber and connected to the router using Wi-Fi connection. The ODROID-C1 acts as the MQTT broker connected through an Ethernet connection to the router. In addition, they connect to the power supply with a power cable. The power measurement hardware device placed between the publisher and the power supply to measure the energy consumption. The parameters that used in the test environment to evaluate the proposed security mechanism are shown in Table II.

Table III illustrates the different layer protocols used in the test environment.

TABLE I. SYMBOLS USED IN THE PROPOSED MECHANISM

symbols	Definition
p	The Prime Case
a, b	Elliptic curve constants
n	The smallest positive number
h	An Integer number called The cofactor
d	Private Key
Q	Public Key
G	Generating Point
S	Secret Key

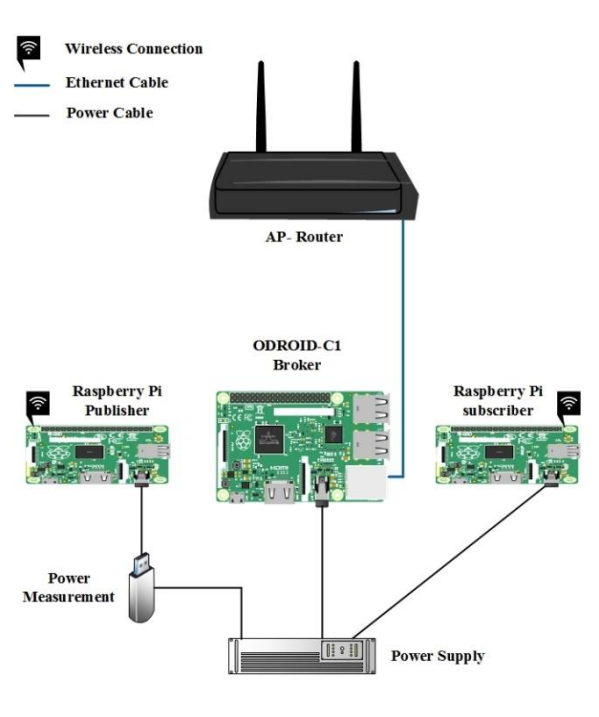

Fig. 1. General Test Environment Architecture.

TABLE II. TEST ENVIRONMENT PARAMETERS

Parameter	**value**
MQTT Broker	1
MQTT Publisher	1
MQTT Subscriber	1
Number of published messages	30
Message Length	152 byte
Power Measurement Device	1
Access Point-Router	1
Test frequency	60 times

TABLE III. TEST ENVIRONMENT LAYER AND PROTOCOL

Layer	Working Protocol
Radio	IEEE802.11
Link	Ethernet
Network	IPv4/IP6
Transport	TCP
Encryption	TLS
Application	MQTT

B. Test Environment Equipment

The Raspberry Pi Zero W embeds Wi-Fi and Bluetooth v4.1. Its processor is a 1GHz BCM2835 SOC (32-bit ARM-based processor) that operates with 512MB RAM. The ODROID-C1 Single Board Computer (SBC) equipped with an Ethernet interface and two High-Speed USB A-type connectors. It has 1Gbyte DDR3 RAM and 1.5GHz Quad-Core ARMv7 processor. The power measurement device can measure voltages up to 30V and currents up to 5.1A. The AP-Router has Wi-Fi and LAN ports used to connect the Raspberry Pi devices and The ODROID-C1 together.

Besides hardware equipment, some analyzer tools and software used to measure the evaluation metrics. The PERF performance analyzer tool used to monitor CPU clock and execution time. The NMON analyzing tool was used to monitor the CPU utilization percentage. The Wireshark software used to monitor packets transferred between the clients and the broker to calculate bandwidth consumption. The OpenSSL used for Raspberry Pi devices that act as clients and also for the broker, and the C scripts run on the clients. In addition, The Eclipse Paho MQTT installed on the Raspberry Pi devices that act as MQTT clients, and the Eclipse Mosquitto installed on the ODROID-C1 acts as an MQTT broker. Two cipher suites were chosen according to the Internet Engineering Task Force (IETF): TLS-ECDHE-ECDSA-CHACHA20POLY1305-SHA256 and TLS-ECDHE-PSK-CHACHA20POLY1305-SHA256.

The Raspberry Pi device (publisher) starts the tests. A script on it specifies which cipher suite to be used, and then it sends messages to the Broker several times. The resource consumption by each of the MQTT transactions was measured and registered. After all the tests were done, tables and graphs related to the test data were generated.

IV. PERFOMANCE EVALUATION AND DISCUSSION

The proposed security mechanism evaluation was done using a real test environment. In order to evaluate the developed mechanism, three types of scenarios were conducted and the results of them investigated to conclude which security mechanism is more appropriate for the IoT devices. The first scenario is the default security mechanism of MQTT, the second scenario is the ECDHE-ECDSA security mechanism, and the last scenario is the proposed security mechanism which is the ECDHE-PSK. In each scenario, 30 messages with 152-byte length were sent and the test repeated 60 times and the average values were calculated.

The work tested based on four evaluation metrics: including CPU utilization, execution time, bandwidth usage, and power consumption. All these performance metrics tested on three scenarios.

A. Average CPU utilization

Fig. 2 compares the average CPU utilization percentage between three security mechanisms. As can be seen, ECDHE-ECDSA consumed the most CPU percentage, about 60%, since the signing and verification process needs more computation. In comparison, the ECDHE-PSK security mechanisms used 10% less than the ECDHE-ECDSA because it does not use any certificate for authentication. The default security mechanism of MQTT, which used only a simple user name and password for authentication, utilized the lowest CPU.

B. Average Execution Time

According to the bar chart shown in Fig. 3, the highest execution time related to the ECDHE-ECDSA is 2.91 seconds. While the average time fell to 2.58 seconds when the ECDHE-PSK security mechanism executed on the device.

C. Average Bandwidth Consumption

To measure the consumed bandwidth for each security mechanism the following bandwidth equation was used:

$$\text{Consumed Bandwidth (bps)} = \frac{\text{The total size of packets}}{\text{Total Simulation Time}}$$

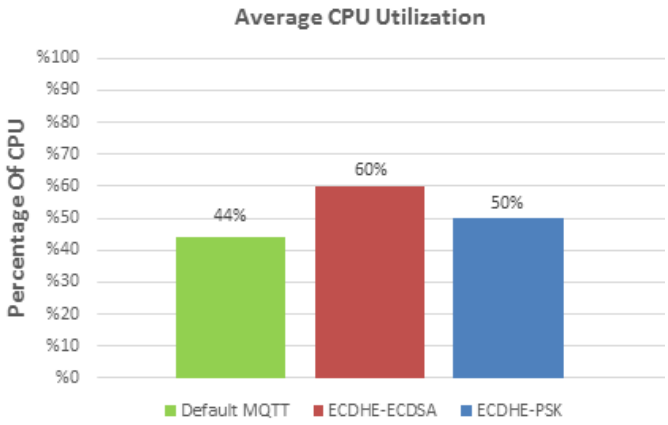

Fig. 2. Average CPU Utilization Percentage of Default MQTT, MQTT with ECDHE-ECDSA and ECDHE-PSK.

Fig. 3. Average Execution Time of Default MQTT, MQTT with ECDHE-ECDSA and ECDHE-PSK.

As shown in Fig. 4, in the default security mechanism of MQTT, the average bandwidth consumption was only 80,391 bps. In the implementation of MQTT with ECDHE-ECDSA security mechanism, the average bandwidth consumption has been increased significantly and reached 154,269 bps because of the certificate which is transferred between publisher and broker in this security mechanism. While the proposed mechanism used only 38,084 bps more than the default security mechanism of the MQTT, its average bandwidth consumption is around 118,475 bps. As can be seen, the authentication mechanism has an effect on increasing bandwidth consumption where extra packets are added to negotiate and they include extra bytes for handling authentication parameters.

D. Average Power Consumption

Power consumption is the other metric. When the traffic volume increases, it will result in higher processing. Therefore, the power consumption increases too. As shown in Fig. 5, the ECDHE-ECDSA security mechanism used by far the highest power due to the signing and verifying certificate.

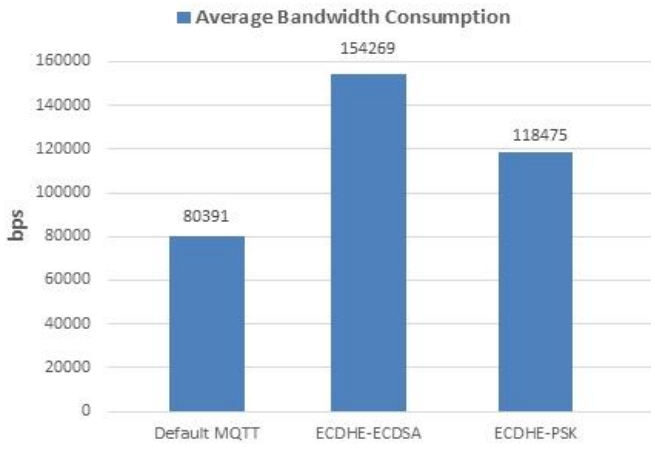

Fig. 4. Average Bandwidth Consumption of Default MQTT, MQTT with ECDHE-ECDSA and ECDHE-PSK.

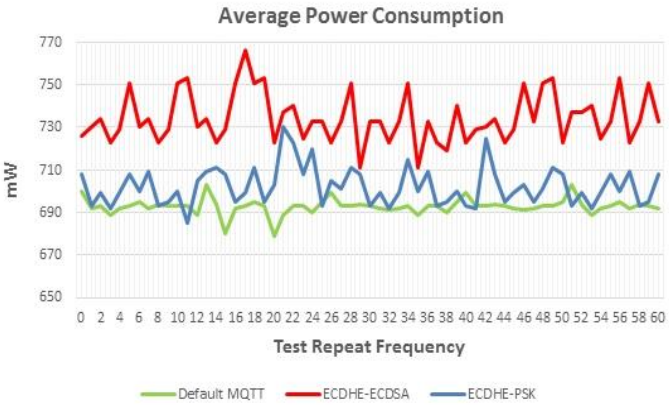

Fig. 5. Average Power Consumption in Default MQTT, MQTT with ECDHE-ECDSA and ECDHE-PSK.

V. Conclusion and Future Work

The aim of this paper was to propose a lightweight security mechanism for the IoT resource-constrained devices over the MQTT protocol. Since authentication is an unavoidable step to secure communication, the impact of the different authentication algorithms on IoT nodes was evaluated. Most secure authentication algorithms for securing the TLS communications like RSA and ECDSA use a certificate to authenticate other parties that is heavy for IoT devices. The use of a Pre-shared key instead of a certificate can be useful to authenticate IoT devices. Therefore the performance and resource consumption of the ECDHE-PSK authentication algorithm with the ECDHE-ECDSA certificate-based authentication algorithm was evaluated and compared. Since the PSK algorithm does not require a certificate to authenticate; it can decrease the resource consumption of IoT devices significantly. In addition, the ECDHE-PSK provides the Perfect Forward Secrecy (PFS) feature to ensure more secure communication.

After running different evaluations, the proposed security mechanism ECDHE-PSK outperformed ECDHE_ECDSA in all evaluation metrics. It utilized less CPU, execution time, bandwidth, and energy than the ECHDE-ECDSA security mechanism which is one of the most popular and reliable TLS authentication algorithms recently used. It is clear that using a certificate to validate the data will result in more data transfer as well as more CPU usage. The increase in these two factors has an effect on the amount of power consumption too. IoT devices are usually vulnerable because their hardware cannot accommodate the required certificates while reducing power consumption and CPU processing. By using ECDHE-PSK the IoT devices can overcome these limitations.

Since the IoT field is still a new research topic, the existing IoT simulators do not support the latest cipher suites and security mechanisms on the MQTT protocol. Therefore, to evaluate the proposed mechanism, a real testbed was required. It was the first time that this TLS authentication algorithm was implemented and evaluated over the MQTT protocol. Real environment scenarios testing are the only way to determine which security algorithms are more suitable for IoT devices.

An interesting issue for future work is implementing this security mechanism on a bigger scale with more IoT devices and sensors to evaluate different metrics where the network is under high traffic load. Besides this, some attack scenarios can be simulated to evaluate attack resistance of the security mechanism against common attacks.

Acknowledgment

This research is carried out under the Computer Security and Software Verification Research Lab, Center for Cyber Security (CYBER), www.ftsm.ukm.my/cybersecurity, Faculty of Information Science and Technology.

This research is funded by Universiti Kebangsaan Malaysia (UKM) research project grant GGPM-2017-021.

References

[1] S. Madakam, R. Ramaswamy, S. Tripathi, “Internet of Things (IoT): A literature review, ” Journal of Computer and Communications 3 (05), 164, 2015.

[2] K.K. Goyal, A. Garg, A. Rastogi, S. Singhal, "A Literature Survey on Internet of Things (IoT), " Int. J. Advanced Networking and Applications. Volume: 09 Issue: 06 Pages: 3663-3668, 2018.

[3] A.Oracevic, S. Dilek, & S. Ozdemir, "Security in internet of things: A survey, " International Symposium on Networks, Computers and Communications (ISNCC), 2017.

[4] N. Naik , " Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP, " IEEE International Systems Engineering Symposium (ISSE), 2017.

[5] H. Sardeshmukh, and D. Ambawade, "Internet of Things: Existing protocols and technological challenges in security," International Conference on Intelligent Computing and Control (I2C2), 2017.

[6] D. Soni, A. Makwana, "A Suever on Mqtt: A Protocol of Internet Of Things (IOT)", April 2017.

[7] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) protocol version 1.2, " RFC 5246, Internet Engineering Task Force (IETF), August 2008. [Online] Available: Https://Tools.Ietf.Org/Html/Rfc5246.

[8] A. Cornel - Cristian, T. Gabriel, M. Arhip-Calin, & A. Zamfirescu, "Smart home automation with MQTT," 54th International Universities Power Engineering Conference (UPEC), 2019.

[9] M.A. Triawan, H. Hindersah, D. Yolanda and F. Hadiatna, "Internet of things using publish and subscribe method cloud-based application to NFT-based hydroponic system," 6th International Conference on System Engineering and Technology (ICSET, 2016.

[10] K. D. Mayoof, R. Hassan, A. S. Ahmed & A. Marwan, "Performance Evaluation of Handover in WiMax with TCP and UDP as Underlying Protocol, " Journal of Computer Sciences, 2015.

[11] S. Kraijak, and P. Tuwanut, "A survey on internet of things architecture, protocols, possible applications, security, privacy, real-world implementation and future trends, " IEEE 16th International Conference on Communication Technology (ICCT), 2015.

[12] M. B. Yassein, M. Q. Shatnawi, S. Aljwarneh, & R. Al-Hatmi, "Internet of Things: Survey and open issues of MQTT protocol," International Conference on Engineering & MIS (ICEMIS), 2017.

[13] MA. Azzawi, R. Hassan, KAA. Bakar, "A Review on Internet of Things (IoT) in Healthcare," International Journal of Applied Engineering Research, November 2016.

[14] AS. Sadeq, R Hassan, A. Mahdi, "Enhanced MQTT for Providing QoS in Internet of Things (IoT)," American Scientific Publishers, 2018.

[15] V. Lampkin, WT. Leong, L. Olivera, S. Rawat, N. Subrahmanyam, R. Xiang, "Building smarter planet solutions with mqtt and ibm websphere mq telemetry" IBM Redbooks Publication, First Edition September 2012.

[16] R. T. Tiburski, L. A. Amaral, E. de Matos, D. F. G. de Azevedo & F. Hessel, " Evaluating the use of TLS and DTLS protocols in IoT middleware systems applied to E-health, " 14th IEEE Annual Consumer Communications & Networking Conference (CCNC), 2017.

[17] A. K. Ranjan, V. Kumar, & M. Hussain, "Security analysis of TLS authentication," International Conference on Contemporary Computing and Informatics (IC3I), 2014.

[18] S. Kamil, M. Ayob, SNHS Abdullah, Z. Ahmad, "Challenges in multi-layer data security for videosteganography revisited, " Asia-Pacific Journal of Information Technology and Multimedia. Vol. 7 No. 2-2, December 2018: 53 – 62, 2018.

[19] A. A. Alkandari, I. F. Al-Shaikhli, & M. A. Alahmad, "Cryptographic Hash Function: A High Level View," International Conference on Informatics and Creative Multimedia, 2013.

[20] E. Rescorla and N. Modadugu, "Datagram Transport Layer Security Version 1.2," RFC 4347, Internet Engineering Task Force (IETF), January 2012. [Online] Available: https://tools.ietf.org/html/rfc6347.

[21] A. Capossele, V. Cervo, G. De Cicco, & C. Petrioli, "Security as a CoAP resource: An optimized DTLS implementation for the IoT," IEEE International Conference on Communications (ICC), 2015.

[22] A. Haroon, S. Akram, M. A. Shah, & A. Wahid, "E-Lithe: A Lightweight Secure DTLS for IoT," IEEE 86th Vehicular Technology Conference (VTC-Fall), 2017.

[23] P. Deshpande, S. Santhanalakshmi, P. Lakshmi, & A. Vishwa, "Experimental study of Diffie Hellman key exchange algorithm on embedded devices," International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS), 2017.

[24] MA. Azzawi, R. Hassan, KAA. Bakar, "Reliable and Secure Traffic Exchange Approach for Internet of Things (IoT) Devices". 3rd International Congress of Technology, Management and Social Sciences-17 (ICTMS-17), 2017.

[25] M. Bafandehkar, S. M. Yasin, R. Mahmod, & Z. M. Hanapi, "Comparison of ECC and RSA Algorithm in Resource Constrained Devices," International Conference on IT Convergence and Security (ICITCS), 2013.

[26] Z. Yang, J. He, Y. Tian, & J. Zhou, "Faster Authenticated Key Agreement with Perfect Forward Secrecy for Industrial Internet-of-Things," IEEE Transactions on Industrial Informatics, 2020.

[27] R. K. Kodali, H. S. Budwal, K. Patel, & N. Sarma, "Fuzzy controlled scalar multiplication for ECC," IEEE Tencon , Spring 2013.

[28] R. K. Kodali, and A. Naikoti, "ECDH based security model for IoT using ESP8266," International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), 2016.

[29] J. Seo, J. Park, Y.J. Kim, D. Hwang, K. Kim, K. Kim, and K. Lee, "An ECDH-based light-weight mutual authentication scheme on local SIP, " Seventh International Conference on Ubiquitous and Future Networks, 2015.

[30] T. K. Goyal, and V. Sahula, "Lightweight security algorithm for low power IoT devices," International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2016.

[31] M. A. Mughal, X. Luo, A. Ullah, S. Ullah, & Z. Mahmood, "A Lightweight Digital Signature Based Security Scheme for Human-Centered Internet of Things," IEEE Access, 6, 31630–31643, 2018.

[32] W. Wang, Y. Cui, & T. Chen, "Design and implementation of an ECDSA-based identity authentication protocol on WSN," 3rd IEEE International Symposium on Microwave, Antenna, Propagation and EMC Technologies for Wireless Communications, 2009.

[33] S. Lamba, and M. Sharma, "An Efficient Elliptic Curve Digital Signature Algorithm (ECDSA)," International Conference on Machine Intelligence and Research Advancement, 2013.

[34] M. Suarez-Albela, T. M. Fernandez-Carames, P. Fraga-Lamas, & L. Castedo, "A Practical Performance Comparison of ECC and RSA for Resource-Constrained IoT Devices," Global Internet of Things Summit (GIoTS), 2018.

[35] P. Eronen and H. Tschofenig, "Pre-shared key ciphersuites for transport layer security (TLS)," RFC 4279, Internet Engineering Task Force (IETF), Dec. 2005. [Online] Available: http://www.ietf.org/rfc/rfc4279.txt.

[36] L. Barken, E. Bermel, J. Eder, M. Fanady, M. Mee, M. Palumbo, & A. Keobrick, " Wireless Hacking," Chapter1- A Brief Overview of the Wireless World. Publisher: Syngress; 1 edition November 25, 2004.