International Journal of Engineering & Technology, 7 (4.11) (2018) 223-226

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Lightweight Security for MQTT-SN

M. H. Amaran*, M. S. Rohmad, L. H. Adnan, N. N. Mohamed, H. Hashim

Faculty of Electrical Engineering, Universiti Teknologi MARA, 40450 Shah Alam, Selangor, Malaysia
**Corresponding author E-mail: muhammadharith@salam.uitm.edu.my*

Abstract

This study discusses a method to secure Message Queuing Telemetry Transport-Sensor network (MQTT-SN). MQTT-SN is a popular data communication protocol used in developing IoT applications and can be secured by augmenting it with a security scheme. In this work, several lightweight encryption schemes to be used in tandem with MQTT-SN were tested and analysed. The best algorithm is identified based on the speed of encryption and overall power consumption when implemented in IoT environment. It was found that L-BLOCK is the overall performer in securing MQTT-SN and should be highly considered when developing IoT applications.

Keywords: *MQTT-SN; MQTT; Internet of Things; Lightweight Security; Encryption; LBLOCK; UDP; Block Cipher.*

1. Introduction

Internet of Things (IoT) is the concept wherein physical things are interconnected to form an intelligent environment. Advanced manufacturing techniques have allowed for smaller and more efficient electronic devices hence making Internet of Everything widely available. Pervasive connected things or internet of things have become a common occurrence now and can even be found in the consumer market. These consumer products focus on connecting home devices to the internet to provide control of said devices or additional data for further processing. The interaction and processing of data collected from devices form a seamless environment of smart ecosystem.

IoT devices on the market focus more on function and usability compared to security. Most IoT devices either have no or very minimal security causing serious vulnerability among IoT devices. One example of this vulnerability being exploited is the massive Distributed Denial of Service (DDOS) attack in October 2016 [1] targeting Dyn, a Domain Name Server (DNS) service provider. That attack caused congestion and disabled internet connection for most of the east coast area of USA. This shows that vulnerable IoT devices post serious threat to the internet.

Most manufacturers did not integrate security into their device because of high energy and computational cost. High energy and computational cost comes from ancient security and communication protocols which were originally designed for computers and regular devices, not IoT devices. These drawbacks caused the manufacturers to abandon the security protocol altogether since the IoT device are not able to support the security protocol, due to it being restricted by battery capacity and/or computation power. Consequently, it is found that most IoT devices are being produced with minimal or no security functionalities present.

One of the costs to implementing security lies in securing the communication stage. A secure communication needs a long header for the actual encrypted communication packet and multiple transactions for key exchange. Key exchange is usually performed using asymmetric cryptography between two parties. Both parties then use the identical keys and symmetric cryptography encryption to encrypt data packets sent between them.

Transport Layer Security (TLS) is currently the standard protocol to securing data for communication on top of Transmission Control Protocol (TCP). Another variant of the protocol, Datagram Transport Layer Security (DTLS) is the security protocol for securing User Datagram Protocol (UDP). UDP is preferred for IoT devices because their smaller packet header (8 bytes) and connectionless nature. However, adding DTLS to UDP adds at least 33 bytes to the original packet header. The DTLS header is shown in Table 1.

Table 1: DTLS Record Layer [2]

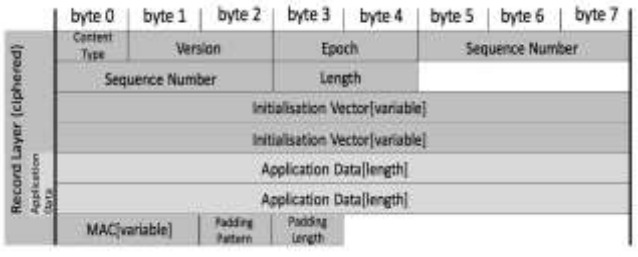

Table 1 shows an example of a DTLS Record Layer. The numbers 0 to represent a group of 8 bits (bytes). The stated Record Layer requires 33 bytes. There are other layers such as Handshake, ClientHello and ServerHello which adds more bytes on top of Record Layer depending on the communication process.

In addition to long packet header, another cost of implementing security comes from the usage of large symmetric block cipher algorithms. A block cipher will produce an encrypted ciphertext similar in size to the block even if the plaintext is shorter. For example, the industry standard Advanced Encryption Scheme 128 (AES128) with the size of 128-bit will produce a ciphertext with a minimum size of 128 bits. Even if the original message is 8 bit, the encryption process would still produce a ciphertext of 128-bit minimum size. As a result, the encrypted packets are bigger in size and this drives the transmission energy cost higher. An obvious solution to the problem would be to use a shorter secure symmetric block cipher suited for IoT devices, which is partly what is being proposed in this paper.

Copyright © 2018 Authors. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

2. Related Works

Message Query Telemetry Transport (MQTT) is an application communication protocol designed for IoT device. MQTT requires only 2 bytes for its packet header. MQTT normally runs on TCP. It can also be used with UDP through a variant called Message Query Telemetry Transport for Sensor Networks (MQTT-SN). This feature has made MQTT to be very flexible to be used in a mixed network of both TCP and UDP.

A hybrid network gives better performance overall for local and internet connection since our earlier study [3] showed that UDP perform better in local small connection and TCP perform better in large multiple nodes network such as the internet. Since most IoT devices usually need a gateway to connect to the internet, a hybrid network is very practical in most setups. The network can be easily deployed with the help of MQTT and MQTT-SN. A hybrid network suggested by MQTT-SN standard [4] is shown as in Figure 1.

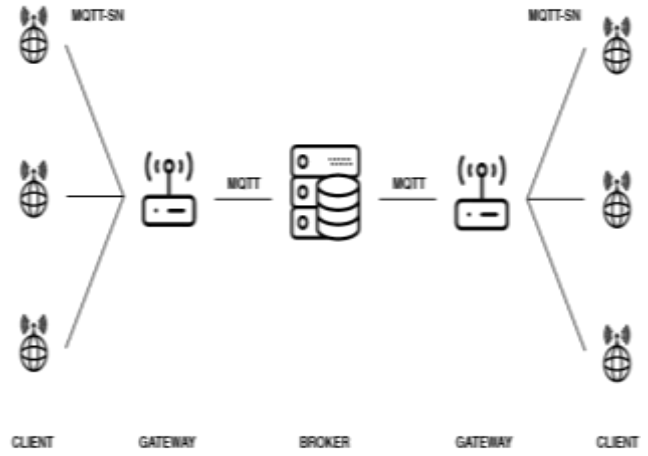

Fig. 1: Hybrid MQTT Network

In the hybrid network shown in Figure 1, IoT clients communicate using MQTT-SN (over UDP) with the gateway. The gateway then forwards the message to the broker using MQTT over TCP. The process is then reversed until the message is received by the receiver on the other side. This method is seamless compared to using other communication protocols for each step since MQTT and MQTT-SN is very similar, apart from MQTT-SN not requiring a TCP/IP stack.

This study is focused on the security protocol of IoT devices communicating with the gateway where in this situation these IoT devices need to communicate with the gateway using MQTT-SN only. A modification to MQTT-SN with addition of security elements adopted from DTLS is proposed to replace the DTLS protocol to enable for shorter lightweight packet headers. The modified header is shown in Table 2.

Table 2: Modified MQTT-SN header

byte 0	byte 1	byte 2	byte 3	byte 4	byte 5	byte 6	byte 7
MsgLength	MsgType	Flags	TopicID		Msg ID		
Initialization Vector (variable)							
Application Data [length]							

A shorter overall packet size is expected to reduce the cost incurred by transmission overhead. In this case, the MQTT-SN packet is modified by making available additional space to accommodate a 64-bit Initialization Vector (IV). The actual payload of MQTT-SN will subsequently be encrypted using lightweight block ciphers, which employs the Initialization Vector to start the encryption process.

3. Lightweight Block Ciphers

Short block ciphers are better suited for IoT applications since IoT devices usually send short periodical message. To minimize the cost of communication, shorter block ciphers with size of 64-bit were analysed. Before AES, Data Encryption Standard (DES) which was a widely-used encryption cipher with 64-bit block size is the standard. However it was proven to be insecure and since, the 64-bit block cipher has been replaced by the more secure 128-bit AES. After DES, new secure 64-bit block ciphers have been developed to fill in the vacuum for small block ciphers at the same size providing security for lightweight devices. We analysed three block ciphers to be used in IoT devices for their lightweight properties. Many new lightweight ciphers have 64-bit block as the smallest block size. This size seems to provide the trade-off between the security of larger block sizes and the performance limitation of IoT devices.

3.1. LBLOCK

LBLOCK [5] is a 64-bit block cipher with 80-bit key using a variation of Feistel Network architecture for encryption. LBLOCK encryption consists of 32 rounds which includes Round, Confusion, and Diffusion functions. LBLOCK can be implemented on software utilizing 3955 clock cycles on 8-bit microcontroller or 1320 gate equivalent (GE) for hardware implementation. LBLOCK claims of a balanced trade-off between security and performance and targeted for RFID and similar resource constrained devices.

3.2. PRESENT

PRESENT [6] is a cipher with 64-bit block cipher and either 80-bit or 128-bit keys. PRESENT utilizes Substitution-permutation Network (SPN) as the fundamental architecture of the cipher. The cipher is designed to be simple and is based on AES finalist candidate (Serpent). Encryption in PRESENT is done using 31 rounds cycle. PRESENT claims it is specialized for hardware implementation. This is achieved by using the same s-box multiple times for encryption, decryption and key scheduling[3]. This helped minimized the area needed to implement on hardware to just 1570 GE.

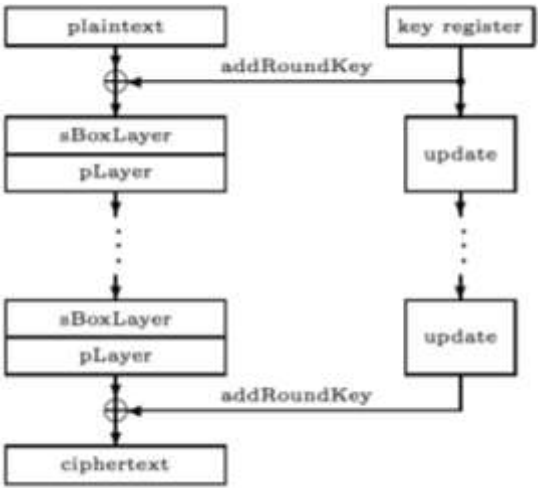

Fig. 2: Present Encryption Routine

3.3. KLEIN

KLEIN [7] is a 64-bit block cipher also intended for RFID and Sensor Nodes. The cipher claims it has advantage when implemented in software compared to hardware. Klein has three variations which each take a different key length 64, 80, and 96 bits respectively. KLEIN-64 (64-bit key) is recommended hash and message authentication while the other two are preferred for en-

crypting the actual plaintext. KLEIN-80 can be implemented on hardware for 2097 GE. Authors of [6] also found that software implementation of KLEIN-64 consumes 4148 bytes of ROM and 97 bytes of RAM.

4. Results and Discussion

All experiments are performed using two Raspberry Pi (RPi) single board computer with 1 GB of RAM and a Wireless Access Point. One RPi serves as the client node and another RPi serves as the gateway. The connection setup for the test bed is shown in Figure 3.

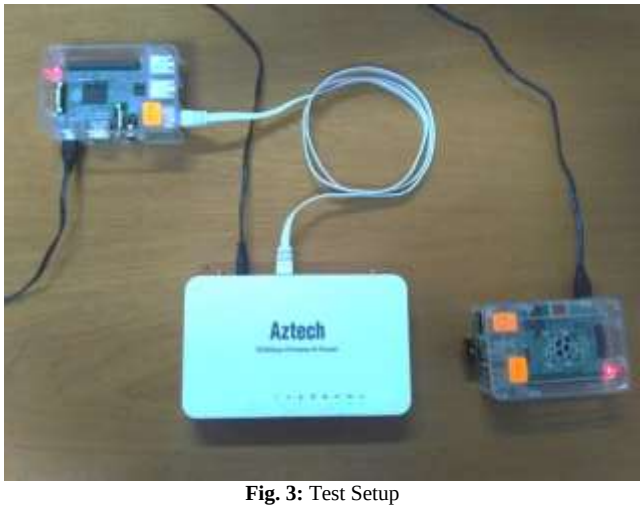

Fig. 3: Test Setup

The Client RPi connects to the WiFi Access Point through WiFi and acts as the IoT device. Gateway RPi which connects to the Access Point via Ethernet serves as the MQTT-SN gateway. The first analysis is to test the performance of encrypted communication against time. This is done by encrypting the message on the Client and sending the packets through the network to the gateway for two seconds. After two seconds, the program will show how many packets have been sent during the period. This process is repeated several times to make sure that the numbers does not vary significantly. The average result is recorded and compared among different ciphers.

Encryption is performed using cipher block chaining (CBC) mode. This mode will chain the blocks of data encrypted using block ciphers resulting in what seems to be random characters called ciphertext. Since block ciphers are used, plaintext must be padded to be equal to the size of the block, which is 64-bit for all the block ciphers mentioned except for AES (128-bit). Plaintext were padded with PKCS7 [8] padding. After padding, initialization vector (IV) were generated using PCG [9] random number generator. The IV will be used for initializing the CBC mode encryption. CBC mode is explained in Figure 4.

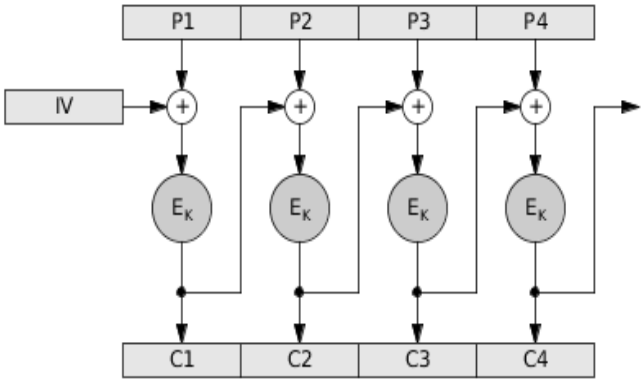

Fig. 4: CBC Mode of Operation

CBC Mode starts with a random number generated as the IV. The size of IV is the same as the block size which is in this case 64-bit. Plaintext which was segmented into blocks of 64-bit then XORed with the IV. The result of the XOR step is the input to the Encryption step. The product of the first Encryption round is denoted as Ciphertext 1 (C1). A copy of C1 will then be XORed with the next Plaintext (P2) and then encrypted resulting in Ciphertext 2. The process will continue up to the last block of Plaintext. The outcome of all Ciphertext (C1, C2… Cn) are subsequently concatenated forming the final Ciphertext.

Two types of payload length are used for this experiment. The first was using minimum possible length of payload and the second is the maximum possible length payload. Maximum payload means a maximum number of bytes in one UDP packet (1280) is used. For minimum payload, a single byte which then padded to the size of the respective block cipher is used.

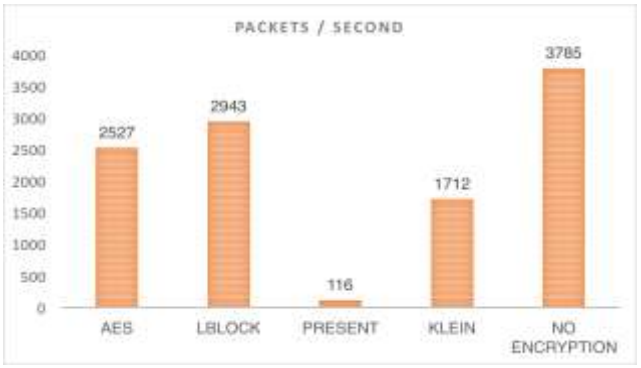

Fig. 5: Average Speed using Maximum Payload

Figure 5 shows that LBLOCK performed best in term of speed when compared to AES and other block ciphers in maximum payload test, managing to send 2943 packets per second which is 16.5% increase over AES.

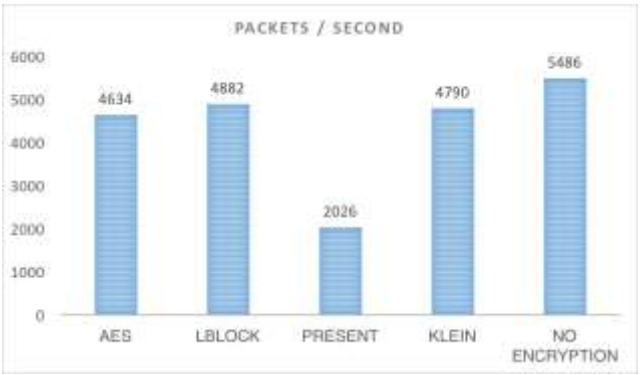

Fig. 6: Average Speed for Minimum Payload

Figure 6 shows that in minimum payload test, LBLOCK, AES and KLEIN present almost similar performance. However, LBLOCK is still the fastest with 4882 packets per second losing only to no encryption.

The previous two figures show the different performances between the encryption algorithms in minimum versus maximum payload setting. This shows that in minimum payload setting, the processing time for encryption is not significant compared to the transmission time. The fact is made clear in Table 3.

Table 3: Performance Difference Payload

Payload	LBLOCK	No Encryption	Difference
Maximum	2943	3785	28.6%
Minimum	4882	5486	12.4%

Table 3 also indicates that encryption process did hamper performance of communication and is more prominent when the payload

is large. This is the case since bigger payload involves more encryption rounds.

A second analysis which measures the power consumption of RPi during encrypted communication was also done. In this analysis, the same procedure is repeated with the addition of an ammeter connected to RPi power supply cable to measure the ampere and calculate the power consumption. The current reading is taken during the communication process. A loopback mode in which the RPi encrypts and sends messages to itself is also performed. Loopback was added to isolate energy consumption between the encryption process and transmission process.

The idle ampere for the RPi is measured at 285mA. Since the power supply for the RPi is rated at 5V, the power consumption can be calculated using the formula P = IV. The measured ampere during communication then subtracted by 285mA to get the actual ampere used by the process hence arriving at the power consumption of the communication process.

Maximum payload is sent for 10 seconds. Total packets sent during 10 seconds is recorded. The current is measured during the process and then the power consumption is calculated using the formula mentioned before. Power consumption is then divided by the total number of packets sent to arrive at average power consumption per single packet as shown in Figure 7-8.

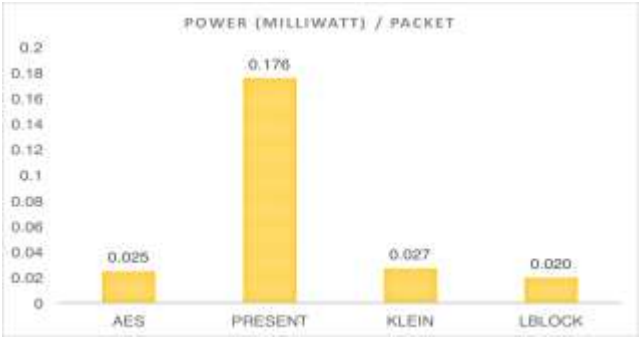

Fig. 7: Power Consumption with Transmission

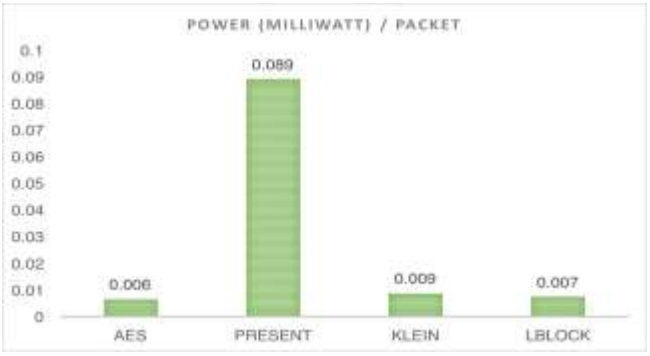

Fig. 8: Power Consumption without Transmission

Figure 7 shows power consumption per encrypted packet sent via WiFi, while Figure 8 shows the power consumption for loopback mode or encryption without transmission. The difference between these two figures can be regarded as the power consumed by WiFi communication.

Both Figures 7-8 show that PRESENT is very poor in power consumption. This is a result from very few number of packets sent in the period of the 10 seconds compared to other ciphers. This may come from the fact that PRESENT is said to be optimised for hardware implementation rather than software as used in the experiment.

Table 4: Comparison of Encryption and Total Power

	AES	PRESENT	KLEIN	LBLOCK
Encryption Power (µW)	6	89	9	7
Total Power (µW)	25	176	27	20
Transmission Power (µW)	19	87	18	13
Encryption/Total Power	24%	50.1%	33.3%	35%

In Table 4, it is observed that power consumption of encryption process is generally lower compared to transmission process except for PRESENT. For AES, KLEIN and LBLOCK the power consumed by encryption process is in the range of one third of the whole process. This shows that cost of WiFi transmission is significantly higher than the cost of encrypting the data using block cipher.

5. Conclusion

IoT devices need efficient algorithm to secure the network from adversaries. Efficient secure communication scheme is needed since most IoT devices run on batteries and use costly wireless communication.

LBLOCK is the promising algorithm to be used in IoT devices in tandem with MQTT-SN. The algorithm performs better than AES in term of encryption speed, and almost the same as AES in power consumption.

Since the power cost of transmission in IoT devices is greater than encryption cost, shorter block cipher is more efficient. In this environment, LBLOCK is preferred compared to AES.

More work must be done in the future to further decrease the number of total bytes sent to increase efficiency in secure IoT communication. One method of doing this is modifying and simplifying DTLS to reduce the number of bytes in packet header.

Acknowledgement

This project was made possible by funds from Malaysian National Research Grant Scheme (NRGS) 600-RMI/NRGS 5/3 (5/2013) from the Ministry of Education, Malaysia.

References

[1] Almeida, V. A., Doneda, D., & de Souza Abreu, J. Cyberwarfare and digital governance. IEEE Internet Computing, 2017, 21(2), 17–20.

[2] Hennebert C., and Dos Santos J. Security protocols and privacy issues into 6LoWPAN stack: A synthesis. IEEE Internet Things J., 2014, 1(5), 384–398.

[3] Amaran M. H, Noh N. A. M, Rohmad M. S, and Hashim H. A comparison of lightweight communication protocols in robotic applications. Procedia Comput. Sci., 2015, 76, 400–405.

[4] International Business Machines (IBM). MQTT for sensor networks (MQTT-SN) protocol specification. IBM Corporation, 2013.

[5] Wu W and Zhang L. LBlock: A lightweight block cipher. Proceedings of the International Conference on Applied Cryptography and Network Security, 2011, pp. 327–344.

[6] Bogdanov A, Knudsen L. R, Leander G, Paar C, Poschmann A, Robshaw M. J, Seurin Y, and Vikkelsoe C. PRESENT: An ultra-lightweight block cipher. Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems, 2007, pp. 450–466.

[7] Gong Z, Nikova S, and Law Y. W. KLEIN: A new family of lightweight block ciphers. Proceedings of the International Workshop on Radio Frequency Identification: Security and Privacy Issues, 2012, pp. 1–18.

[8] Kaliski B, PKCS# 7: Cryptographic message syntax version 1.5. 1998, http://www.rfc-editor.org/rfc/pdfrfc/rfc2315.txt.pdf.

[9] O 'neill M. E. A PCG: A family of simple fast space-efficient statistically good algorithms for random number generation. ACM Transactions on Mathematical Software, 1988, 1–46.