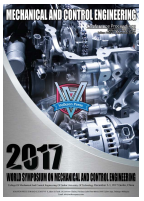
ISBN: 978-1-948012-06-5

Contents List available at VOLKSON PRESS

World Symposium on Mechanical and Control Engineering (WSMCE)



RESEARCH AND DESIGN OF LIGHTWEIGHT ENCRYPTION FOR MQTT PROTOCOL

Yanping Pan, Xiaohui Cheng*, Congcong Xia

College of Information Science and Engineering, Guilin University of Technology Jiangan road No.12, Guilin, China.
Corresponding Author Email: cxiaohui@glut.edu.cn

ARTICLE DETAILS

Article History:

Received 02 october 2017
Accepted 06 october 2017
Available online 11 november 2017

Keywords

MQTT, publish/subscribe, lightweight encryption, TEA algorithm

ABSTRACT

MQTT is a lightweight messaging protocol based on publish/subscribe model, which is a transmission protocol for mobile devices designed specifically for network unreliable and limited resources. The emergence of MQTT protocol increases the timeliness of information acquisition in practical application, and replaces traditional drawing with push method to carry out message transmission. Its emergence provides a new way for the realization of message push on Android platform. Due to the lightweight property of the MQTT protocol, there is no encryption control for message push, instead of sending the original bytecode directly. The methods are no built-in encryption, which has some security risks. In this paper, the lightweight encryption algorithms of ECC, DES and TEA are compared and analyzed, and the lightweight features and encryption requirements takes into account. Finally, the TEA encryption algorithm is selected by the paper. The TEA algorithm's superior encryption performance is verified on the message delivery platform based on the MQTT protocol. The results show that TEA can meet the basic requirements of the lightweight encryption algorithm.

1. INTRODUCTION TO MQTT

1.1 The Background of the MQTT

MQTT (Message Queuing Telemetry Transport) is an protocol, which is open, simple, lightweight and easy to implement, especially suitable for devices and mobile terminals of low bandwidth, network instability, expensive network and limited processor and memory resources. IBM has successfully used the MQTT protocol in many projects, and Facebook also has successfully used the MQTT protocol on mobile platforms [1-3]. In March 2013, the structured information standards organization announced that MQTT would be the preferred standard for emerging Internet of Things messaging protocols. The development of the future will move towards the Internet of Things, and the MQTT protocol will be strongly welcomed and widely used by many well-known companies.

The MQTT protocol is a typical publish/subscribe model based on the principle of publishing messages and subscription topics. The client subscribes to a topic and then receives all the messages that are published to that topic. The client can publish the message to the topic so that the published message can be sent to each subscriber of the topic. MQTT can be divided into two parts: an MQTT client, an MQTT message broker. Using the C/S structure, the MQTT client uses the MQTT protocol to connect to an MQTT message broker directly [4].

1.2 MQTT Protocol Message Format

The MQTT message body consists of three parts: fixed head, variable head, and payload. Only the fixed head is part of all the message body, and the variable head and payload are part of the message. Its structure is shown in table 1.

Table 1: *MQTT Message Fixed Header Format*

Bit	7	6	5	4	3	2	1	0
byte1	Message Type				DUP flag	QoS level		Retain
byte2	Remaining Length							

1.3 Characteristics of the MQTT Protocol

MQTT has the following characteristics:

a) Using the publish/subscribe messaging model, you can make a one-to-many message release, which removes the coupling between the applications.

b) When a message is passed, the content of the load is unknowable.

c) Network connection services are provided using the TCP/IP protocol.

d) Provide three levels of messaging service quality: at most once (Qo S0), at least once (Qo S1), only once (Qo S2)

e) Small transfers, small overhead (a fixed-length header has only 2 bytes), and protocol exchange minimization to reduce network traffic.

f) Provides a mechanism for notifying the client of an abnormal disconnect with last will and the notification feature.

The MQTT protocol is mainly designed for the platform of Internet of Things, in order to maintain the features of streamlining, lightweight and efficient. The MQTT protocol does not use built-in encryption, because the built-in encryption will bring a burden on the transmission connection, therefore, it is necessary to consider both lightweight and encryption. The common

lightweight encryption algorithm has ECC (Elliptic Curves Cryptography),DES(Data Encryption Algorithm) and TEA(Tiny Encryption Algorithm) and so on. In the second chapter, these three encryption algorithms are introduced respectively, and the comparison of algorithm performance is carried out.

2. INTRODUCTION OF LIGHTWEIGHT ENCRYPTION ALGORITHM

2.1 ECC Algorithm

The ECC algorithm is a method of public key cryptography based on elliptic curve mathematics, which is one of the highest encryption intensity for each bit in the known public key system [5,6]. Initially, both Koblitz and Miller proposed that their mathematical basis was to make use of rational points on the elliptic curve to make the calculation of elliptic discrete logarithm in the ellipse additive group difficult in 1985.Encryption and decryption using different keys, and the main process of encryption and decryption as the following six steps.

a) User A enters the plaintext to be sent.

b) Code to the plaintext to get the plaintext point on the elliptic curve.

c) Use the public key to encrypt to get the ciphertext point on the elliptic curve.

d) Transmitted to the user B, and user B is decrypted using the private key after receiving the ciphertext point.

e) Obtain the plaintext point on the elliptic curve after decryption, and decode the plaintext point.

f) User B gets the plaintext after decrypted.

Test the ECC encryption algorithm, the test results shown in Figure1 below.

Figure 1: *Test Effect of ECC Algorithm*

2.2 DES Algorithm

DES algorithm is a symmetric cryptographic encryption algorithm developed by IBM in 1972, and it's also called the U.S [7]. data encryption standard. DES is a packet encryption algorithm, which is typical of DES in 64 bits for data encryption, encryption and decryption using the same algorithm. The plaintext is grouped by 64-bit, the key length is 64-bit. Actually, there are only 56 involved in the operation, and the first 8,16,24,32,40,48,56,64 bit is the parity bit. The encryption method of the ciphertext group is according to a replacement or exchange after the grouping. The algorithm is characterized by shorter grouping, shorter key, shorter password life cycle and slower operation speed.

Test the DES encryption algorithm, the test results shown in Figure 2 below.

Figure 2: *Test Effect of DES Algorithm*

2.3 TEA Algorithm

In 1994, David Wheeler and Roger Needham of the Cambridge University Computer Laboratory invented the TEA encryption algorithm. It is a packet cipher algorithm with a cip-hertext length of 64bit and a key length of 128bit.Firstly, TEA encryption algorithm converts the data content into a byte array, and then uses the shift and easy or operation to get the ciphertext. Different keys to get the ciphertext is different, but the encryption and decryption keys must be consistent. TEA algorithm also uses the ever-changing golden division rate Delta, so that each round of the encryption rules are not the same, TEA algorithm iteration can also be changed, the official recommended number of iterations is 32 rounds. TEA encryption algorithm has the characteristics of fast speed, high efficiency, and the implementation is also very simple and so on [8].

Compare DES (Data Encryption Standard) algorithm, TEA encryption algorithm is relatively simple, but the TEA algorithm encryption faster than DES, which using a 128-bit key to encrypt 64-bit data, the security is better, and it has a strong ability to resist differential analysis. The TEA encryption algorithm mainly uses shift and exclusive or operation. The key is always the same during the encryption process [9]. Its security is guaranteed by the number of encryption wheels rather than the complexity of the algorithm.

TEA encryption algorithm has also been a very good application. For example, the QQTEA encryption algorithm used in QQ chat software developed by tencent is based on the standard TEA encryption algorithm, which uses the encryption rules of 16 times. QQ used some additional padding and interleaving algorithms to deal with the problem of encrypting indefinite length data.

Test the TEA encryption algorithm, the test results shown in Figure 3 below.

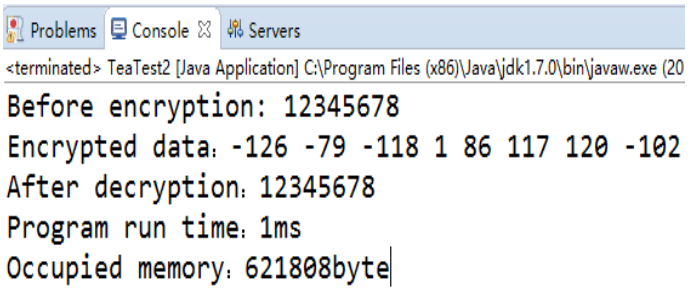


Figure 3: *Test Effect of TEA Algorithm*

2.4 Comparison of Algorithm Performance

In the above, figure 1, figure 2 and figure 3 respectively test the effect of each algorithm. The three algorithms are running on the same computer, and the input parameter is 12345678, the performance of the three algorithms to compare the results shown in table 1 below.

Table 2: Comparison of Algorithm Performance

Algorithm name	Input parameters	Program runtime	Take up ory
ECC Algorithm	12345678	97ms	3097752byte
DES Algorithm	12345678	119ms	1935744byte
TEA Algorithm	12345678	1ms	621808byte

It can be seen from Table 2 that the execution time and memory of the TEA algorithm are the least, but the TEA algorithm can only handle 8 bytes each operation. If the processing is more than 8 bytes, it needs to be encapsulated and encapsulated of the ciphertext data is generally an integer multiple of 8 bytes, no more than the 8 bytes than the original, completely in the acceptable range. As a result, this paper decided to introduce TEA algorithm to achieve the realization of lightweight encryption.

3. EXPERIMENT ANALYSIS

Build a messaging push environment by using the MQTT protocol to verify lightweight encryption. The server uses Apache Apollo to build a message push system server, and the client is divided into the publisher client and the subscriber client, which is to implement the message push function based on Eclipse Paho API programming. The specific implementation-n steps are as follows:

a) Firstly, download and open the Apache Apollo server and listen to the client connection.

b) Run the publisher client, connect the Apache Apollo server and send message content "test the TEA encryption algorithm", as shown in figure 4 below.

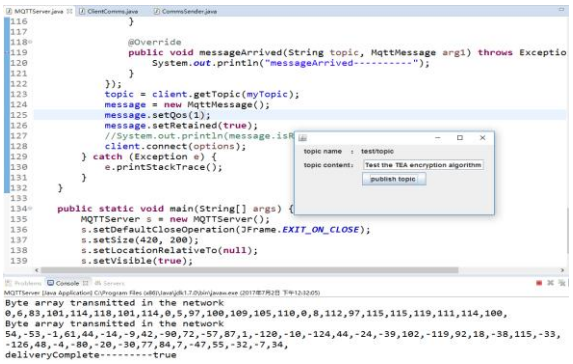


Figure 4: *Publish the Theme Interface for the Publisher Client*

c) Run the subscriber client and connects the Apache Apollo server on mobile terminal, as shown in Figure 5 below.

d)

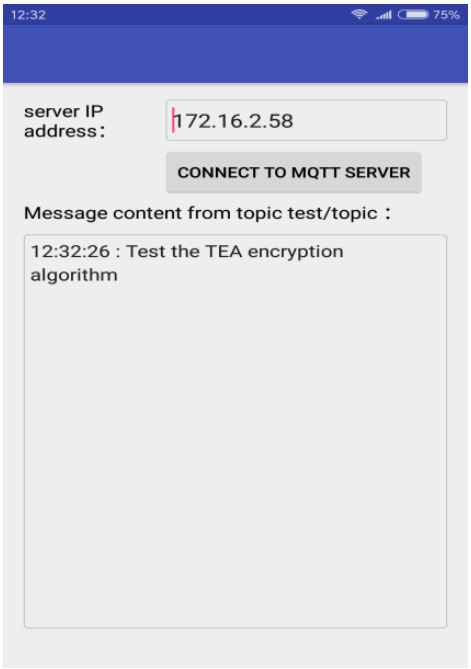


Figure 5: *The Subscriber Client Main Interface*

As the subscriber client is based on the Android platform development, the publisher client based on the windows platform development, and there will be text compatibility problems of different platforms. Therefore, a larger character set utf-8 is required for the publisher client, using the String.getBytes ("UTF-8") method to get the UTF-8 byte array as {-26,-75,-117,-24,-81,-107,84,69,65,-27,-118,-96,-27,-81,-122,-25,-82,-105,-26,-77,-107}, and encrypted with the ciphertextdata{5,-65,-107,-37,41,123,-12,112,37,8,-89,-91,-126,119,75,0,104,-59, 105,73,3,127,39,-65}. In figure 4, you can see that the final generation of ciphertext data is printed in the background. As shown in figure 5, you can see the normal plaintext message content.

Each byte in the byte array using the default encoding format occupies two bytes, and each byte that uses the UTF-8 encoded format occupies three bytes. Both English characters, punctuation and numbers occupy a word Section. Then you can calculate the byte array size is 21 bytes, which is using the default send message content UTF-8 encoding format. After TEA algorithm encrypted, the byte array size is 24 bytes [10]. There are only 3 bytes more than the original byte array, meet the system's lightweight encryption requirements. So it can be concluded that the experimental results are correct.

4. CONCLUSION

This article selects the MQTT protocol, which is simple, efficient, lightweight, based on the publish/subscribe pattern. The low bandwidth features of the MQTT protocol are popular in areas with limited bandwidth, but there are also some drawbacks. Therefore, this paper introduces the TEA algorithm of lightweight encryption algorithm in consideration of the two problems of lightweight and encryption. TEA algorithm can be effectively compatible with lightweight requirements and encryption requirements, the use of increasing Delta value after 32 rounds of shift and easy to calculate the encrypted data, and the ciphertext data size than the original data no more than 8 bytes. The experiment shows that the TEA encryption algorithm adopted in this paper is good enough to satisfy the demand of lightweight data transmission and encryption. However, TEA encryption algorithm used in this paper can be compatible with both lightweight data transmission and encryption requirements, but the encryption algorithm is not too difficult to understand, there is the possibility of being cracked. Therefore, further research on lightweight encryption is the next step in this article.

ACKNOWLEDGMENTS

As the research of the thesis is sponsored by National Natural Science Foundation of China (No: 61662017), major scientific research project of Guangxi higher education (No: 201201ZD012), Guilin Science and Technology Project Fund (No:2016010408) and Guangxi Graduate Innovation Project (No: SS201607), we would like to extend our sincere gratitude to them.

REFERENCES

[1] Lu, A.P., Yang, J.M., Li, F. 2014. A Comparative Study of Several Lightweight Encryption Algorithms [J]. Modern electronic technology, 2 (12), 37-41(In Chinese).

[2] Guo, K., Liu, Y., Ma, J. 2013. AMPS: An Adaptive Message Push Strategy for Ubiquitous Terminals[C]//Proceedings of the 2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing. IEEE Computer Society, 731 - 736.

[3] Ren, H., Ma, Y., Yang, H.B. 2014. Message push server based on MQTT protocol [J]. Application of computer system, 23 (03), 77-82(In Chinese).

[4] Yu, J.G., Geng, Y.F., Yang, Y.H.B. 2016. Design and implementation of message engine server based on MQTT protocol [J]. Small and microcomputer systems, 37 (10), 2238-2243(In Chinese).

[5] Liu, J., Sun, J., Xu, Z.Q. 2011. Target-based image lightweight encryption [J]. Journal of huazhong university of science and technology: natural science edition, 1 (6), 15-18(In Chinese).

[6] Li, L., Li, R.F., Li, K.L. 2014. Research on Power - based Encryption of Lightweight Presence Encryption Algorithm Based on Target Image [J]. Computer Application Research, 31 (3), 843-845(In Chinese).

[7] Singh, M., Rajan, M.A., Shivraj, V.L. 2015. Secure MQTT for Internet of Things (IoT)[C]// Fifth International Conference on Communication Systems and Network Technologies. IEEE.

[8] Bao, X. 2013. Design and Implementation of News Reading Software Based on Android Platform [J]. computer application, 33 (s2) (In Chinese).

[9] Guan, Y.Q., Li, H.B., Yu, B. 2014. Research and Application of MQTT Protocol on Android Platform [J]. Computer system applications, 23 (04), 197-200(In Chinese).

[10] Ji, Z., Ganchev, I., O'Droma. 2014. A Push-Notification Service for Use in the UCWW [C]. 2014 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, 318-322.