The Journal of Supercomputing

Providing a Lightweight Message Authentication Scheme for IoT Environment

--Manuscript Draft--

Manuscript Number:	SUPE-D-17-00004
Full Title:	Providing a Lightweight Message Authentication Scheme for IoT Environment
Article Type:	Manuscript
Keywords:	Internet of Things, MQTT Protocol, Constrained Device, SPEKE, Authentication, lightweight cryptography
Abstract:	The term IoT (Internet of Things) describes a world where all things have digital identity for themselves and allow computers to organize and manage them. In short, IoT is a technology in which the ability to send data via communications networks, including the Internet or Intranet, is provided for every being. The platform of IoT is placed on wireless radio waves which allow different devices to communicate with each other via the Internet, or in other words, it describes the idea of designing different devices which are able to communicate wirelessly to intercept and control via the Internet. Many devices in IoT have limited resources and use the battery. These embedded devices cannot usually use the high-level protocols; they use binary protocols such as lightweight MQTT instead. MQTT protocol is a pub/sub model-based protocol designed for devices with limited resources. Since these devices are connected directly to the Internet, they are vulnerable to DoS attacks which lead to quick battery-drain. Therefore, authentication must be done in an efficient way, and battery, memory and limiting computing power of the device must be considered. In this paper, a lightweight mechanism is proposed for message authentication. Its main idea is the message lightweight authentication in MQTT protocol without having to send on-line key through scenario of creation and management of the proposed key. In this way, only the authenticated messages are accepted and processing of invalid useless messages is prevented leading to save the energy. Security analysis of proposed mechanism has been done by ProVerif which is a perfect tool for verifying the security features of cryptographic algorithms. The results of communication overheads indicated that the proposed mechanism is a lightweight design that provides security and authentication in MQTT protocol.

Providing a Lightweight Message Authentication Scheme for IoT Environment

Anahita Aghajani[1], Seyed Hamid Haj Seyed Javadi[2], Mahdi Mollamotalebi[3]

[1,3] Department of Computer, Buinzahra Branch, Islamic Azad University, Buinzahra, Iran

[2] Department of Computer, Parand Branch, Islamic Azad University, Parand, Iran

Abstract

The term IoT (Internet of Things) describes a world where all things have digital identity for themselves and allow computers to organize and manage them. In short, IoT is a technology in which the ability to send data via communications networks, including the Internet or Intranet, is provided for every being. The platform of IoT is placed on wireless radio waves which allow different devices to communicate with each other via the Internet, or in other words, it describes the idea of designing different devices which are able to communicate wirelessly to intercept and control via the Internet. Many devices in IoT have limited resources and use the battery. These embedded devices cannot usually use the high-level protocols; they use binary protocols such as lightweight MQTT instead. MQTT protocol is a pub/sub model-based protocol designed for devices with limited resources. Since these devices are connected directly to the Internet, they are vulnerable to DoS attacks which lead to quick battery-drain. Therefore, authentication must be done in an efficient way, and battery, memory and limiting computing power of the device must be considered. In this paper, a lightweight mechanism is proposed for message authentication. Its main idea is the message lightweight authentication in MQTT protocol without having to send on-line key through scenario of creation and management of the proposed key. In this way, only the authenticated messages are accepted and processing of invalid useless messages is prevented leading to save the energy. Security analysis of proposed mechanism has been done by ProVerif which is a perfect tool for verifying the security features of cryptographic algorithms. The results of communication overheads indicated that the proposed mechanism is a lightweight design that provides security and authentication in MQTT protocol.

Keywords:
Internet of Things, MQTT Protocol, Constrained Device, SPEKE, Authentication, lightweight cryptography

I. Introduction

Today's real world embedded devices usually lacks the ability to handle high-level protocols like HTTP and they may be served better by lightweight binary protocols. In this regard, Andy Stanford-Clark, and Arlen Nipper proposed in 1999 what now has become the

MQTT[12][8] protocol. MQTT is an application layer protocol suitable for the constrained devices. It is fast, lightweight, power efficient and provides various levels of Quality of Service [9]. It handles broker-based lightweight messaging [10]. The characteristics of MQTT have made it ideal to be used in environments with limited resources (for example, the network with high usage costs and lack of reliability, or embedded devices with limited processing/memory capacities) [11]. MQTT uses a topic-based publish/subscribe architecture. This means that when a client publishes a message M to a particular topic T, then all the clients subscribed to the topic T will receive the message M. Like HyperText Transfer Protocol (HTTP), MQTT relies on Transmission Control Protocol (TCP) and Internet Protocol (IP) as its underlying layers. However, compared to HTTP, MQTT is designed to have lower protocol overheads [12].

If IoT devices are directly connected to Internet without protection by firewall or gateway, they will be exposed to many security threats. One of these threats is preventing the devices from going into sleep mode which runs their battery out very quickly. Such attack is usually called “Denial of Sleep” which is usually a real threat against small battery powered devices [6]. Limited devices are rarely protected by intrusion detection and prevention systems used for servers on the Internet. However, different firewalls and gateways are used to protect these devices from traffic which is received from outside the local network [7].

In order to establish the security in MQTT V3.1 protocol, it is possible to define a username and a password for MQTT in the connect packet. In MQTT protocol, encryption could be done using SSL/TLS despite of the fact that SSL is not the most lightweight protocol and it imposes significant overloads over the network. Although a cryptography algorithm can be used to establish security, for keeping MQTT protocol simple and lightweight, such encryption is not included in the protocol. Therefore, those lightweight applications which provide authenticity and confidentiality will be useful although confidentiality is not always necessary. In some uses, authenticity and integrity of data in computer systems and networks is regarded as a necessity. In particular, two parties communicating through an insecure channel need a method by which helps one party to validate the information sent by another party as authentic and unmodified [8].

In those uses which authenticity of information is significant, one could use a lightweight scheme. The proposed scheme in this paper, M_MAC (MQTT_Message Authentication Code), is an extension of MQTT protocol which adds MAC to the payload of PUBLISH packets. This means that no new field needs to be defined, and essential modifications of the protocol are few. Since MAC is a part of data of the message, its overload is transparent for those devices which use M_MAC extension. The main idea of proposed scheme M_MAC is the lightweight authentication of message in MQTT protocol without requiring a key to be sent online. This is due to the scenario of developing and management of the suggested key. As a result, only those messages are accepted which are authenticated. In addition, unnecessary processing of invalid messages is reduced which subsequently leads to saving the power.

II. Related work

MQTT is based on pub/sub model [31] in which clients connect to the broker for exchanging messages [12]. Clients that are interested in a specific topic should be a subscriber of that in order to receive messages sent to it. Any entity that is connected to the broker, having a message and a title, can publish a message [12]. Figure 1 shows the message format in MQTT.

Bit	7	6	5	4	3	2	1	0
byte1	Message Type				DUP flag	QoS level		RETAIN
byte2	Remaining Length							
Variable Header								
Payload								

Figure 1.MQTT Header & Payload [12]

As shown in Figure 1, each message in MQTT consists a header and payload. Header includes a fixed and a variable part. Remaining Length indicates the number of bytes remaining from the current message containing data of variable header and payload.

In the future, the need for secure communication between a large number of objects and people in the context of IoT is one of the biggest challenges that we have to pay attention in order to prevent many problems of personal privacy [1]. In this respect, European Union is considering IoT regulations [9]. In the M2M [30] literature, the required security and privacy are not provided and MQTT is not an exception [1]. MQTT Standard does not specify requirements and security features that must be supported in the implementations.

The OASIS MQTT Security Committee is created to provide MQTT security solutions and integration with other security standards. In the subcommittee page [13], a set of documents has been formed that includes discussion describing how MQTT solutions may be made secure, and how this may match with existing security standards. MQTT Standard has not specified requirements and security features that must be supported in the implementations, and many of the open source and proprietary implementations available, only support some of the features of authentication and authorization [14].
MQTT V3.1 [12] has defined no security management function but authentication protocol using a simple user name and password embedded in the connect packet. Public draft of MQTT V3.1.1 [13] includes a section with tips about security threats and mechanisms that should be provided in MQTT implementations. However, any implementation of MQTT is free to implement the non-standard version of its security functions for authentication, access control, integrity, and privacy. Securing technical checklists available in the public draft include mutual authentication between client and server, integrity/privacy of messages and control packets, non-repudiation messages and identification of malicious and abnormal behaviors of clients and servers [14].

Neisse et al. [14] provided SecKit model-based security tool [15] to address the security aspects of IoT systems, including requirements for privacy and protection of information. SecKit allows the end user to design and implement a set of security and privacy policies that

are completely customized. The end user decides for the optimal trade-off between information disclosure and privacy/security. The main idea in [14] is the implementation developed by the open source MQTT broker which is integrated with SecKit and is able to handle the security policy requirements.

Singh et al. [16] proposed a secure version of MQTT and MQTT-SN [26] protocols called SMQTT in which the security features are added to the existing MQTT protocol based on KP/CP-ABE [27][28] using lightweight Elliptic Curve Cryptography (ECC) [29]. In addition, feasibility of security has been studied for MQTT using CP/KP-ABE. The advantage of using Attribute Based Encryption is supporting broadcast encryption due to its inherent design (the message is delivered to the intended users with an encryption), and therefore, it is suitable for IoT applications.

One of the ways to encrypt communications in MQTT is using TLS [17][25], a cryptographic protocol that uses handshake mechanism to negotiate about various parameters in order to create a secure connection between the client and the broker. It provides a secure communication channel between the client and the broker. Using TLS, we can be sure that the contents of the communication cannot be read or altered by third parties. In some studies, the system architecture is integrated with the hardware security controller to process the TLS client authentication, although the use of MQTT on TLS has its own difficulties. Given that the security has processing costs and communication overheads, it can be problematic for limited devices that are not designed to perform complex tasks. If long-term TCP connections are established by MQTT, the TLS overhead, particularly handshake protocol overhead, is negligible. But if there are a large number of connections and MQTT connections are short-term, and there is no possibility of session resumption, the handshake protocol overhead is significant, especially if small MQTT messages are needed to be exchanged.

III. Proposed scheme

In this section, our proposed scheme for lightweight authentication is described in detail. In the proposed scheme, the key is not sent on-line to avoid being stolen. Instead, it uses a low computational overhead method by the password placed between the broker and the specific client to make the required keys and enhance the security of data transmission by MQTT. The scheme is divided into two steps as follows:

a. Key exchange and initialization

At this step, the Simple Password Exponential Key Exchange (SPEKE) protocol is used where p is considered to be a prime number as $p = 2q + 1$, q is a prime number and SPEKE is defined in the subset of prime integers (Z_q^*). We assume that the two sides (client & broker) have shared password π and want to exchange a password-authenticated key. SPEKE defines function f that maps the password π to a generator. For example, $f(\pi) = \pi^2 \, mod \, p$ in SPEKE or $f(\pi) = (h(\pi))^2 \, mod \, p$ in revised version of SPEKE, where h is a cryptographic hash function (e.g., SHA-256 or AES-128). Suppose g is a generator of π, g = f (π). The squaring of h (π) ensures that generator g will not be placed in subgroup {1, p-1}. Details of SPEKE are as follows:

1. Client → Broker: Send $A = g^a \bmod p$, where a is randomly seleted in Z_q^*.
2. Broker → Client: Send $B = g^b \bmod p$, where b is randomly seleted in Z_q^*.
3. Client & Broker: Compute $K_{SPEKE} = B^a \bmod p$ and $K_{SPEKE} = A^b \bmod p$, respectively.

At this step, client and broker have shared the secret key, K_{SPEKE}.

b. Authentication

The primary key obtained from the SPEKE key distribution method in the previous step is called K_{SPEKE} and is used for key generation in the calculation of CMAC. According to equation (1), K_{SPEKE} and $userID \,||\, clientID$ are used as input of a pseudo-random function (PRF) to create a key for the specific session of a user and a client.

$$K_S = f_{PRF}\,(K_{SPEKE}, userID \,||\, clientID) \qquad (1)$$

According to equation (2), K_S with an initial PID from MQTT publish packet is given to a pseudo-random function as input to produce CMAC key for a specific session. A session is known with the initial PID of the first received packet. For MQTT, this PID is a 16-bit value in the header of PUBLISH messages that is transmitted; and it will automatically increment for each message.

$$K_{CMAC} = f_{PRF}\,(K_S\,, PID) \qquad (2)$$

In M_MAC, an attacker can obtain MQTT request with a valid MAC which is in the payload of MQTT message, and send it again in the next session. This will help attacker to break M_MAC security and send a particular packet many times and request the resources related to this packet. There is no clear distinguishing characteristic that enables the recipient to identify that the packet is replayed, because the packet is the same as a valid packet. The occurrence of such an attack causes problems for recipients of message; and a way is needed to prevent replay attacks. To solve the above problem, when the client sends $A = g^a \bmod p$ to the broker, a step is also sent with it. If $PID \bmod Step$ is zero, K_{SPEKE} is recalculated. With this, the key is recalculated after exchanging a specified number of messages.

Another advantage of using M_MAC is that it is compatible with devices that use MQTT protocol but do not implement M_MAC. When a MQTT request is sent to the broker, by the data received in the CONNECT message, the broker will be noticed that the M_MAC extension is implemented or not. According to equation (3), if the sender has a password and sends it to the broker, then the broker will perform the authentication as usual as MQTT. If the sender has a password but does not send it to the broker, then the broker will use M_MAC.

$$\forall\, P_i \in \{\text{publisher}\} \mid P_i \;has\; pass \;\&\; pass\; in\; connect(P_i)\; is\; null \;\rightarrow Switch\; to\; SPEKE\; mode \qquad (3)$$

Figure 2 shows an example of how to apply the M_MAC scheme. This figure describes how MQTT is developed to provide authentication for a client that communicates with the broker.

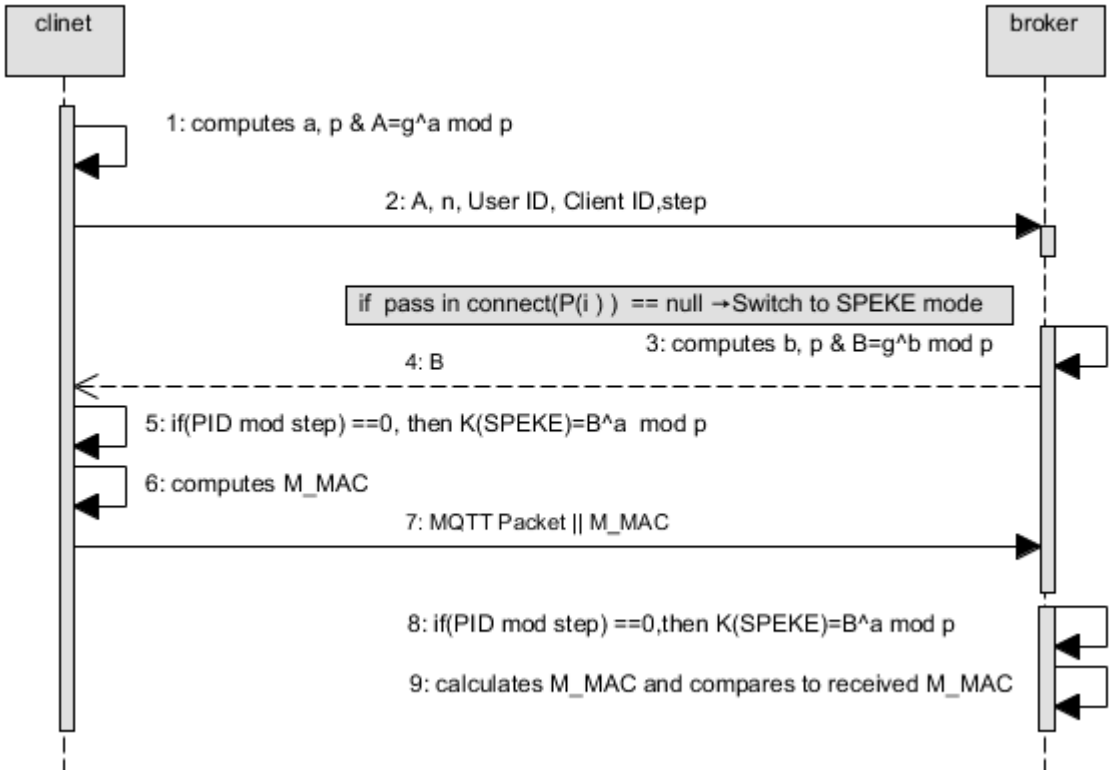

Figure 2. The proposed scheme

As seen in Figure 2, there are two devices on the network: a broker and a client. The following steps prepare clients and then exchange the authenticated messages between the client and the broker:

1. Client: Compute g = $f(\pi) = (h(\pi))^2 \bmod p$.
2. Client: Compute $A = g^a \bmod p$, where a is randomly seleted in Z_q^*.
3. Client → Broker: Send A, n, step, UserId & ClientId
4. Broker investigates whether there is a need to use the SPEKE protocol. M_MAC can also act as both one-way and two-way authentication, which means that the broker can choose to embed M_MAC or not. First, it is investigated whether a password have been received for UserId or not. If a password is defined for that UserId, but the password field is blank in the received connect message, it starts calculating K_{SPEKE}.
5. Broker: Compute g = $f(\pi) = (h(\pi))^2 \bmod p$.
6. Broker: Compute $B = g^b \bmod p$, where b is randomly seleted in Z_q^*.
7. Broker → Client: Send B.
8. Client & Broker: Compute $K_{SPEKE} = B^a \bmod p$ and $K_{SPEKE} = A^b \bmod p$, respectively.
9. Client and broker give K_{SPEKE} and userID || clientID to the CMAC as inputs, and calculate K_S as follows:

$$K_S = f_{PRF}\,(K_{SPEKE}, userID \,||\, clientID)$$

10. Client and broker give K_S and initial PID to the CMAC function as inputs, and calculate K_{CMAC} as follows:

$$K_{CMAC} = f_{PRF}(K_S, PID)$$

11. Client calculates CMAC using K_{CMAC} and MQTT payload.
12. Client sends the first MQTT packet which is protected by M_MAC.
13. Client receives a packet and checks to see if the initial PID matches. If so, broker generates CMAC of the received packets using K_{CMAC} and MQTT payload and checks to see if it complies with the CMAC available in the packet. If so, the packet is accepted and marked as read; otherwise it is rejected.
14. Broker: Replies to the message
15. Now client uses the PID of the second packet to create CMAC for the second packet sent.

With regard to the above, the SPEKE key exchange protocol was used at first, and an identical secret key was shared by the client and the broker using UserID and password of the client, without having to send the password online. Then, using the generated key and a pseudo-random function, the generated CMAC was sent to the broker with the message. Finally, by comparing the CMAC received from the client and computed CMAC, broker accepts or rejects the message. Thus, using SPEKE and CMAC protocols and without having to send the key online, a lightweight design is proposed for message authentication in MQTT protocol.

IV. Implementation and evaluation of the proposed method

In this section, implementation of the proposed scheme is described in details. The proposed scheme is implemented and tested by ProVerif [21] which uses Dolev-Yao [20] model. It assumes that cryptography is perfect, and tests protocol errors against an active attacker (that can receive/send messages, or perform cryptographic operations if the relevant keys are available). Given that ProVerif is appropriate especially for authentication and confidentiality, it is ideal for the purposes of this article. It uses the language specification based on the pure Pi-calculus [22] which is a tool for modeling the systems that are in synchronized connection and are implemented in parallel. As the input, ProVerif accepts the encryption protocol model (which is mentioned as π-calculus) as well as security features that must be proven [21]. The security features are modeled as derivable queries. ProVerif supports a number of cryptographic primitives, including encryption/decryption (symmetric and asymmetric), digital signatures, hash functions, and Diffie-Hellman [32] key agreement [23].

To analyze the proposed scheme, a public insecure channel, called *chan*, is created for communication between client and broker:

```
free chan:channel
```

The public parameters (fixed and variable) are defined as follows:

```
free a:bitstring [private]
free b:bitstring [private]
free k_Clinet:bitstring [private]
free k_Broker:bitstring [private]
free pass:bitstring [private]
const g:bitstring [private]
free UserID:bitstring [private]
free ClientID:bitstring [private]
free step:bitstring [private]
free PID:bitstring [private]
free CMAC_Client:bitstring [private]
free CMAC_Broker:bitstring [private]
free payload:bitstring [private]
free MQTT_Packet:bitstring [private]
```

CMAC_Client and CMAC_Broker are authentication codes that are produced by client and broker, respectively. Payload is the data of the message and MQTT_Packet is the packet of MQTT message that is sent from the client to the broker. In addition, the following functions are defined as constructors:

```
fun exp(bitstring,bitstring): bitstring
fun CMAC(bitstring, bitstring): bitstring
fun square(bitstring):bitstring
fun mod(bitstring,bitstring):bitstring
fun conct(bitstring,bitstring): bitstring
```

To verify the properties of secrecy and reachability of the proposed scheme, the following queries are implemented. Attacker (M) checks the accessibility of M by attacker; it means that the accessibility of k_CMAC_Client key for attackers, is tested.

```
query attacker(k_CMAC_Client)
query attacker(k_CMAC_Broker)
```

To investigate the one-to-one relationship between each of the number of protocol performed for each participant, the following queries are executed. One-to-one relation means that the broker implements only one authentication for each event begun by the client (each message sent) in MQTT.

```
query id:bitstring; inj-event(endClient(a))==>inj-event(beginClient(a))
query id:bitstring; inj-event(endBroker(a))==>inj-event(beginBroker(a))
```

We have also defined the following equation. This equation means that $(g^a)^b = (g^b)^a$ is equivalent to the condition in which client and broker calculates $K = B^a \ mod \ p$ and $K = A^b \ mod \ p$ respectively.

```
equation forall a:   bitstring, b:   bitstring; exp(exp(g,a),b)   =   exp(exp(g,b),a).
```

Moreover, to analyze characteristics of the certification, four events are defined as following:

```
event beginClient(bitstring)
event endClient(bitstring)
event beginBroker(bitstring)
event endBroker(bitstring)
```

Start and end events of *beginClient* and *endClient* are considered for the client, and events of *beginBroker* and *endBroker* are considered for the broker. Two processes of *pClient* and *pBroker* are defined for the client and the broker, respectively. *pClient* calculates A (the amount that the client calculates using a, g and p as $A = g^a \ mod \ p$), and sends it through insecure channels to the broker using *out* function, and receives B (the amount that broker calculates using b, g and p as $B = g^b \ mod \ p$) from the broker through insecure channels using *in* function. Then, client calculates $K\_Client = B^a \ mod \ p$ (a key that the client obtains using SPEKE protocol). Client produces K_S (a key calculated per session) using CMAC function with inputs of $userID \,||\, clientID$ and K_Client. Then, it produces K_CMAC_Client using CMAC function with PID and K_S as inputs of the first packet sent. Finally, $CMAC\_Client$ is produced by CMAC function and sent to the broker through insecure channels.

```
(*Client*)

let pClient=
let g = square(hashFun(pass)) in
let A = exp(g,a) in
out(chan,(A,n,UserID,ClientID,step))
in(chan,(xB:bitstring))
event beginClient(a)
let k_Clinet = (exp(xB,a)) in
let ks = CMAC   (k_Clinet,conct(UserID,ClientID)) in
let k_CMAC_Client = CMAC(ks,PID) in
let CMAC_Client = CMAC(payload,k_CMAC) in
out (chan,(MQTT_Packet,CMAC_Client))
event endClient(a)
```

pBroker calculates B, and sends it to the client through public insecure channels using the function *out*. Then, the broker calculates $K\_Broker = A^b \ mod \ p$. Broker produces K_S using CMAC function with $userID \,||\, clientID$ and K_Broker as inputs. Then it produces *K_CMAC_Broker* using CMAC function with PID of the first packet and K_S as inputs. Finally, *CMAC_Broker* is produced by the CMAC. The broker compares *CMAC_Broker* with *CMAC_Client* which is received from the client using the function *in*, and confirms the message if they are equal.

```
(*Broker*)

let pBroker=
let g = square(hashFun (pass)) in
let B = exp(g,b) in
```

```
out(chan,(B))
event beginBroker(b)
in(chan,(xA:bitstring,xn:bitstring,xUserID:bitstring,xClientID:bitstring,xstep:bitstring))
let k_Broker = (exp(xA,b)) in
in(chan,(xMQTT_Packet:bitstring,xCMAC_Client:bitstring,xpayload:bitstring))
let ks = CMAC(k_Broker,conct(xUserID,xClientID)) in
let k_CMAC_Broker = CMAC(ks,PID) in
let CMAC_Broker = CMAC(xpayload,k_Broker) in
if(xCMAC_Client = CMAC_Broker) then
event endBroker(b)
else 0
```

The proposed scheme is modeled as a parallel execution of two distinct processes. Now, as described below, we execute the queries to verify the security of keys:

```
process ((!pClient) | (!pBroker))
```

Then, ProVerif executes the queries of *query attacker(k_CMAC_Client)* and *query attacker(k_CMAC_Broker)*. The result obtained is *RESULT not attacker<M> is ture* and represents the security and confidentiality of *k_CMAC_Client* and *k_CMAC_Broker* keys in the protocol. Then, the performance of the proposed scheme is evaluated in terms of energy consumption and communication overhead.

Communication overhead is measured by the number of extra bytes transferred between client and broker in each communication. Table 1 shows communication overhead for the proposed scheme M_MAC, and MQTT protocol that uses TLS to create security.

Table 1. Communication overhead analysis

Scheme	Overhead(byte)
MQTT with TLS	2697
M_MAC (step=1)	768

The results obtained for the communication overhead shown in Table 1 indicates that the total number of bytes needed for handshake in TLS is 2697 bytes. The total number of bytes that are transferred between client and broker in the proposed scheme is between 258 and 768 bytes; because A and B are between 1 and 256 bytes, and p is equal to 256 bytes.

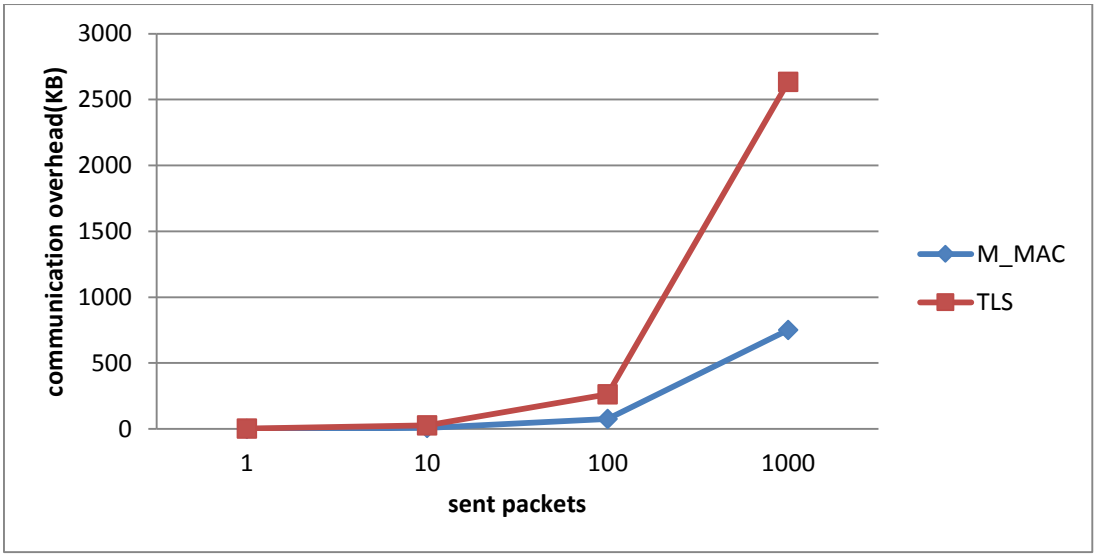

Figure 3. Comparison of communication overhead for M_MAC and TLS

As seen in Figure 3, in the conditions with large number of connections, the proposed scheme M_MAC, has less communication overheads compared to TLS, which indicates that it is more appropriate than TLS for devices with limited resources.

In this paper, *Tmote Sky* sensors are considered as devices with limited resources. Tmote Sky is an ultra low power wireless module which is used in sensor networks for monitoring and rapid application prototyping [24]. Tmote Sky sensors use 2 AA batteries with the total amount of 30780 Joules [24].

Power consumption of a sensor while it sends or receives a packet (even when it discards a packet), can be described by the following linear equation [24]:

$$\mathbf{E} = \mathbf{0.12} \times \mathbf{size} \qquad (4)$$

In equation (4), *size* is the number of bytes needed for handshake in TLS or the number of transferred bytes in M_MAC. The fixed value of 0.12 mJ is the amount of power consumed for sending a byte.

Table 2. Computation cost analysis

Scheme	Overhead(mJ)
MQTT with TLS	323.64
M_MAC(step=1)	92.16

The results shown in Table 2 indicate that the computation energy consumption for handshake in TLS is 323.64 mJ, while it is 92.16 mJ for M_MAC.

During sending a message (according to Tables 1 and 2) using M_MAC (and if the step is equal to 1), it was observed that communicational and computational overhead was improved

at least 70% compared to the case that TLS is used. This indicates that greater values of steps results to more power consumption reductions.

V. Conclusion

Security is a major concern in IoT, and implementing secure protocols to use the resources as optimal is a challenging issue. Constrained devices in the IoT are subject to a number of security threats, especially if they are connected to the Internet directly and unprotected. The proposed scheme M_MAC is an extension of MQTT protocol which is consisted of adding MAC to the payload of PUBLISH packets. This means that no new field needs to be defined, and essential modifications of the protocol are few. Since MAC is a part of data in the message, its overload is transparent for those devices which use M_MAC extension.

The main idea of M_MAC is having lightweight message authentication in MQTT protocol without needs to send on-line key. In this way, only the authenticated messages are accepted and processing of invalid useless messages is prevented, saving the energy. M_MAC can be deployed on many devices that have limited resources, and enables devices to communicate with less energy and without bearing the overhead of TLS for each connect, so that they can send data securely and when needed go to sleep mode in order to save energy consumption.

The results of implementation, and evaluation of the proposed scheme compared to TLS protocol, indicate that the proposed scheme detects invalid messages with less computational and communicational overheads especially in large number of connections.

References

[1] M. Collina, G. E. Corazza and A. Vanelli-Coralli, "Introducing the QEST broker:Scaling the IoT by bridging MQTT and REST," *IEEE International Symposium on*, p. 36–41, 2012.

[2] V. Karagiannis, P. Chatzimisios, F. Vazquez-gallego and J. Alonso-zarate, "A Survey on Application Layer Protocols for the Internet of Things Research motivation," p. 1–10, 2015.

[3] B. S. Ullas, S. Anush, J. Roopa and M. Govinda, "Machine to Machine Communication for Smart Systems using MQTT," vol. 3, no. 3, pp. 8242 -8248, 2014.

[4] C. Pereira and A. Aguiar, "Towards Efficient Mobile M2M Communications:Survey and Open Challenges," p. 19582–19608, 2014.

[5] X. Ma, A. Valera, H. Tan and C. K. Tan, "Performance Evaluation of MQTT and CoAP via a Common Middleware.".

[6] D. Raymond, R. Marchany, M. Brownfield and S. Midkiff, "Effects of Denial-of-Sleep Attacks on Wireless Sensor Network MAC Protocols," IEEE Transactions on Vehicular Technology, vol. 58, p. 367–380, January 2009.

[7] C. Lerche, N. Laum, F. Golatowski, D. Timmermann and C. Niedermeier, "Connecting the web with the web of things: lessons learned from implementing a CoAP-HTTP proxy," IEEE 9th International Conference on Mobile Ad-Hoc and Sensor Systems (MASS 2012), pp. 1-8, 2012.

[8] M. Bellare, R. Canetti and H. Krawczyk, "Keying Hash Functions for Message," p. 1–15, 1996.

[9] "EU investigating IoT regulations," [Online]. Available: http://bit.ly/HyYSb2. [Accessed 17 Jun 2016].

[10] D. Giusto, A. Iera, G. Morabito, L. Atzori, C. Medaglia and A. and Serbanati, "An overview of privacy and security issues in the internet of things," Springer New York, p. 389–395, 2010.

[11] T. Heer, O. Garcia-Morchon, R. Hummen, S. L. Keoh, S. S. Kumar, and K. Wehrle, "Security Challenges in the IP-based Internet of Things", Wireless Personal Communications, vol. 61, no. 3, pp. 527–542, Sep. 2011 [Online].

[12] "mqtt ver3.1.1," 29 Oct 2014. [Online]. Available: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.doc. [Accessed 17 Jul 2016].

[13] HiveMQ, "Lightweight authentication and authorization for mqtt with stormpath," 2014. [Online]. Available: http://www.hivemq.com/lightweight- authentication-authorization-mqtt-stormpath. [Accessed 17 Jul 2016].

[14] R. Neisse, G. Steri and G. Baldini, "Enforcement of Security Policy Rules for the Internet of Things," Third International Workshop on Internet of Things (IoT) Communications and Technologies broker, p. 165–172, 2014.

[15] R. Neisse, I. Fovino, G. Baldini, V. Stavroulaki, P. Vlacheas and a. R. Giaffreda, "A model-based security toolkit for the internet of things," The 9th International Conference on Availability, Reliability and Security (ARES), 2014.

[16] M. Singh, M. A. Rajan, V. L. Shivraj and P. Balamuralidhar, "Secure MQTT for Internet of Things (IoT)," *5th International Conference on Communication Systems and Network Technologies, CSNT*, p. 746–751, 2015.

[17] C. Lesjak, D. Hein, M. Hofmann, M. Maritsch, A. Aldrian and P. Priller, "Securing smart maintenance services: Hardware-security and TLS for MQTT," IEEE International Conference on Industrial Informatics, 2015.

[18] D. Jablon, "Strong password-only authenticated key exchange," ACM SIGCOMM Computer Communication Review, vol. 26, no. 5, 1996.

[19] D. Jablon, "Extended password key exchange protocols immune to dictionary attack," Sixth IEEE Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, p. 248–255, 1997.

[20] L. Chen and M. Ryan, "Attack, solution and verification for shared authorisation data in TCG TPM," FAST'09 Proceedings of the 6th international conference on Formal Aspects in Security and Trust, pp. 201-216, 2009.

[21] M. Abadi, B. Blanchet and H. Comon-Lundh, "Models and proofs of protocol security: A progress report," Computer aided verification. Springer, p. 35–49, 2009.

[22] Q. Xie, "A new authenticated key agreement for session initiation protocol," International

Journal of Communication System, vol. 25, no. 1, p. 47–54, 2012.

[23] Q. Xie, N. Dong, X. Tan, D. Wong and G. Wang, "Improvement of a three-party password-based key exchange protocol with formal verification," Inf Technol Control , vol. 42, no. 3, 2013.

[24] M. Amiri, "Measurements of energy consumption and execution time of different operations on Tmote Sky sensor motes," MSc thesis, MASARYK UNIVERSITY, 2010.