Efficient and Robust Security Architecture for Enhancing Security of Message Queue Telemetry Transport Protocol in Internet of Things Applications

Atul Oak
Ph.D. Research Scholar
Dept. of Electrical Engineering
Veermata Jijabai Technological Institute (VJTI)
Mumbai, India

R.D. Daruwala
Professor
Dept. of Electrical Engineering
Veermata Jijabai Technological Institute (VJTI)
Mumbai, India

ABSTRACT
Internet of Things (IoT) technology is one of the contributors to Industry 4.0 revolution. IoT is a system of interconnected devices called as things or nodes. Data is shared between such two physical devices like a sensor and a smart mobile phone using an existing internet and different types of network infrastructures. IoT systems are usually deployed using special protocols called light weight protocols which are different from the traditional internet based communication protocols. There are many constraints on devices used in IoT system such as use of low end micro-controllers with limited computing power, limited power consumption, and use of unreliable networks with low bandwidth. To work with and manage such devices with constraints, different types of light weight protocols have been developed and used in the IoT systems. Message Queue Telemetry and Transport (MQTT) is machine to machine (M2M) light weight data protocol most commonly used in the deployment of IoT systems and for sharing and delivery of data at application layer. When data is shared in the IoT system between the two physical devices with MQTT protocol, the security of data is a very critical requirement for reliability and adaptation of IoT systems. MQTT protocol being a light weight protocol does not provide any built in powerful and robust security techniques and it only supports very basic security mechanism like a password authentication. When used, this password is also shared in a clear text and it may be visible to the intruders. Due to the weak security features in MQTT protocol, many times the data is compromised and it is vulnerable to different types of malicious attacks like sniffing attack. Most common way to provide security to the MQTT based IoT application is at the transport layer by using an existing Transport Layer Security (TLS) protocol. TLS protocol may be suitable for internet applications like web applications and browsing but it is not suitable and designed for use with constrained IoT devices which works with light weight protocols.TLS protocol demands lots of complex computations and it needs more powerful resources like high end microprocessors and large memory which are only available with general computing machines like laptop or desktop computers. TLS approach may not provide end to end security in IoT applications since MQTT is asynchronous protocol. To achieve end to end security independent of TLS, robust security architecture is proposed in the IoT systems for MQTT protocol. These researches aims at developing efficient security architecture for MQTT protocol and improve the security of MQTT protocol.

General Terms
Industry 4.0, Technology, Internet, Devices, Attack,

Keywords
Internet of Things, Protocol, MQTT, Security, Architecture

1. INTRODUCTION
Internet is an interconnection of many heterogeneous networks of computers. Internet is widely used for people to people (P2P) communication using applications like email or browsing. In 1999, Kevin Ashton from MIT thought of sharing information electronically between the two physical devices [3]. Information sharing between the two physical devices or things through the existing infrastructure of an internet is called Internet of Things (IoT). IoT is a type of thing to thing (T2T) communication or device to device (D2D) communication and it can be considered as a next step of people to people (P2P) communication. IoT can be considered as connectivity in three dimensions like connectivity at any time, connectivity at any place and connectivity of anything. IoT is a network of physical objects built with sensors like vehicles, buildings or a network of devices like sensors which are embedded with required electronics with driving software and network connectivity. It enables all these objects or devices to collect and share the information. The deployment of IoT applications is increasing rapidly. According to IoT Analytics report, global IoT market size grew 22% in 2021[13] as shown in figure 1.

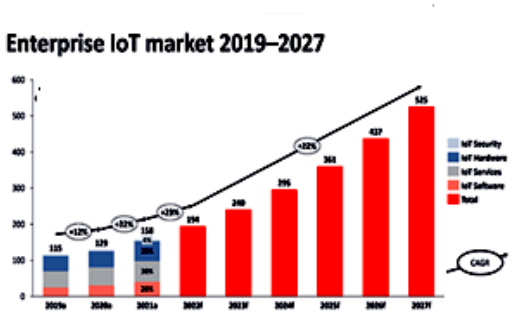

Fig 1: Growth of IoT

Gartner report has already estimated that by year 2021 about 20 billion devices are already connected to the internet. Applications like vending machines, connected cars, jet engines and more than 65% of enterprises will adopt IoT products [6][11][14] in coming years..

2. LAYERS IN IoT SYSTEM
IoT system deployment normally follows layered architecture like Open System Interconnection (OSI) model or a Transmission Control Protocol/Internet Protocol (TCP/IP)

model. Though there is no standard architecture defined for IoT systems, one of the most common layered architecture is called Service Oriented Architecture (SOA) of IoT [1]. There are four layers in SOA architecture as follows:

1. Perception layer: This is a physical layer of IoT system which measures and collects the information from different types of devices. These devices are light weight devices with lots of physical and operating constraints like limited or no computational power, limited power consumption and limited resources like memory.
2. Network layer: This layer of IoT system is responsible for routing of information using either Medium Access Control (MAC) address or Internet Protocol (IP) address. IoT access technologies like IEEE 802.15.4 called Low Rate – Wireless Personal Area Network (LR-WPAN) is widely used at perception layer and network layer.
3. Service layer: This layer of IoT system provides different services to application layer and works like a business logic. This layer may be further divided in more sub layers depending on the functionality.
4. Application layer: User interacts with the IoT system using an application layer and may share, view or display the collected information.

Due to the various constraints on IoT system, special light weight protocols are required for working with low level devices at different layers. There are also other layered models for IoT such as five layered model referred in the literature of IoT.

3. PROBLEM STATEMENT

MQTT protocol is not designed to be secure protocol. Most of the IoT systems using MQTT protocol rely on existing SSL/TLS protocol for security. SSL/TLS protocol is not been designed for IoT systems. SSL/TLS protocol do not provide end to end security especially with MQTT protocol which is based on publish subscribe architecture. SSL/TLS protocols are heavy weight protocols for resource constrained devices used in IoT systems. MQTT protocol being a light weight protocol does not provide any direct, powerful and efficient security mechanism for sharing information using unreliable network. It is observed during study of literature that many IoT systems running on MQTT protocol do not provide any security measure due to many challenges in providing security and SSL/TLS implementation is not practical. When adequate security is not provided for sharing of information, such information is vulnerable to different types of malicious attacks and IoT system is compromised. Security is a very fundamental requirement to protect the information. End to end security can be achieved in IoT system using standard encryption techniques. Key management like computation of keys, sharing and distribution of keys are the main challenges using the resource constrained devices in IoT systems. Identification of devices, authorization of devices is other security challenges in MQTT based IoT systems. To mitigate these different types of security challenges, more novel approach is do design a robust security architecture framework around MQTT protocol so that different security challenges can be handled in the future IoT systems running on MQTT protocol.

4. APPLICATION LAYER PROTOCOLS

The application layer is the topmost layer of the IoT layered architecture which provides the services requested by user. For example, the application layer can provide temperature and humidity measurements to the user requesting for such a data [16]. Significance of application layer is an ability to provide high-quality smart services to meet the requirements of the users. Many different IoT applications like smart city, smart healthcare, and smart factory can be implemented within this level. An application support sub layer called service layer also called middleware supports all sorts of business services and used to realize intelligent computation. Resource allocation could be implemented throughout specific middleware and cloud computing platforms. Application layer provides a direct interface to the end user where information generated in an IoT system can be delivered. Application layer is crucial layer and it is often an overlooked challenge in terms of security and quality of service. Application layer of IoT system widely uses five major protocols [9] [8] as follows

1. Constrained Application Protocol (CoAP)
2. Message Queue Telemetry and Transport Protocol (MQTT)
3. Extensible Message Presence Protocol (XMPP)
4. Advanced Message Queue Protocol (AMQP)
5. Hyper Text Transfer Protocol (HTTP)

HTTP and CoAP protocols being synchronous protocols require regular polling and posting of updates in the form of redundancy and it imposes considerable strain on already constrained IoT devices. HTTP is based on request and response architecture with lots of overhead. XMPP, AMQP and MQTT are message oriented protocols which allows devices to communicate through an asynchronous, event driven approach according to their own context. XMPP requires processing and storing of extended markup language data (XML) which in turn requires more resources for large number of constrained IoT devices. AMQP is more suitable for applications like server to server communication than device to device communication.

MQTT is asynchronous protocol based on publish and subscribe architecture. MQTT protocol was released by IBM as a lightweight machine-to-machine (M2M) communication protocol [18]. The MQTT protocol runs on top of the Transmission Control Protocol (TCP) and Internet Protocol (IP) stack. The MQTT was designed to work with constraints like unreliable networks with low bandwidth and constrained devices in IoT which are normally battery operated devices. MQTT protocol is more reliable since it provides a technique called Quality of Service (QoS) which can be used as per the application requirements. The MQTT protocol is dependent on the security features of Transport Layer Security (TLS)/ Secure Sockets Layer (SSL) similar to the HTTP protocol used over the Internet. Typical IoT applications based on MQTT protocol are [10] home automation like gardening, light control, power monitoring and energy monitoring, constrained networks, medical applications, smart homes, mobile software. The Face book messenger uses the MQTT protocol.

HTTP is a type of synchronous request and response protocol which is used for internet applications like browsing of web pages. It is based on Representational State Transfer (REST) architecture. REST architecture uses methods like GET and POST to provide the message system where all actions are performed by HTTP commands. HTTP also works on the top of the Transmission Control Protocol (TCP) and Internet Protocol (IP) stack. It supports a transfer of large data using HTTP message based on Extended Markup Language (XML) format. It is not suitable for the IoT system with constraint devices since it adds lots of overhead while using XML format.

CoAP is based on synchronous request and response protocol. This protocol was designed by the Internet Engineering Task Force (IETF). It is interoperable with HTTP and it can be considered as a light weight version of REST architecture. The CoAP runs over the User Datagram Protocol (UDP) unlike MQTT or HTTP to keep light weight implementation. The UDP-based application layer protocol reduces the bandwidth requirements and it supports multicast and unicast features unlike the Transmission Control Protocol (TCP), which does

not support multicast. CoAP supports four message types. Compared to TCP based MQTT, CoAP may have lower overhead but CoAP gets low on package losses since it does not support TCP retransmission technique. Security is provided by Datagram Transport Layer Security (DTLS) on top of UDP. DTLS is used to secure UDP data but it is not designed for IoT since DTLS handshake needs more packets which may increase the resource requirements of constrained devices. Typical IoT applications of CoAP protocol are smart city development, smart grid and building automation, group communications and transport logistics.

AMQP is commonly used in financial industry like banking. JPMorgan, an American banking and financial services company uses AMQP to send almost 1 billion messages per day. AMQP has an underlying reliability when runs over TCP protocol. It provides an asynchronous publish and subscribe based messaging system. It ensures reliability with message-delivery guarantee. The security in AMQP is handled by using of the TLS/SSL. It is used by many financial institutions for heavier data transmission.

XMPP was also standardized by the IETF. It is designed for near real-time communications like chatting and message exchange and it runs over TCP and provides publish/subscribe and also request/response messaging system. The XMPP Protocol has Transmission Layer Security (TLS)/ Secure Socket Layer (SSL) security built into its core specifications. However, it does not support a QoS option useful to achieve the reliability so this makes protocol impractical for IoT systems or M2M communications. XMPP uses XML messages that create additional overhead due to tags and XML parsing and may require more computations which are difficult for constrained devices since it also increases the power consumption of constrained devices. Typical applications for XMPP protocol are group chat, gaming, system control and voice over IP. In 2014 Google stopped XMPP standard support due to lack of world wide support.

Figure 2 shows common protocols used in IoT system at different layers.

Application Layer	MQTT	HTTP	CoAP	AMQP	XMPP
Network Layer	IPv6 over 6LowPAN				
Adaption Layer	6LowPAN				
Perception Layer	IEEE802.15.4 (LR WPAN)				

Fig.2: Typical Protocols used in the IoT System

5. MQTT PROTOCOL MODEL

MQTT is a machine to machine data protocol used in the application layer of IoT system. It is a light weight protocol based on publish subscribe architecture. This protocol was designed in year 1999 by Andy Stanford (IBM) and Arlen Nipper (Eurotech). There are three versions of MQTT protocol like MQTT V3.1 and MQTT V3.1.1 which are deployed in various IoT applications and third version is been released recently called MQTT 5.0. MQTT is standardized in 2013 from Organization for the Advancement of Structured Information Standard (OASIS) [15]. MQTT protocol works with three main components using publish subscribe architecture [10] as follows:

1. **MQTT Client:** MQTT clients are the device which works as a publisher of information or subscriber of information. MQTT client collects the information from various sensors like temperature sensors and humidity sensor and displays information on devices like smart phones. MQTT client can be any device from a microcontroller board interfaced with sensors up to the server which runs the MQTT library and communicates using any kind of network. Commonly these devices are constrained microcontroller boards with sensors and actuators and lots of constraints like limited power, limited memory, limited computational capability and unreliable networks with limited bandwidth.
2. **MQTT Broker:** MQTT broker is a central point of communication and works like a server between the MQTT client as a publisher and MQTT client as a subscriber. Each MQTT client who works as a publisher of information, publish the information to the MQTT broker. Every information that is published includes metadata called topic. Topic is a routing technique used by the MQTT broker for sharing and routing information to appropriate MQTT client who works as a subscriber. MQTT clients who works as a subscriber receives the information by subscribing to a particular topic. MQTT broker delivers all messages with matching topics to MQTT client working as a subscriber.

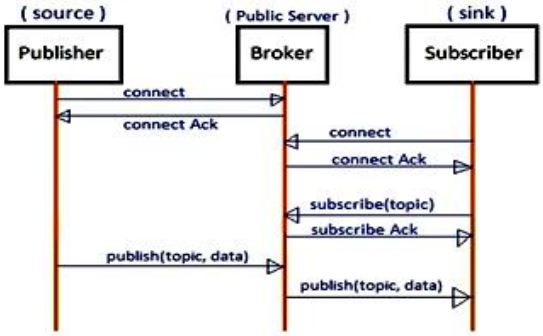

Fig 3: MQTT protocol model

MQTT Broker is responsible for receiving all the information in the form of messages, filtering the information, deciding who is interested in the information and sending information messages to the appropriate subscribers. Broker can also perform authentication and authorization of MQTT clients.

3. **MQTT Connection:** MQTT protocol works on the top of existing Transmission Control Protocol (TCP) and Internet Protocol (IP). It's necessary that MQTT clients like publisher and subscriber as well as MQTT broker have a support for TCP/IP protocol stack. The connection is always between one MQTT clients like a publisher or a subscriber and the MQTT broker such that no two MQTT clients are connected to each other directly. Connection is initiated through the MQTT client sending the CONNECT message or packet to the MQTT broker. MQTT Broker responds with CONNACK message or packet to the MQTT client. Once connection is initialized, MQTT will keep it open as long as MQTT client do not request using DISCONNECT message or packet. There are two standard TCP ports used for sharing information using MQTT protocol. PORT 1883 is used for non encrypted communication without TLS while PORT 8883 is used for encrypted communication with TLS. Figure 3 shows

working of MQTT protocol for sharing information using publish and subscribe model.

The main advantage of publish and subscribe architecture of MQTT protocol is the decoupling of transmitter and the receiver or clients, which offers the following advantages

1. Space decoupling: Publisher and subscriber do not need to know each other.
2. Time decoupling: Publisher and subscriber do not need to run at the same time.
3. Synchronization decoupling: Operations on both components are not halted during publish or receiving.
4. Publish and Subscribe approach also provides a greater scalability than the traditional client server approach. This is because operations on the broker can be highly parallelized and processed event driven.

MQTT protocol commonly uses 14 control packets [12] for communication of information between clients. Packets are as follows:

1. CONNECT – Client requests to connect the server (broker).
2. CONNACK – Connection of client is acknowledged.
3. PUBLISH – A message or information sharing to broker.
4. PUBACK – Quality of Service (QoS1) response to a publish message.
5. PUBREC – First part of QoS2 message flow.
6. PUBREL – second part of QoS2 message flow.
7. PUBCOMP – Last part of QoS2 message flow.
8. SUBSCRIBE – A message used by client to subscribe to specific topic.
9. SUBACK – acknowledgement of subscribe message.
10. UNSUBSCRIBE – A message used by clients to unsubscribe from specific topic.
11. UNSUBACK – Acknowledgement of an unsubscribed message.
12. PINGREQ – Heart beat message to keep connection.
13. PINGRESP – Heart beat message acknowledgement to keep connection.
14. DISCONNECT – Graceful disconnect message sent by Client.

Out of these 14 control packets of MQTT protocol, only four packets are used directly by clients like PUBLISH, SUBSCRIBE, UNSUBSCRIBE and CONNECT and other packets are the part of publish and subscribe mechanism.

MQTT is designed to be a lightweight protocol to support requirements of IoT environment. Being light weight protocol, it does not support much security techniques and mechanisms. CONNECT packet provides a feature of password authentication but this password is shared in a clear text. To provide the adequate security, MQTT protocol depends on other widely accepted security mechanisms at transport layer like Transmission Layer Security to guard against different kinds of security risks and malicious attacks. But security at transport layer does not provide end to end security.

6. MQTT PROTOCOL AND SECURITY

CIA model of security is the most basic and standard model for security. Model refers to three major aspects of a security called Confidentiality (C), Integrity (I) and Availability (A) of information as shown in the figure 4 [16]. AAA model of security is enhanced version of CIA model which refers to extended aspects of security like Authentication (A), Authorization (A) and Accounting (A). Confidentiality refers to ensuring that devices that should have access to the information should able to access the information.

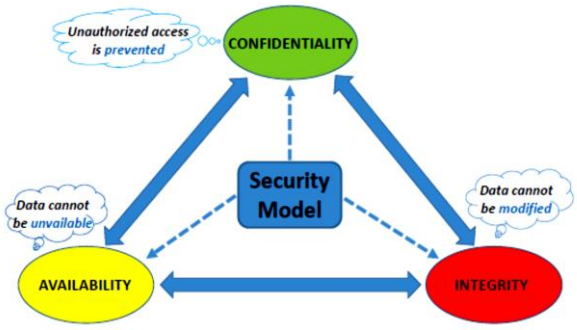

Fig.4: CIA model of Security

Confidentiality is important in IoT systems because many devices may collect and handle the critical information like personal information. Integrity refers to ensuring that information is correct and information is not corrupted or modified in any way by unauthorized access. Availability refers to ensuring that system is available to all valid users all the time. As time will progress [13], billions of devices will be connected to internet and it's unlikely that this horde will be protected by adequate security measures [2] [5]. In the IoT system, it is possible that any time the personal data can become public data due to different types of malicious attacks and data is compromised. To make IoT systems secure, there are some basic security requirements [4] [7] which must be fulfilled. The security requirements are as follows:

1. Confidentiality: it must be ensured that only authorized users can able to access the information.
2. Integrity: it is used to ensure that the completeness and accuracy of information and absence of any unauthorized data manipulation.
3. Availability: it is necessary to confirm that all system services are available when requested by an authorized user any time.
4. Accountability: It makes user responsible for their own actions performed in the system.
5. Auditability: Ability of system to conduct persistent monitoring of all actions.
6. Trustworthiness: It is verification, identification and establishing trust in a third party.
7. Non repudiation: Ability of a system to confirm occurrence or non occurrence of an action.
8. Privacy: It is necessary to ensure the privacy policies that enable individuals to control their personal information.

A thing that obeys all the security requirements is called a secure thing. An attack that threatens at-least one of the security requirements is called security threat. There are various types of threats which make IoT system vulnerable. The main security threats in the application layer [1] [14] are as follows:

1. Data leakage: The attacker can easily steal critical data like password of the user by knowing vulnerabilities of the service or application. Example is snooping attack.
2. Denial of Service (DoS) attack: The attacker can destroy the availability of the application or service. Example is Distributed Denial of Service attack (DDoS).
3. Malicious code Injection: The attacker can upload the malicious codes in software applications exploiting the known vulnerabilities in the system and get access to critical resources.

MQTT protocol has no imposed and robust security mechanism in order to keep the protocol light weight [20]. MQTT protocol supports only basic features of security like username and password but this information is communicated using MQTT

protocol in a clear text. MQTT protocol depends on transport layer for security of data. In MQTT environment the security of communication between MQTT clients and MQTT broker is provided by Secure Socket Layer (SSL) and Transport Layer Security (TLS) protocols. TLS is a protocol that is designed to ensure authentication, confidentiality and integrity at the transport layer. TLS allows the two devices to negotiate a shared key between the two devices on the internet. This shared key is used to create a secure communication channel between the two devices. TLS is a resource intensive protocol and it is challenging for constrained IoT devices to run this protocol. TLS is been designed to function over a reliable communication channel. It means in case of packet loss or message appearing out of order protocol will drop the connection. IoT devices are not only constrained devices but these devices may also run on unreliable channel which may result in lots of connections being dropped. The internet protocol suite TCP/IP used for traditional internet is not suitable directly for IoT system. Different types of IoT access technologies like 6 Low Power Personal Area Network (6LowPAN) are used in IoT applications which runs over Internet Protocol (IPv6) to allow different link layer technologies to communicate [8]. It is a main network layer protocol used in the IoT system which enables IoT devices that cannot handle the traditional IP stack used in the internet. 6 LowPAN do not provide any major security and it depends on other protocols for the security. Figure 5 shows the protocol stack with MQTT.

Fig.5: Protocol Stack with MQTT

Application Layer
MQTT Protocol
TCP/IP Protocol with 6LowPAN
Ethernet/Physical Layer and MAC

TLS is an encryption technique on the transport layer that means the application layer does not have to implement the encryption itself; instead it configures the transport layer to use the encryption protocol. Figure 6 shows the working of TLS for providing the security of information at transport layer. MQTT was designed without any security in the mind and even after 20 years after invention, it lacks many of its own security features. MQTT protocol does not provide and support any encryption by default. MQTT protocol recommends use of TLS/SSL for security. TLS provides an encrypted communication channel over which MQTT messages are sent. Before MQTT channel between the publisher to broker to subscriber is established, TLS uses handshake mechanism to share certificates or keys from the publisher to the broker and also from broker to subscriber.

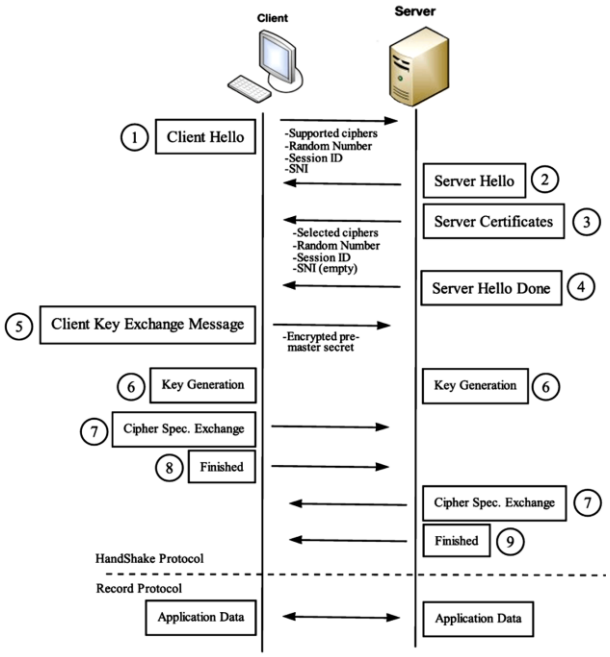

Fig.6: Working of TLS protocol

If certificate or key exchange is successful then TLS creates a secure encrypted communication channel, if not then connection is aborted. It is possible to provide link layer security mechanisms with 6LowPAN protocol but it provides the encryption and integrity check on hop by hop basis. Main concern in this technique is every device on the communication path must be trusted and any traffic leaving 6LowPAN will not be protected and hence the solution is provided by SSL or TLS security. TLS is computationally expensive and requires powerful resources like several kilobytes of primary memory [19]. SSL/TLS do not provide end to end security. Following are the disadvantages of SSL/TLS protocol:

1. IoT traffic needs to be quick and light weight. TLS protocol adds additional two round trips to the start of every session for establishment of secure channel.
2. Certificates can be large files and device memory is limited in IoT system since memory is one of the constraints in IoT system. In TLS handshake, the server can use server side TLS certificate to authenticate itself. Devices often store certificates to authenticate themselves to the application server. Device memory is often limited in IoT and certificates can be large and it is challenging to store the certificates.
3. With TLS we need current time to check certificate validity. NTP client code is required with a Real Time Clock (RTC) preferably with battery to check validity of certificates.

In MQTT environment security of information that is communicated between MQTT broker and MQTT clients is mainly provided by SSL/TLS. This protocol is not sufficient for optimum security at MQTT application layer [13].

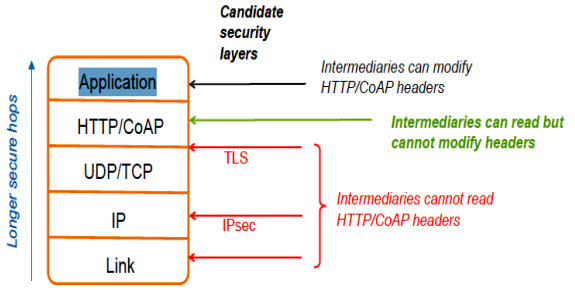

Fig 7: Security at Different Layers

This security does not step in the broker level. A user with the broker access is authorized to access to all information. After connection to the broker, user is listening to the topic and receives all the information and security is compromised. The disadvantage of MQTT security is the use of TLS/SSL with certificates which is not optimized for constrained devices [14]. Using TLS/SSL with certificates and session key management for multiple heterogeneous devices is surely cumbersome and difficult [16]. As discussed in figure 6, SSL/TLS do not provide end to end security in publish subscribe model like MQTT protocol. The solution can be provided by the use of symmetric encryption techniques like Data Encryption Standard (DES) or Advanced Encryption Standard (AES) and asymmetric encryption techniques like RSA. These encryption techniques when in IoT systems have a problem of key management. MQTT protocol was specially designed as a lightweight protocol with minimum focus on the security aspect [11] [17]. Data privacy and confidentiality in MQTT protocol is a major concern since MQTT protocol does not directly provide any support for data encryption [13]. Figure 7 shows security provided by lower layer protocols. Security of MQTT protocol is mainly dependent on lower layer protocol like TCP based on a mechanism called Secure Socket Layer (SSL) or Transport Layer Security (TLS). SSL or TLS protocols are common for applications like internet browsing using personal computers (PC). But, this mechanism is not suitable for constrained devices used in the design of IoT system like use of an 8 bit microcontroller. TLS mechanism does not provide end to end security in MQTT protocol. Different types of data encryption techniques can be used with MQTT protocol to provide an end to end encryption of data that is electronically communicated between MQTT clients [11]. End to end encryption may avoid the dependency on TLS and the associated problems due to various constraints. Data encryption techniques can be classified into two basic types called symmetric encryption and asymmetric encryption. Symmetric encryption technique uses only private key between the two MQTT clients for encryption and decryption of data. Asymmetric encryption technique uses private key along with public key between the two MQTT clients for encryption and decryption of data. Asymmetric encryption technique is computationally more resource incentive than symmetric encryption and hence may not be suitable for constrained devices used in the IoT system. Symmetric encryption techniques can be further classified into two types called block ciphers and stream ciphers. In block ciphers, plain text or data is divided into the larger blocks of fixed size and each block is encrypted separately into a cipher text. In Stream cipher, encryption is performed on the plain text bit by bit to generate the cipher text. After literature survey, we focused our research on three types of block cipher algorithms called Data Encryption Standard (DES), Advanced Encryption Standard (AES) and Blowfish. AES is most commonly used block cipher and Blowfish is more suitable for constrained devices. There are several code blocks available with block ciphers. This research has performed experimentation to measure the performance of MQTT protocol using standard symmetric block encryption techniques like DES, AES and Blowfish and by varying the code blocks for the encryption. In Electronic Code Book (ECB) mode, data is divided into the fixed size blocks and each block is encrypted independently. ECB is like a raw cipher in which for each block of the input we get the corresponding encrypted output. ECB may not be the most preferred mode for the encryption of data since same data pattern in the block may have similar encryption patterns. It is less secure since pattern becomes predictable and becomes vulnerable to the attacks. In Cipher Block Chain (CBC) mode, each block of plaintext is EX-ORed (XOR operation) with the previous block of plaintext and the result of XOR operation is finally encrypted into a cipher text. CBC has an initialization vector EX-ORed with first block of plain text. It is observed that CBC is one of the common modes used for encryption but the process of encryption looks serial and it may require more time. Propagating Cipher Block Chaining (PCBC) mode is an extension of CBC mode in which each block of plain text is EX-ORed with previous block of plaintext and the cipher text and result of XOR operation is finally encrypted into cipher text. It is observed that this technique is not so common in the applications. Cipher Feedback mode (CFB) is a derivative of CBC mode which may work on block cipher as a stream cipher. Output Feedback Mode (OFB) uses key stream block to XOR with the plain text and perform encryption and to get cipher text.

7. PROPOSED SECURITY ARCHITECTURE FOR MQTT

The Proposed Security architecture framework for MQTT protocol shown in the figure 8. This is a de-centralized architecture in which security aspects are separated from MQTT clients and MQTT broker. Key Management can be efficiently performed using a separate dedicated unit called Security Center (SC). All the MQTT clients before establishing connection with MQTT broker will request SC for the appropriate keys. SC will authenticate and authorize the MQTT clients and may generate and distribute the keys to the MQTT clients. There are following phases for publishing data using MQTT client via broker

1. Generation of request for the key.
2. Authentication of the device.
3. Authorization of the device.
4. Generation of the key.
5. Sharing of the key with the MQTT client.
6. Requesting connection with MQTT broker.
7. Authentication of MQTT client by broker (optional).
8. Encryption of data and sharing of data with the broker using pre shared key.
9. Request for connection termination

This decentralized architecture reduces the burden of all security related mechanisms from the constrained IoT devices. This approach does not depend on broker for providing the security and achieves end to end security. This is a distributed architecture such that dependency and overload for generating and managing of security keys with individual devices is shifted to the central device like SC. MQTT clients called publisher use the keys for encrypting the data and sharing the data after connection is established. with the broker. MQTT broker can verify the identity of the MQTT client called publisher from the SC before establishing the connection with MQTT client.

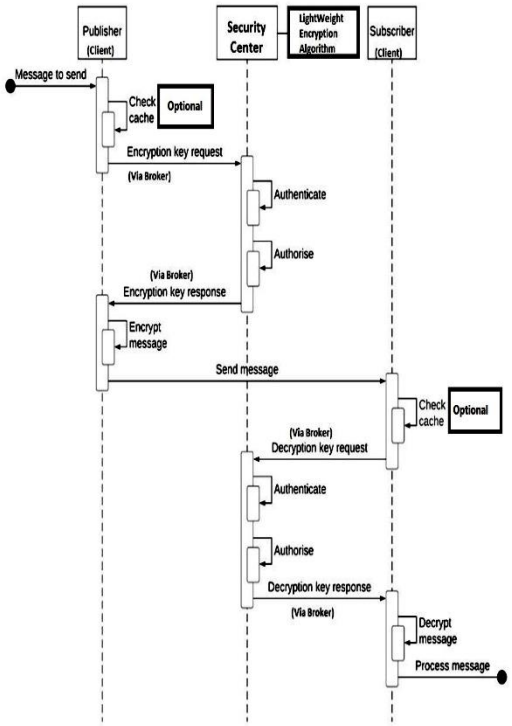

Fig. 8: Proposed Security Architecture

MQTT clients called subscriber also use the keys for decrypting the data. MQTT broker can verify the identity of the MQTT client called subscriber from the SC before establishing the connection with MQTT client. Since generation and distribution of keys are separately managed at SC, it is possible to select the appropriate encryption technique depending on the specifications of applications and nature of the data. For light weight MQTT clients, encryption techniques like Blowfish can be used while for moderate MQTT client's encryption techniques like AES can be used. It is also possible to configure the different types of cipher blocks at SC depending on the need of an application. Depending on the application and availability of resources like memory, it is possible to reduce the number of request from MQTT clients for procuring the keys from SC. Keys may be stored locally at MQTT clients using a facility called cache memory. This security architecture is a dynamic architecture such that according to the applications and exact specifications, security facilities and techniques can be provided. It is observed that the implementation overhead of the proposed architecture is relatively low, particularly in compared with implementation using TLS. It is seen that in the resource-constrained environment and devices of IoT system, TLS is not suitable and it requires extra bandwidth and computations. IoT system can be configured without TLS with this proposed architecture by providing different levels of security. The proposed architecture is dynamic security architecture and designed specifically for IoT environment. The SC handles all of the computations and validations which allow light weight IoT devices to work with minimum knowledge of the security infrastructure and practices used in the system which also adds a hidden layer of security.

8. PERFORMANCE OF SECURITY ARCHITECTURE

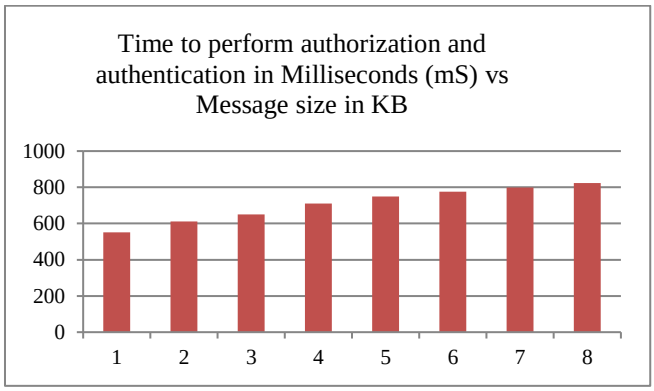

Fig. 9: Response of Security Architecture

Figure 9 shows response of proposed security architecture by varying the size of the message block and time to perform authorization and authentication at SC. Publisher or subscriber request for a key and duration of time is measured from request to response at SC. It is observed that comparatively small time is required for authorization and authentication by providing these facilities using central security center. This time progressively increases with increase in the size of the message. Further experimentation is possible to measure round trip time including network delay. Further experimentation is also possible to measure time by changing the configuration of security centre.

9. CONCLUSION

IoT is a pervasive technology and today it is used in wide range of applications. IoT based applications use devices like sensors with lots of constraints and with the use of low end eight bit microcontroller, limited memory, limited computational capability, limited power consumption, unreliable network and limited bandwidth. Traditional internet protocols like HTTP are heavy weight protocols for IoT based applications due to these various constraints. Specialized light weight protocols like MQTT are used for use in IoT systems with such constrained devices. MQTT is one of the most widely used application layer protocol in the IoT systems and can be considered as a de facto standard for IoT system. IoT system may use a massive number of devices for collecting and sharing the information. This information can be sensitive information like personal health information. Security of such information is very important for adoption of IoT based application. MQTT being a light weight protocol do not have robust security mechanisms. Different types of vulnerabilities and malicious threats have been identified in IoT systems due to weak security. All such threats and attacks must be defended with a smart security mechanism. The security for MQTT protocol is mainly provided by lower layer protocols like TCP with techniques like SSL or TLS. SSL or TLS is an expensive protocol in terms of resources for use in constrained devices in IoT system running on MQTT protocol. It is necessary to provide the efficient light weight and robust security architecture for MQTT protocol to reduce the dependency on heavy weight SSL or TLS security protocol. Security architecture also handles challenges like key distribution, authorization and authentication.

10. ACKNOWLEDGMENTS

We are thankful to Veermata Jijabai Technological Institute (V J T I) Mumbai and Vidyalankar Institute of Technology (VIT), Mumbai for providing the required facilities and support for doing our research.

11. REFERENCES

[1] Jie Li, Wei Yu, Nan Zhang, Xinyu Yang, Hanlin Zhang and Wei Zhao, "A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications", IEE Internet of Things Journal 2016.

[2] Sye Loong Keoh, Sandeep S. Kumar, and Hannes Tscho, "Securing the Internet of Things: A Standardization Perspective", IEEE Internet of Things Journal, Vol. 1, No. 3, June 2014.

[3] Jatinder Singh, Thomas Pasquier, Jean Bacon, Hajoon Ko, and David Eyers, "Twenty Security Considerations for Cloud Supported Internet of Things", IEEE Internet of Things Journal, Vol. 3, No. 3, June 2016.

[4] Arsalan Mohsen Nia, Student Member, IEEE and Niraj K. Jha, Fellow, IEEE, "A Comprehensive Study of Security of Internet- of-Things", IEEE Transactions on Emerging Topics in Computing.

[5] Yixian Yang, Haipeng Peng, Lixiang Li and Xinxin Niu, "General Theory of Security and a Study Case in Internet of Things", IEEE Internet of Things Journal 2015.

[6] https://www.gartner.com/imagesrv/books/iot/ iotEbook_digita .pdf

[7] S. Sicari, A.Rizzardi, L.A.Grieco and A.Coen-Porisini, "Security,privacy and trust in Internet of Things: The road ahead", Elsevier journal on Computer Networks 2014.

[8] Jorge Granjal, Edmundo Monteiro and Jorge Sa'silva, "Security of Internet of Things: A survey of existing protocols and open research issues", IEEE communications surveys and tutorials, 2015.

[9] Aimaschana Niruntasukrat, Chavee Issariyapat, Panita Pongpaibool, Koonlachat Meesublak, Pramrudee Aiumsupucgul and Anun Panya "Authorization Mechanism for MQTT based IoT", IEEE 2016 workshop on IoT.

[10] Satya Sankar Sahoo, "Getting Started With MQTT A Practical Guide".

[11] https://www.google.co.in/search?q=impact+of+iot ++ forec ast+2020 &source=lnm

[12] Journal of Physics: Conference series "IoT real time data acquisition using MQTT Protocol", R.A. Atmoko et el. 2017 J physics: conf ser 853012003

[13] https://iot-analytics.com/iot-market-data/global-iot-enterprise-spending/

[14] Abdessamad Mektoubi et al., "New Approach for a Securing Communication Over MQTT Protocol, A comparison between RSA and Elliptic Curve", IEEE 2016.

[15] Mario FRUSTACI et al., "Evaluating critical security issues of the IoT world: Present and Future challenges", IEEE Internet of Things DOI 10.1109/JIOT.2017.2767291.

[16] [16] Tara Salman and Raj Jain, "A Survey of Protocols and Standards for Internet of Things", Department of Computer Science and Engineering, Washington University,

[17] Sejal Gupta et. al., "Energy-efficient dynamic Homomorphic Security scheme for fog computing in IoT networks", Journal of Information Security and Applications 2021.

[18] https://mqtt.org/

[19] A. R. Alkhafajee et. al., "Security and Performance Analysis of MQTT Protocol with TLS in IoT Networks", IEEE 2021.

[20] Lukas Malina et. al., "A Secure Publish/Subscribe Protocol for Internet of Things", ACM 2019.