Journal of Engineering Sciences
Vol 11, Issue 4 ,April/2020
ISSN NO:0377-9254

SECURE IOT DEVICES USING AES ENCRYPTION

K. Siva Kumar Swamy[1],G.Sony[2], Ch.Jagadeesh Ram[3],B.Navven[4],J.Harshitha[5]
[1] Professor, Department of ECE, Bapatla Engineering College,oxygenswamy@gmail.com
[2]Scholar, Department of ECE, Bapatla Engineering College,sonysekhar2901@gmail.com
[3]Scholar, Department of ECE, Bapatla Engineering College,jagadeesh841@gmail.com
[4]Scholar, Department of ECE, Bapatla Engineering College,naveenbanka666@gmail.com
[5]Scholar, Department of ECE, Bapatla Engineering College, harshithavardhan0001 @gmail.com

***Abstract*— The IOT is a system of connecting devices to the internet .It interacts with the real world with wide range of applications. But, IOT has a disadvantage that is it has no security .To provide security for protecting information to be delivered and communication through the use of codes we use cryptography. In cryptography we are using AES algorithm, in this message will be passed in turns of block ciphers. To connect IOT and cryptography we require MQTT (message queuing telemetry transport) broker to publish and subscribe system takes place, in which we can publish and receive message as a client. These encrypted message cannot be decoded until it has decryption key, so the device can secure from attacks.**

***Keywords*— IoT, MQTT Broker, AES Encryption and Decryption**

1. INTRODUCTION

The Internet of things (IoT) is the system of interconnection of processing objects with the Internet network using existing technologies and communication protocols without requiring human to human or human to computer interaction. The use of IoT is growing rapidly which raises many new serious issues related to security. Many recent reports on cyber security have highlighted IoT vulnerability and the risk with the deployment of intelligent networks. The objects connected to the network can create a lot of attacks and thus can present a danger for the integrity of data. To avoid these possible attacks, security approaches must take place to ensure a set of criteria, like resistance to attack, data authenticity, user privacy, and access control.

As IoT is having a very limited capacity of memory, bandwidth, and energy a new type of protocols and ideas are developed to improve the quality of service for this type of network. One of the protocols is MQTT (Message queue telemetry transport protocol). This protocol is created by IBM which uses publish and subscribe pattern. This protocol requires a very small bandwidth.

This work proposes a new approach for secure communication which is based on MQTT protocol. By using standard techniques of cryptography such as digital signature & payload encryption to secure communications, in an IoT network.

The properties for which the network will be secured are confidentiality, integrity, and authenticity. Confidentiality is the ability to restrict the attackers from identifying the initial plain text message that has been transmitted by the sender. Integrity is the ability to restrict an active attacker from changing the message without the notice of the legitimate user. Authenticity is the assurance that a message, communication of the information via source, it entitles from authenticity which encompasses evidence of uniqueness.

MQTT[4] is used in IoT as a client-server publish/subscribe messaging transport protocol MQTT is lightweight, simple, free and can be implemented easily. MQTT is a binary protocol. MQTT has a minimal packet overhead. One more important thing is MQTT can be easily implemented on the client-side. This most useful suits for constrained devices like IoT which are having bounded resources.

Cryptography is an important feature of computer security. It is dependent on the privacy of the secret or private key. The user accepts an easily remembered passcode that is used to encrypt the cryptographic key and this key is then stored in a database. The security of the cryptographic key is weak due to the practical problems of memorizing pass codes. Since the pass code is not directly matched to a user the system is unable to differentiate between the legitimate user and the attacker.

Advanced Encryption Standard (AES)[1],[2] is the most famous and secure symmetric system which is meant to replace DES for commercial applications and it is a term for the encryption of data.

AES is the first publicly available and open cipher established by the National Security Agency (NSA) for the top security information. In Comparison to AES, DES is insecure due to the small key. Sometimes the algorithm is called Rijndael, which is combined by the names of the two Belgian cryptographers, Joan Daemen and Vincent Rijmen. The basic structure of AES is a substitution-permutation network, which can work fast on both software and hardware. The cipher takes the plaintext block size of 128 bits. The key sizes can be 128, 192 or256 bits. AES operates on a 4x4 square matrix of bytes. This block named into the State array and the AES cipher consists of several repetitions of transformation rounds, where the number of rounds depends on the key length.

Using these algorithm we generate a encoded message while sending a message/data to the receiver IoT device and decrypted the message present in the receiver device and check with the device, if the message is received is correct then the message/data get executed in the device. By this method we can secure the IoT devices from attacks[7].

2. Implemenatation:

In this work we taken ESP-32 board as IoT device and AES alogrtihm for encryption.

2.1 Deployment Architecture:

The scheme outlined has been deployed as a start to finish IoT bundle, encompassing every single important segment to be effortlessly coordinated into an IoT-over-MQTT application .This scheme has been disconnected to library structure, to give secure transmission functionalities to the different application components. The vital MQTT setup has been outlined and recognized, utilizing Eclipse Mosquitto as a message broker. The Key Management Service (KMS) is deployed as a self-governing utility offering types of assistance to the system components. A C library has been created to give the capacities required for the asset compelled IoT gadgets. At last, utilities have been given, as Ardunio module, for asset unconstrained components taking an interest in the IoT system.

a) IoT Devices

We used ESP32 board [3] for this implementation through the Arduino Idle, and also generated using Python(3.7) Idle by adding some cryptographic libraries. This device sends the information to MQTT broker through Wi-Fi which was in build on chip.

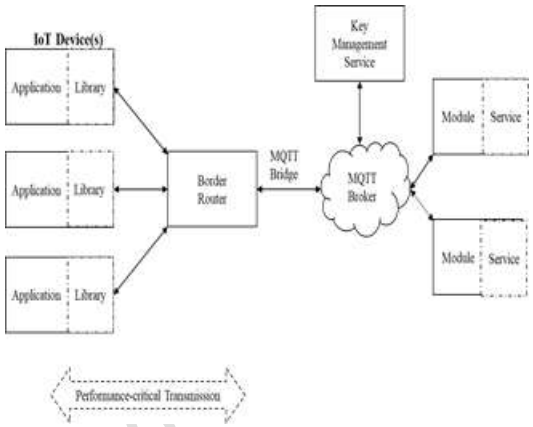

Fig 1.Deployment Architecture

b) MQTT Broker

Mosquitto is a lightweight, open-source usage of an MQTT broker[5] that is appropriate for use on a wide range of devices. It requires almost no arrangement, simply tuning in on port 1883 for any MQTT parcels to be dealt with in like manner. Subjects don't have to be recently designed on the broker to be distributed or bought in to. Or maybe, if the broker encounters a request for a formerly unencountered theme, it will arrangement resources for it and course any consequent messages for them suitably.

C) MQTT Bridge

The broker deployment, as seen in Fig.1, is implemented using a border router in the IoT device environment, forwarding MQTT packets to Local Server(We can use the any cloud platform). An MQTT bridge is configured between the two brokers. This essentially allows one of the brokers to act as a client to the other, publishing and subscribing to all relevant topics. Some simple configuration is done on one of the brokers' mosquitto. conf files, identifying the address and port of the remote broker, along with any topic specifications or remapping necessary. This essentially implements MQTT packet forwarding.

2.2 Encryption and Decryption process

The sender message get encrypted using AES algorithm (ESP32) .It was send through MQTT broker and MQTT bridge to Receiver. The receiver has decrypted key, hence from the decryption code the message get decrypted.

The figure 2 depicts the steps undertaken when an application requires a new message to be published. Functionality is provided for the device to be both a publisher and subscriber to topics. For the purpose of description, the encryption and publishing process will be outlined for the IoT device, and message receipt from subscription and decryption process will be outlined for the resource-unconstrained service. This is based on the conventional perspective of IoT devices such as sensors publishing information about their environment which is then utilized by services to make sense of the data provided. It is also noted that systems operate bi-directionally, and services can publish messages in order to actuate some function on the IoT device.

Message to Send: The aim of most IoT systems is for the application to lie in idle state until some event (a timer, change in environment, received message etc) triggers a response. The response usually requires some form of communication. When this arises, the application will call a function from the library to send the message.

Check Key: The first step in the process is to verify whether the encryption key currently in use for the topic is valid. The first check is on whether the key exists. When a device starts initially or restarts, it will not have any keys stored in the key cache. If one does exist, the expiration time is checked to make sure it has not expired yet. Finally, it is checked to make sure the key itself is valid, in that it is the correct length and format for the encryption algorithm.

Request Encryption Key: In the case where one of the key checks fails, a new encryption key must be requested from the KMS. The request is composed of the client's credentials, the topic, and the QoS level [8]. The same security header format is used, except request that the key ID field is replaced by the client ID. The other fields are encrypted with the private key and stored in the payload. This request is sent to the encryption key request topic, and the client waits for a response from the KMS.

Response Received: The client should always receive a response from the KMS, whether the request is authorized or not. If no response is received after a certain amount of time (network dependent), the request times out and the client sends another request. If the request has not been approved, a response is received with an error code corresponding to the reason for denial.

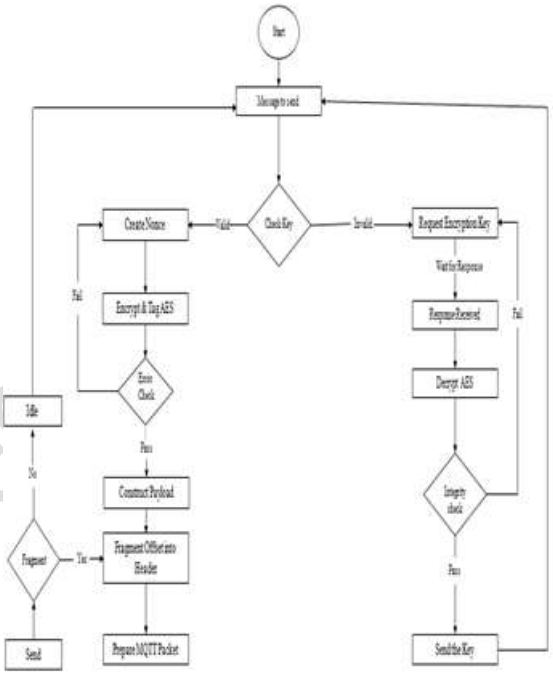

Fig.2: Encryption and decryption Process

Decrypt AES-CCM: Upon receipt of the response for an approved request, the message must be decrypted using the private key. This is achieved, as with all communications within the system, with AES-CCM encryption. This process should reveal the response, containing the key itself, its ID and time (milliseconds) until expiry.

Integrity Check: The first eight bytes of the decrypted message contains the MAC, which is used to make sure that the message is authentic and has not been tampered with in transmission. The plaintext message MAC is calculated and compared with the MAC received, and if they are equal then the message is accepted. If this check reveals that the message has been tampered with, a new key is requested and the receiver (original) message is destroyed.

3. RESULTS AND DISCUSSION

In this work, we have generated a chipper text by AES encryption using Ardunio[6] module using ESP-32 Board and also decrypted by using AES algorithm.

This can be secure our data while there was cyber attacks. The output was shown in serial monitor of ardunio module in fig.3

4. CONCLUSION

In this work we generated encrypted messages using cryptographic algorithms (AES, SHA-1) for different types of boards (ESP32, ESP8266, Raspberry pi). These codes can be linked to the normal IoT device codes which provide a secure connection for both publisher and subscriber.

After having done work, we concluded that MQTT was the best option to have as the IoT protocol. The other goal to keep the

Fig.3 Output in Ardunio Module

Light weight communication which is achieved with the MQTT protocol itself. The results show that TLS although being a good security option implies the exchange of much more bytes that in the case when payload encryption is used.

In this work, AES encryption are proposed to be a good solution. This is the case where the number of bytes exchanged in the communication link is more similar to the case where standard MQTT, i.e. MQTT over TCP, is used.

We consider thus that the case where AES is used as MQTT payload encryption find's the right balance between the lightweight aspect and the security aspect in which this solution guarantees data authenticity and confidentiality.

5. REFERENCES

[1] William Stallings "Cryptography and Network Security" Principles and Practice Seventh Edition

[2] "Advanced Encryption Standard" <:https://en.wikipedia.org/wiki/ Advanced _Encryption_ Standard >

[3] "Data sheet of ESP32" <URL:https://www.espressif.com/sites/default/files /documentation/esp32_ datasheet_en.pdf>

[4] "MQTT Protocol Tutorial: Technical description, MQTT security and Mosquitto"<URL:https://www.survivingwithandroid.com/mqtt-protocol-tutorial/>

[5]"Welcome to the home of MQTT.fx"<URL:https://mqttfx.jensd.de/>

[6] "Arduino IDE" URL:https:// en. wikipedia .org/wiki/Arduino_ IDE/>

[7] "5 Common Cyber Attacks in the IoT - Threat Alert on a Grand Scale" <URL:https://www.globalsign.com/en/blog/five-common-cyber-attacks-in-the -iot>

[8] "MQTT Quality of Service". <URL:https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-qualityof-service-levels>