Avrupa Bilim ve Teknoloji Dergisi
Sayı 16, S. 414-426, Ağustos 2019
© Telif hakkı EJOSAT'a aittir
Araştırma Makalesi

www.ejosat.com ISSN:2148-2683

European Journal of Science and Technology
No. 16, pp. 414-426, August 2019
Copyright © 2019 EJOSAT
Research Article

Secure Gateway for the Internet of Things

Cengiz Toğay[1*], Gökhan Mutlu[2], Durmuş Kurtuluş[3], Faik Özgür[4]

[1] Uludag University, Faculty of Engineering, Computer Engineering Department, Bursa, Turkey (ORCID: 0000-0001-5739-1784)
[2] EMKO Electronic, Research Center, Bursa, Turkey (ORCID: 0000-0002-0674-2908)
[3] EMKO Electronic, Research Center, Bursa, Turkey (ORCID: 0000-0002-1154-5300)
[4] Uludag University, Faculty of Engineering, Computer Engineering Department, Bursa, Turkey (ORCID: 0000-0001-5363-5737)

(First received 10 February 2019 and in final form 11 June 2019)

(**DOI:** 10.31590/ejosat.524783)

ATIF/REFERENCE: Toğay, C., Mutlu, G., Kurtuluş, D. & Özgür, F. (2019). Secure Gateway for the Internet of Things. *European Journal of Science and Technology*, (16), 414-426.

Abstract

Internet of Things (IoT) includes connected devices such as industrial embedded devices, vehicles, smart home appliances, sensors, and actuators. Even non-internet-enabled physical devices can be part of the IoT system through gateways. IoT platforms are getting the attraction of the attackers because of the security weakness of the constrained devices. They can use the IoT devices for DDOS attacking or directly attack the device to damage the overall system. Since several communication industry standard protocols such as MQTT, AMQP, and COAP can be utilized in an environment, communication between devices can be provided through a broker. Unencrypted communications can be sniffed; therefore, username and passwords can be stolen or message content can be modified by an attacker. Therefore, secure authentication and encrypted communication are required in order to make the systems secure. TLS based approaches can be utilized to provide encrypted communication. However, constrained devices cannot handle asymmetric encryption algorithms. In this paper, we propose a new approach for IoT gateways with the utilization of a secure element which has storage for keys, true random generator, and AES 128-bit encryption capability. The proposed approach includes authentication and encrypted communication between the gateway and the broker. We also proposed a new method to provide simultaneous encryption and MQTT based communication with the utilization of physical I^2C property of the ARM Cortex-M3. The secure element/chip is utilized in two different embedded devices, namely new developed embedded device (ARM Cortex-M0) and a demo card (ARM Cortex-M3) to test the approach and measure performances. We also investigate message integrity methods through the cryptographic hash (such as MD5, SHA-1, and SHA-2) or cyclic redundancy check (CRC32/64) algorithms.

Keywords: Internet of Things, Authentication, Encryption, Modbus, Embedded Software.

Nesnelerin İnterneti için Güvenli Ağ Geçidi

Öz

Nesnelerin interneti cihazları, endüstriyel gömülü sistemler, araçlar, akıllı ev aygıtları, sensörler ve işleticiler gibi birbirine bağlı cihazlardan meydana gelmektedir. İnternete bağlanma imkanı olmayan cihazlar dahi ağ geçitleri sayesinde bir nesnelerin interneti sisteminin parçası olabilmektedirler. Nesnelerin interneti sistemleri gömülü sistemlerin sahip oldukları donanım sınırları nedeni ile saldırganların hedefi olmaya başladı. Saldırganlar bu cihazları DDOS ataklarından kullanabilmekte veya doğrudan ilgili cihaza yapılan

[1]Corresponding Author: Uludag University, Faculty of Engineering, Computer Engineering Department, Bursa, Turkey, ORCID: 0000-0001-5739-1784, ctogay@uludag.edu.tr

saldırılar ile bağlı oldukları sistemlerde çok ciddi hasarlara neden olabilmektedirler. Bir ortamda birden fazla MQTT, AMQP ve COAP gibi iletişim protokolünün kullanılması nedeni ile cihazlar arasındaki iletişimde aracı olarak bir aracı/broker kullanılabilir. Saldırganlar şifresiz iletişimin bir sonucu olarak kullanıcı adı ve parolası gibi bilgileri ağ üzerinden elde edilebilmekte ya da mesaj içeriklerini değiştirebilmektedirler. Sistemin güvenli hale getirmek için güvenli yetkilendirme ve şifreli iletişimi sağlamamız gerekmektedir. TLS tabanlı yaklaşımlar uygulanabilir. Ancak, kısıtlı gömülü sistemler asimetrik şifreleme yaklaşımlarını uygulamakta güçlük çekilmektedirler. Bu makalemizde nesnelerin internet ağ geçitleri için güvenli anahtar depolama, gerçek rastgele üretici ve 128 bit AES şifreleme/çözme özelliklerine sahip olan bir chipi baz alan bir yaklaşım önerilmektedir. Sunulan çalışma broker ile cihaz arasında kimlik doğrulama ve şifreli iletişim imkanı sunmaktadır. Sunulan çalışmada ayrıca ARM Cortex-M3'ün sahip olduğu fiziksel I^{2}C özelliğini kullanan bir metod ile iletişim ve şifrelemenin aynı zamanda gerçekleştirilmesini sağladık. Bu çalışma için ARM Cortex-M0 işlemcisine sahip yeni bir gömülü system cihazı geliştirildi ayrıca ARM Cortex-M3 işlemcisine sahip bir demo kart kullanılarak önerilen yaklaşım test edildi ve performans değerleri ölçüldü. Ayrıca, mesajların bütünlüğüne yönelik olarak kriptografik hash (MD5, SHA-1 ve SHA-2) ve çevrimsel fazlalık sınaması (CRC32/64) algoritmaları kullanıldı.

Anahtar Kelimeler: Nesnelerin İnterneti, Kimlik doğrulama, Şifreleme, Modbus, Gömülü Sistemler

1. Introduction

Today, the Internet of Things (IoT) devices are used widely, such as in smart agriculture, smart city, smart transportation, smart grid, e-health, and industrial solutions. IoT systems consist of various sensors and actuators. They need to share their status and be controlled by a remote entity such as other device or application. Because the IoT devices can support various physical and communication protocols, middleware approaches can be applied as represented in Figure 1. The broker is responsible for converting among well-known protocols such as the Message Queuing Telemetry Transport (MQTT) (Banks & Gupta, n.d.), Advanced Message Queuing Protocol (AMQP) (ISO/IEC 19464:2014: Advanced Message Queuing Protocol (AMQP) 1.0, 2014), and WebSockets. Data generated by IoT devices can be delivered to various subscribers by the Broker. For example, the consumers collect data and process them to conclude an action, or store them. Applications can also send some commands to IoT devices. Even non-internet-enabled physical devices can be part of the IoT system through IoT gateways as represented in Figure 1. The gateways provide communication between the devices in the network and the cloud services. They can also be responsible for data fusion, protocol conversion and adaptation, authentication, and secure data transportation.

In industry, both data and commands are considered valuable. IoT devices have to be prepared against various attacks such as hijacking the device or intercepting the traffic related to IoT devices. When an attacker modifies data or commands, production processes can be affected, or even some hazardous situations can occur. Numerous IoT devices are deployed without security implementations regarding the list of OWASP's IoT vulnerability list such as weak passwords, poorly implemented encryption, or unencrypted services ("OWASP IoT Vulnerabilities," n.d.)(Choi, Yang, & Kwak, 2018). IoT devices can be separated depending on the restrictions such as memory, computing performance, energy consumptions, and reliability of network connection. Security approaches applied to the IoT devices/systems are directly affected by these limitations. If computation performance is not high enough, we should prefer to use a lightweight security protocol with a small code size (Andy, Rahardjo, & Hanindhito, 2017). Because of the hardware costs and environment, some of the sensors communicate with the servers through IoT gateways.

An IoT gateway is developed that can communicate with the applications through a modified broker adapted from ActiveMQ. The modified broker has a new authentication and key exchange mechanism based on an external chip (ATAES132A ("ATAES132A," n.d.)) chip's features and encrypted communication. Although, the proposed approach is protocol and encryption algorithm independent, our gateway is designed to communicate with the Message Queuing Telemetry Transport (MQTT) protocol (ISO / IEC 20922: 2016) and messages are encrypted with The Advanced Encryption Standard (AES) (FIPS 197: Announcing the ADVANCED ENCRYPTION STANDARD (AES), 2001)(Fathy, Tarrad, Hamed, & Awad, 2012). As represented in Figure 1, there are four components, namely:

- an application responsible for data processing and service providing,
- a broker bridging messages among publishers and subscribers such as IoT gateways and internet-enabled devices and applications,
- an IoT gateway bridging devices and broker, and
- devices as data producers or consumers such as sensors or actuators.

Any text or binary messages can be transferred through the broker. The gateway can communicate with an application through broker utilizing any protocol such as JSON, XML, and Modbus protocols ("Modbus," n.d.). Both publishers and subscribers trust to the broker. They only communicate through a topic or a queue mechanism, and they have no information (which user published or which users can reach the published data) about each other. Since only the broker knows the publishers and subscribers, anonymity is provided in terms of the users and devices.

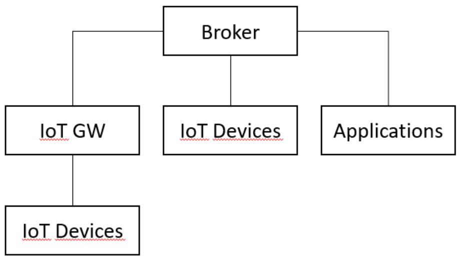

Figure 1. IoT Test Platform

IoT gateways communicate with the broker through TCP over WiFi or Ethernet technologies. Several IoT devices can communicate with the application through an IoT gateway. Advantages of IoT gateway utilization can be listed as below:

- only one user account connects to the broker instead of the several devices' accounts,
- network costs are decreased because of the connection number,
- complex encryption and security approaches can be applied instead of lightweight cryptography,
- IoT devices can save their processing power and memory required for TCP, TLS, and MQTT stacks,
- information fusion can be provided or selected data, such as only changed parameters or some values more than a threshold can be published in terms of all data, and
- all devices can communicate with applications, even not internet capable devices.

The primary focus of this research investigates on cost-effective and secure IoT solutions regarding the storage of pre-shared keys, authentication, session key generation, throughput, and encrypted data transfer with the utilization of the external chip of the Microchip Technology. Both IoT and gateway devices can utilize the proposed approach. We evaluate two embedded devices:

- STM32 Nucleo-144 demo board card includes ARM Cortex-M3 (STM32F207ZGT6) processor (120 MHz), 1 MB Flash, 128KB RAM, and physical I^{2}C and external chip.
- A new industrial IoT gateway includes external chip, ARM Cortex-M0 processor (W7500P) which supports the ARMv6-M Thumb instruction set, Ethernet, Serial communication (RS-232 and RS-485) and I/O interface as represented in Figure 2.

The remainder of this paper is composed of the following sections: background section presents a brief overview of Modbus, MQTT, and AES. Proposed approach section presents a method for encryption and authentication. Analysis and results section includes security analysis and performances of encryption and integrity. Finally, the conclusion section presents a discussion on the results and future works.

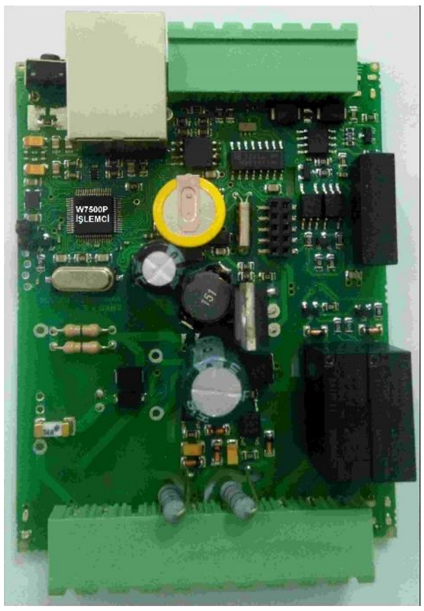

Figure 2. New Industrial IoT Gateway

2. Background

2.1. Modbus Protocol

IoT devices can communicate with the IoT gateway and each other through various technologies such as Z-Wave, ZigBee, WiFi, RFID, Bluetooth, RS-485, RS-232, and Ethernet. Some protocols, such as Modbus, can be applied to devices. In this study, IoT gateway communicates with the devices through the Modbus RTU (binary) protocol ("Modbus," n.d.), one of the most popular peer-to-peer serial communication protocols utilized in industrial control applications and monitoring processes. This protocol is developed by Modicon for use in PLC (programmable logic computers) communication and defines the message structure (slave address, Modbus application protocol data unit (PDU), and an error checking field) and communication rules. We utilized RS-485 as a physical communication channel. One or more master devices (IoT gateway) send a query to slave devices (sensors and actuators), and the slave devices prepare a response depending on the request. In a standard Modbus network, one master device can communicate with up to 247 slave devices. Modbus concentrate on robust communication, and security was not considered during the development phase of the protocol (Urbina et al., 2017). Therefore, various form of attacks can be applied, such as sniffing, identity spoofing, denial of service, and message modification. There are 20 attacks that are classified as interception, interruption, modification, and fabrication for Modbus serial(Huitsing, Chandia, Papa, & Shenoi, 2008). Some of these attacks can cause catastrophic results.

2.2. Message Queuing Telemetry Transport (MQTT)

Message Queuing Telemetry Transport (MQTT) (ISO / IEC 20922: 2016) protocol is standardized by ISO. MQTT is a message exchange protocol designed for Machine-to-Machine (M2M) communications. It is based on a publish/subscribe mechanism. Since it has low bandwidth and memory consumption and supports for various levels of Quality of Service (QoS), MQTT has been preferred in IoT solutions. MQTT messages are exchanged through a server named broker, as presented in Figure 1. IoT devices are connected to the broker, authenticate themselves through username and password, and register to topics for publishing and subscribing. The device only publishes or subscribes to specific/authorized topics/queues. The broker utilizes and maintains an Access Control List (ACL) to control the topics/queues regarding the users of the system (such as IoT devices). When a message arrives at the broker, the message is replicated to authorized subscribers.

Publisher does not know which subscribers its messages are delivered to, and subscribers do not know which publisher published the message. If an attacker can publish a message in a topic, all subscribers assume that the message is published by an authorized user of the system and process it. Weak passwords, unencrypted transportation, lack of a change mechanism for passwords, and weak RSA key lengths such as 1024 bits can cause such a scenario.

Communication security consists of three components, namely confidentiality, integrity, and availability. Other components of security are authentication, authorization, and access control. By default, MQTT has only the authentication with username and password (Andy et al., 2017). Since by default, MQTT does not provide any data encryption, all data can be sniffed during communication. Therefore, all published data such as the value of sensors and password of an account can be obtained, or data can be manipulated during transportation with applications such as Etterfilter or Ettercap ("Ettercap," n.d.). When the value of a sensor is changed during transportation, catastrophic results may occur.

MQTT connection can be encrypted between the device and the broker. The message is encrypted and decrypted with a pre-shared session key on both sides. The session key is generated by a key agreement algorithm. In TLS connections, the key agreement is succeeded with asymmetric encryption (such as RSA or Diffie-Hellman), and then symmetric encryption such as The Advanced Encryption Standard (AES) (FIPS 197: Announcing the ADVANCED ENCRYPTION STANDARD (AES), 2001)(Fathy et al., 2012) is applied. For TLS based encryption, either root certificate or the public key of the server certificate, implementation of cryptographic hash (such as MD5, SHA-1 or SHA-256), asymmetric encryption and symmetric encryption algorithms have to be stored in the IoT devices. There are TLS-enabled MQTT implementations such as PAHO ("Eclipse Paho," n.d.) and Fusesource ("Fusesource MQTT Client," n.d.). TLS based communication can be utilized where root certificates or public key of server certificate can be maintained, processing power and memory requirements are met (Oliveira, Moreira, de Oliveira Silva, Miani, & Rosa, 2018). There are few and widely used brokers such as Mosquitto("Mosquitto," n.d.), ActiveMQ ("ActiveMQ," 2003) and RabbitMQ(Ionescu, 2015). Depending on the search in Shodan, ~25.000 brokers with the default port (1883) are identified (Andy et al., 2017). It should be noted that these MQTT brokers can also operate with TLS support in a different port. The same devices can connect to both ports with the same username and password. Therefore, an attacker can utilize this to obtain credentials through blocking TLS port. It is shown that some IoT devices already connected to the MQTT broker without any encryption (Andy et al., 2017).

Class-0, defined in RFC 7228 (Bormann, Ersue, & Keränen, 2014) IoT devices have limited resources such as computing performance, small ROM size (less than 100 KB), and small RAM size (less than 10 KB) (Andy et al., 2017)(King & Awad, 2016). This kind of device is utilized for collecting and transmitting data to servers for storing and processing. Since their resources are limited, most of the security approaches, such as TLS for transport security, are not feasible (Andy et al., 2017).

As another approach, MQTT message payload can be encrypted with a pre-shared key before publishing in IoT device. The encrypted payload can then be decrypted in either broker or subscriber(s)(Katsikeas, 2016). In the first approach, a custom-developed broker is needed. The broker should decrypt the payload depending on the user-specific pre-shared key. In the second approach, all subscribers have to know the related key to decrypt the payload. The key can be loaded into all devices during production, or it can be set with the registration of the device to the system. One downside of this approach is that when one of the subscribers is exploited, the entire communication of the publisher can be compromised.

2.3. The Advanced Encryption Standard

The Advanced Encryption Standard (AES) adopted by the U.S. government is utilized widely since November 2001 (FIPS 197: Announcing the ADVANCED ENCRYPTION STANDARD (AES), 2001)(Fathy et al., 2012). AES has 128 bits block size and three key sizes, namely 128, 192 and 256 bits. In AES, some rounds (10, 12, and 14 respectively by key sizes) are defined. FIPS 197 (FIPS 197: Announcing the ADVANCED ENCRYPTION STANDARD (AES), 2001) defines the standards of the algorithm and implementations can be verified with test vectors (Bassham, 2002). The algorithm can be implemented by both hardware and software (Wardhani, Ogi, Syahral, & Septono, 2017). The AES algorithm is implemented considering the performance and resources such as memory, processing power, and battery. Some embedded devices have a coprocessor that can perform AES encryption in hardware (Schwabe & Stoffelen, 2016). However, it increases power consumption and the cost of the device. ARM cores are used in the embedded industry. One of the AES software implementations is optimized regarding memory for ARM Cortex-M3 processors (Wardhani et al., 2017). In another approach, ARM instruction sets are utilized to increase the speed of AES (Schwabe & Stoffelen, 2016). Also, some countermeasures regarding side-channel attacks are presented in (Schwabe & Stoffelen, 2016).

A lightweight version of the AES is utilized to encrypt MQTT payloads in ESP8266WiFi (Chowdhury, Istiaque, Mahmud, & Miskat, 2018). The ESP8266WiFi has a 32-bit single core RISC processor clocked at 80 or 160 MHz. AES symmetric key and IV is shared by the MQTT publisher in ESP8266 and a subscriber during the setup of the system. IoT device manufacturers may prefer to leave physical access for maintaining issues. This access can be exploited by an attacker for tampering with firmware, inserting malicious code, or dumping memory to get user account information (Choi et al., 2018). Therefore, the key and IV can be retrieved from the device through console access. MQTT with TLS connection is also implemented on the ESP8266WiFi (Vrettos, Logaras, & Kalligeros, 2018).

We tested one hardware (ATAES132A chip) and two software implementations namely mbedTLS(“MbedTLS,” n.d.) (formerly PolarSSL) and TinyCrypt (“TinyCrypt,” n.d.). All implementations are tested regarding test vectors (Bassham, 2002). TinyCrypt library targets constrained devices and consists of HMAC-PRNG and AES-CTR-PRNG random number generators, AES-128 block cipher (with AES-CBC, AES-CTR, and AES-CMAC encryption modes), AES-CCM authenticated encryptions, ECC-DH key changes, and ECC_DSA digital signatures. Software implementations suffer from the side-channel attacks. Side-channel countermeasures can be implemented but increase overall code size. The TinyCrypt includes certain generic timing-attack countermeasures. The mbedTLS implements symmetric encryption algorithms (Blowfish, Triple-DES (3DES), DES, ARC4, Camellia, XTEA), hash algorithms (MD2, MD4, MD5, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, RIPEMD-160), RSA / PKCS#1, Diffie-Hellman / PKCS#3, and Elliptic Curve Cryptography (ECC). Although pseudo-random implementations are included in the mbedTLS and TinyCrypt, they need to find an entropy source to produce seed.

2.4. ATAES132A microchip

ATAES132A chip (“ATAES132A,” n.d.) is developed by Microchip Technology and presented in Figure 3. This chip can store sixteen 128-bit keys, encrypt/decrypt through AES algorithm standardized by NIST with 128-bit keys, and generate random octets through Internal High-Quality FIPS Random Number Generator (RNG). It supports 10 MHZ SPI (Mode 0 and 3) and 1 MHZ standard I^2C Interface. It can serve in an industrial environment (−40°C to +85°C). ATAES132A chip supports Electronic Codebook (ECB) and the Counter with Cipher Block Chaining-Message Authentication Code (CCM) (Dworkin, n.d.)(Whiting, Housley, & Ferguson, 2003) algorithms. The CCM is a block cipher mode of operation which is used to provide authenticated encryption. AES-CCM mode in the chip provides both confidentiality and integrity with a single key. There are EEPROM Counters (16 high-endurance), and they can count up to 2097134. Counters have a power interruption protection feature to prevent corruption. Each key can be utilized 2097134 times because of the counter limits in CCM mode. When the counter reaches to limit, the related key is disabled. We listed the algorithms tested in this study and their size of memory requirements in Table 1. We did not apply any optimization on source codes of libraries. Therefore, it can be preferred to use where flash memory is a constraint.

Figure 3. ATAES132A chip

Table 1. Source Code and Flash Memory Requirements

Name	Total Flash (Byte)
MQTT	5247
MbedTLS	14052
TinyCrypt	1092
ATAES132A	599
CRC32	1064
CRC64	2099
MD5	2508

2. Material and Method

We propose an approach based on the external (ATAES132A) chip to utilize for authentication and encryption of MQTT payload. As mentioned in previous sections, constrained devices such as (Class-0) can prefer to utilize gateways for publishing their produced data such as sensor values and getting commands from a control application or an administrator. In this study, we concentrate on the communication security between the broker and IoT things such as the IoT gateway, devices, and applications. The processor module in the IoT gateway is responsible for communicating with external chip and connected device (sensors and actuators), task allocation and scheduling, data integration and transmission through the MQTT.

Since the proposed approach requires an adapted broker, we modified the ActiveMQ broker for our implementations. The broker is responsible for the authentication of devices, a key agreement for session keys, encrypt/decrypt messages, and ensure message integrity. Each gateway (device) has sixteen pre-shared AES symmetric key(s) in the external chip. One of the keys named K_1 is utilized during authentication, as presented in Figure 4. K_1 and K_2 are associated with keyId_1 and keyId_2, respectively, in the database of the broker. When the device sets up a TCP connection to the broker, the external chip generates a random 12-octet nonce and a 16-octet Initialization Vector (IV). The nonce is extended with a four-octet counter that includes the number of usage of K_1. Also, an encrypted value (named C_a) is calculated through the encryption of nonce with K_1. The device starts the authentication process by sending the username and password as "nonce:counter:IV:keyId_1:keyId_2: C_a ". The broker authenticates the device with C_a generated by keyid_1 and nonce, as presented in Figure 4. When C_a is verified, both sides have a session key (SK) generated through encryption of the nonce with K_2. When authentication is succeeded, the broker sets a new value of the counter in the database.

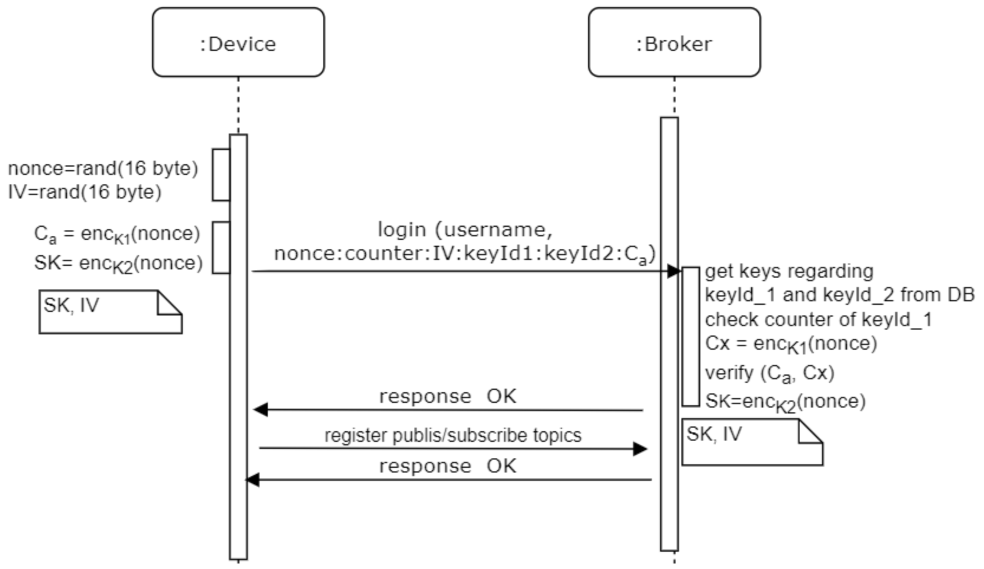

Figure 4. Authentication process

The SK and IV are used to encrypt and decrypt all MQTT payloads during a session as depicted in Figure 5. The client encrypts payload before publishing to the broker, and the payload is decrypted in the broker. After that, using each subscriber's SK and IV, the broker encrypts the payload and sends it to respective subscribers. As represented in Figure 6, N blocks (128 bits) and their digest are encrypted through block cipher, session IV, and session key. Padding can be applied on the plain text to provide 128-bit blocks. When, 128 bits cipher blocks (C_1, C_2, .., C_N, andC_{N+1}) are generated, they are sent to receivers as an MQTT payload.

So far, we concentrate on secrecy and confidentiality issues. Another important part of security is integrity. Cryptographic hash functions are utilized to obtain digest (fingerprint) of the payload. If digest obtained by the receiver without change/modified, it can be used to verify the integrity of the payload. Therefore, it should be transferred via a secure channel. Digest calculator can provide a tag value of the payload. The last block includes the tag value and can be encrypted with another key. In this study, we utilize the same key because of the performance issues. The digest can be generated through a cryptographic hash (such as MD5, SHA-1, and SHA-2) or cyclic redundancy check (CRC) algorithms. MD5 generates directly 128 bits of data. CRC algorithm generates 32 and 64 bits data for CRC32 and CRC64, respectively. The tag value can be extended to 128 bits with zeros or random octets before XOR operation. Therefore, any change/modification on the encrypted payload can be identified. Some well-known approaches, such as HMAC can be applied. Our approach is designed based on parameters such as cost, performance, and hardware issues. Such as some processors have included CRC and cryptographic hash algorithms. The proposed approach can be modified based on the parameters.

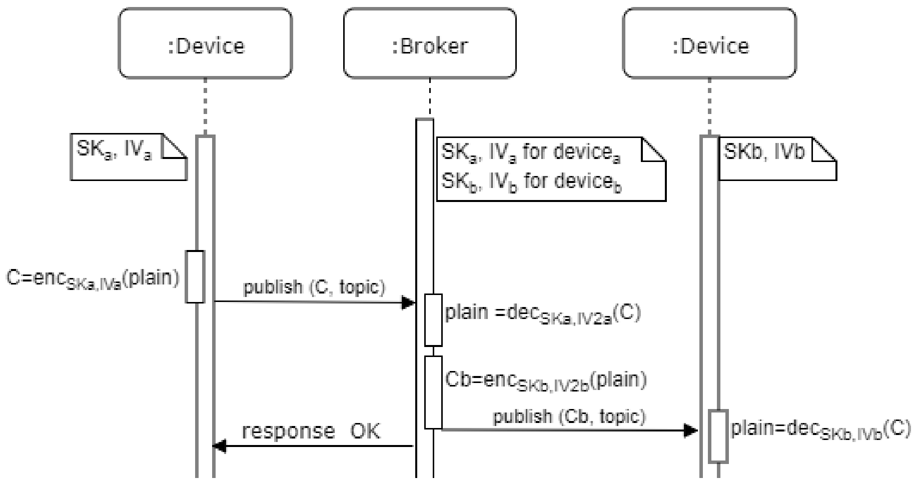

Figure 5. Encrypted communication through custom Broker

The recipient decrypts the payload with its' symmetric session key and IV as depicted in Figure 7. When an encrypted MQTT message which includes blocks (C_1, C_2, .., C_N, andC_{N+1}) are received, each block is decrypted by SK and IV. When all blocks are decrypted, the digest value is recomputed in the recipient side and verified with the last block that consists digest value calculated by the sender.

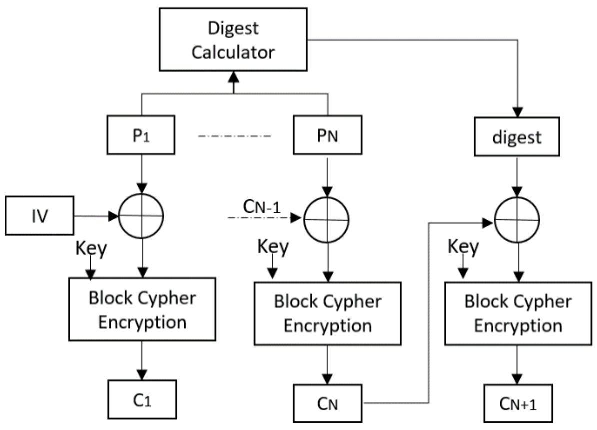

Figure 6. Encryption process

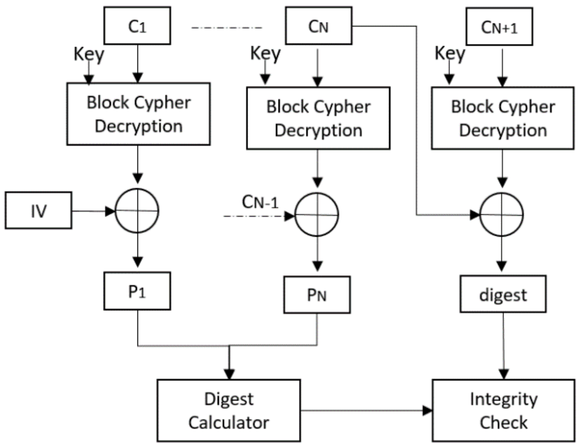

Figure 7. Decryption process

An application can be communicated with the IoT gateway through custom or well-known protocols such as Modbus over MQTT. Therefore, non-internet-enabled physical devices can be monitored and controlled remotely. In our case, the IoT gateway has no capability to control the system. The application in the IoT platform has to know all devices connected to the IoT gateway with their "$slaveId$"s or "$deviceId$"s and register address. The application sends a Modbus request that includes $deviceId$, Modbus address, and size values. This message is delivered to the IoT gateway by the broker, as represented in Figure 8. When the message is received, the IoT gateway saves it to a request buffer. When the gateway is available to process, gets the request from the request buffer and sends to Modbus network. Whenever the requested device gets the message, it prepares a response a Modbus packet and sends to the gateway. When this response arrives, it is also is stored in a buffer. Whenever the gateway is available to publish, gets the response from the buffer, sends it to the application through the broker. As is represented in Figure 8, several Modbus requests can be sent one after another by the application and the IoT gateway process them in the received order. All traffic is encrypted with the session key as defined in Figure 6 and Figure 7.

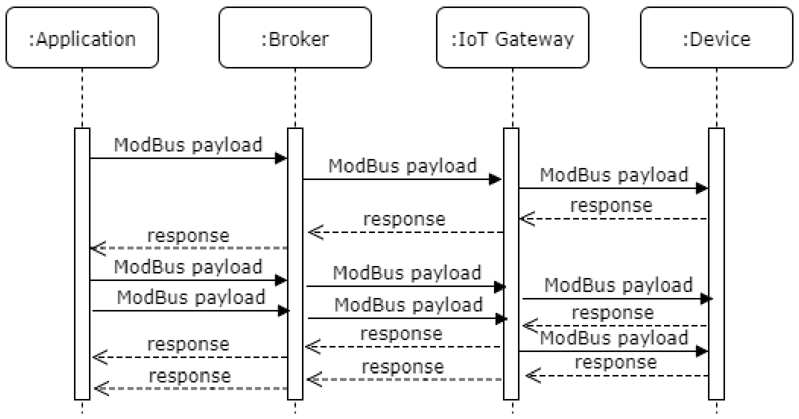

Figure 8. The communication flow between application and device with Modbus over MQTT

3. Results and Discussion

3.1. Security Assessment

IoT security is investigated regarding authentication and encrypted data transfer (Andy et al., 2017)(Chowdhury et al., 2018)(Naik & Maral, 2018). An attacker can sniff the ciphertext and initial parameters such as nonce but must not be able to recover the message. Also, the attacker should not be able to construct a valid ciphertext, a tag, and a nonce.

Username or device identity can be evaluated as a secret as defined in (Naik & Maral, 2018). They can be encrypted during communication. However, a symmetric key for encrypting username should be the same on all devices. When any device is investigated, the key can be compromised. Therefore, username or device ID can be sniffed and decrypted with the key. In this paper, since we query keys depending on device/username in the broker, we are sending username as unencrypted.

Username and password based authentication approaches cause a maintenance problem such as credentials should be changed periodically (Andy et al., 2017). Passwords can be obtained from the devices. Therefore, each device must have a unique password. In our proposed approach, all devices have unique keys. However, even if their keys are different, some replay attacks can be employed. One way is the utilization of the challenge-response mechanism for preventing the replay attacks. However, MQTT has no capability for the challenge-response mechanism. Therefore, we propose a method corresponding with the standard MQTT. The challenge-response mechanism can be applied, but protocol implementations should be modified in both server (broker) and client sides.

MQTT broker listens on port 1883 by default. Any TCP socket application can open a connection to the broker. Therefore, DDOS attacks can be possible for the broker. Brokers can be configured with IP restrictions, and also firewall utilization can be evaluated.

In the proposed approach, the session key is generated through the AES algorithm and random nonce bytes. AES can be utilized as a block cipher-based pseudorandom number generator (PRNG) (Petit, Standaert, Pereira, Malkin, & Yung, 2007). As mentioned before, the seed of the PRNG should be random. Therefore, we utilize a true random generator of the external chip. Some processors also have random generators such as ARM Cortex-M3.

We have implemented an adapted broker for authentication and encrypted data transfer. It should be noted that MQTT or other protocols are not modified. When an attacker sniffs the wire, only username/device Id and a password value consisting of "nonce:counter:IV:keyId_1:keyId_2: C_a" can be obtained. As defined in the proposed approach section, the random nonce value with the counter is encrypted with a pre-shared key associated with $keyId_1$. Broker verifies the encrypted value (C_a) with the utilization of user related key in a user account database. Since nonce and encrypted value is known, the known-plaintext attack can be applied. In that case, only one session related data can be obtained because each session has unique keys.

In order to prevent replay attacks, timestamp, or a counter addition to nonce can be evaluated. In real life, the maintenance of these values can be a problem. For example, counters or the clock of the device can be reinitialized. In that case, synchronization between peers can be broken. Fortunately, a reliable counter mechanism can be utilized with ATAES132A chip. The chip provides high-endurance counters associated with each key. The counters are incremented with each usage (encryption, decryption, etc.) of that particular key. The counter value can then be included in password value. This way, the broker can keep track of previous payloads and reject any payload that has been processed before. The proposed approach is secure in terms of the replay attacks. An attacker can start a replay attack in three ways:

- All packets during a session include authentication and data packets sniffed. Since nonce value includes the old value of the counter, authorization is going to be failed.
- An attacker can start an active attack through stole authentication packets. In that case, it can be authorized and gain the privilege of the user. We assume that keyId_2 related symmetric key is not known by an attacker. Therefore, an attacker cannot encrypt or decrypt messages. When a message is sent by the attacker, the broker will evaluate that message is modified or broken as represented in Figure 7. In that case, the message can be discarded, or an alarm can be generated for administrators.
- During a session, any sniffed packet can be sent to the broker. Since each message and its MAC value encrypted through CBC mode, any change or replay of previous packet cause integrity error as represented in Figure 7.

We should not use the same key and IV for connection. For each connection, two keys are utilized, and their counters are incremented in ATAES312A chip. When a key usage counter reaches a limit, we should use a different key. An attacker can use this method for his advantages. An attacker can prevent connection request of the device to the broker or behave as a broker and return a false error message to the device for providing the device's reconnection. As long as the device continues to try, keys are utilized. As a result, there will be no key to use for connection to the broker. Therefore, the device cannot connect to the broker anymore. As a precaution, if it is possible device waits a limited time and try again. Therefore, all keys are not consumed in a short time.

Software-based AES implementations suffer from the timing attacks, power analysis attacks, and other forms of side-channel attacks (Schwabe & Stoffelen, 2016). The external chip has protections for these kinds of attacks ("ATAES132A," n.d.). If the throughput requirement of the gateway is close to the external chip, the chip can be utilized to encrypt the payload instead of the software implementations. In that case, the CBC mode can be considered.

The tag length can be more than 32 bits, but less than 64 bits is not proposed (Dworkin, n.d.). Since we also encrypt the tag value and payloads are generally smaller than 1024 bytes, 32 byte CRC can be preferred where the processor includes hardware CRC calculator. As represented in the performance section, software implementations of the CRC 64 bits and MD5 have close throughputs.

3.2. Performance of the Encryption Process

For demonstration purpose, we tested two software implementations, namely mbedTLS ("MbedTLS," n.d.) and TinyCrypt ("TinyCrypt," n.d.) in the embedded devices. The embedded device includes ARM Cortex-M0 performs 6272, 28912, and 272742 microseconds for encrypting 1024 bytes through mbedTLS, TinyCrypt, and ATAES132A, respectively as depicted in Figure 9. Based on these values and MQTT publish cost, the throughput of the device is represented in Figure 10. The embedded device includes ARM Cortex-M3 performs 2543, 18670, and 204800 microseconds for encrypting 1024 bytes through mbedTLS, TinyCrypt, and ATAES132A, respectively as depicted in Figure 11. As represented in Figure 9 and Figure 10, ARM Cortex-M3 has about two times better performance than ARM Cortex-M0. It should be noted that M0's (W7500P) price is about 3.49$ and M3's (STM32F207VGT6) price is 13,89$ on a website ("Digikey," n.d.).

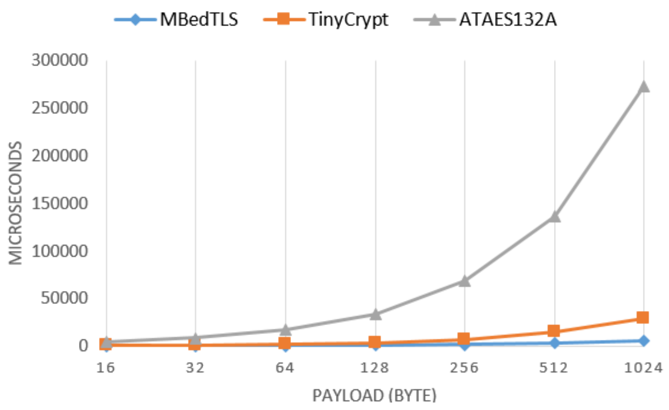

Figure 9. AES 128-Bit encryption with ARM Cortex-M0

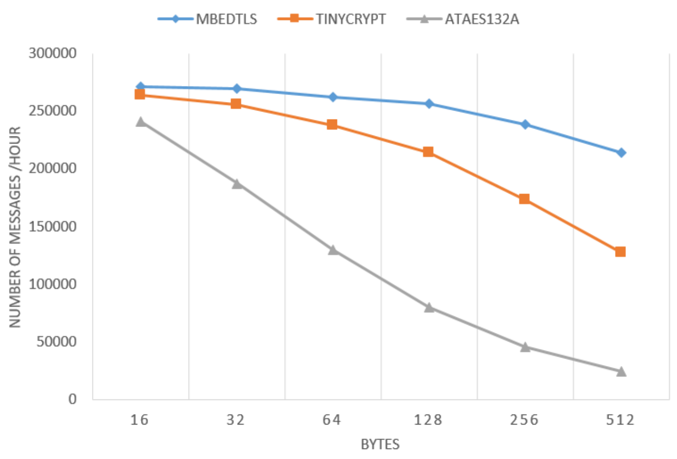

Figure 10. The throughput of encryption approaches for ARM Cortex-M0.

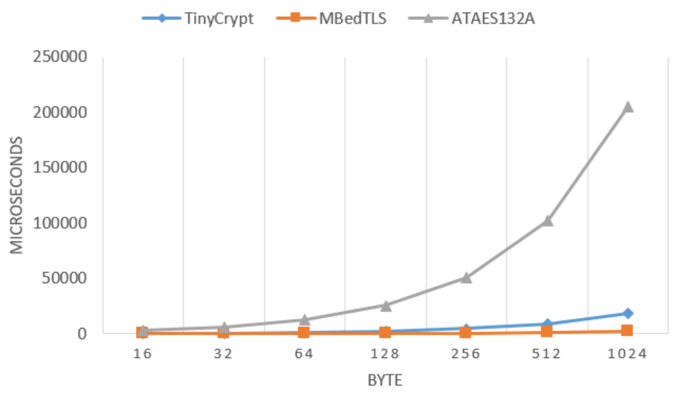

Figure 11. AES 128-Bit in ARM Cortex-M3

Software encryption libraries (mbedTLS and TinyCrypt) consume computation time. Therefore, it is not possible to achieve encryption and to publish simultaneously such as in ARM Cortex-M0. However, since the demo board card includes ARM Cortex-M3 with physical I^{2}C; MQTT communication and AES encryption can be accomplished simultaneously as represented in Figure 12. We used a Tektronix MSO 2014 digital oscilloscope to measure the performance of implementations. While MQTT payload is sending, the external chip can encrypt several new raw data. Encryption operation takes about 3200 microseconds. The external chip can be utilized where MQTT operations take more than 3200 microseconds, effectively. In that case, encryption cost downs to zero. Throughput is affected by block size and network latency. Since, ARM Cortex-M3 enabled gateway can encrypt and publish simultaneously, its throughput high where the network latency is more than 3200 microseconds and block size is small as depicted in Figure 13. Therefore, the external chip-based solution can be selected where the processor has physical I^{2}C, and communication latency is high, or requirement for encryption throughput is low.

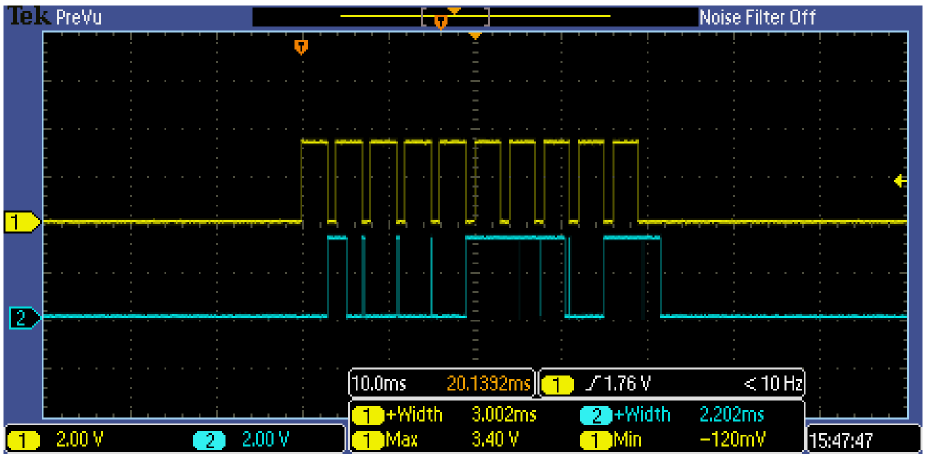

Figure 12. Encryption (channel 1) and MQTT (channel 2)

In the test environment, we evaluated a remote broker. The gateway sends 1024 octet (64 blocks) encrypted payload to a broker in between 3425 and 8104 microseconds for both embedded devices. In our performance evaluation, publish time is assumed as 6400 microseconds. It should be noted that MQTT throughput effected by processor speed, the network latency, and performance of the Broker.

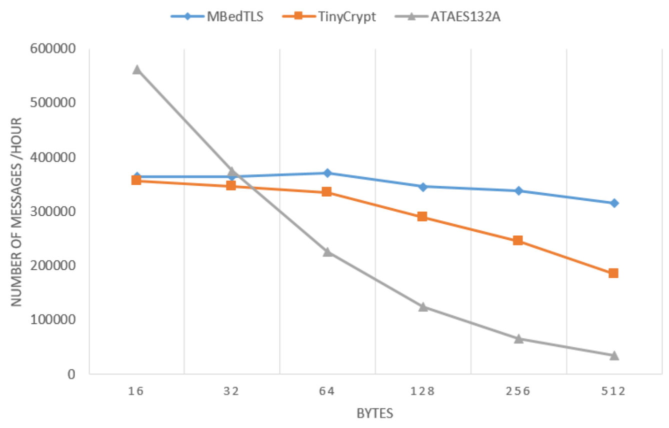

Figure13. The throughput of encryption approaches for ARM Cortex-M3.

3.3. Performance of Message Integrity

The proposed approach can detect the attack after the whole message is decrypted. MD5 is more secure to attacks than CRC 32 and 64 bits. We measure that 256 octet data’s tag value is calculated in 648, 1228, and 1787 microseconds through CRC32 (software), CRC64 (software), and MD5 (software), for ARM Cortex-M0, respectively as represented in Figure 12. The MD5 algorithm can be selected instead of CRC64 in the advantage of security and performance for large message size. Memory consumption for two algorithms is almost the same.

Some processors such as ARM Cortex-M3 has internal CRC32. In that case, CRC32 can be utilized to take advantage of performance and code size. Depending on our test are represented in Figure 14, 256 octet data's tag value is calculated in 15, 74, 126, and 137 microseconds through CRC32 (hardware), CRC32 (software), CRC64, and MD5 (software) for ARM Cortex-M3, respectively as depicted in Figure 13. CRC32 hardware implementation can be selected where throughput is essential. The MD5 algorithm can also be selected in ARM Cortex-M3 instead of CRC64 in the advantage of security and performance for large message size. Some models of the ARM such as STM32F217VG have also AES, MD5, and SHA-1 internal accelerators. Algorithms and implementations should be defined depending on the processor and desired throughput.

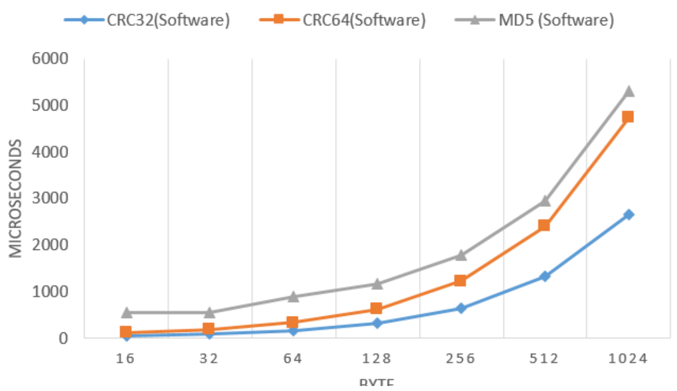

Figure 12. Performance of message authentication algorithms in ARM Cortex-M0

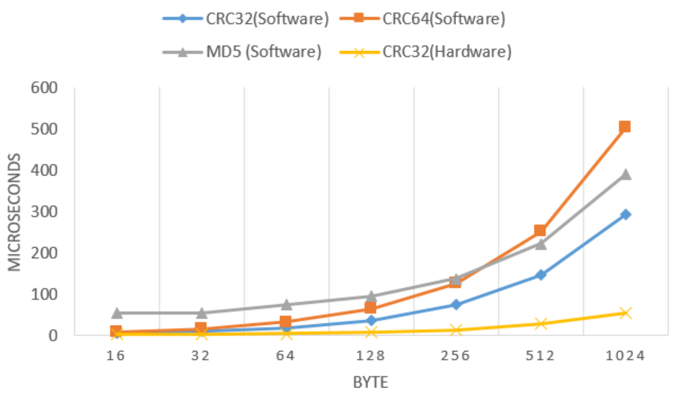

Figure 13. Performance of message authentication algorithms in ARM Cortex-M3

4. Conclusions and Recommendations

This paper is aimed to develop a cost-effective IoT gateway. The gateway sends collected sensor data from various devices with Modbus protocol to a server application for processing through an adapted broker. MQTT is a standardized communication protocol over TCP for IoT devices. By default, data is transported unencrypted. However, TLS based communication can be set up by asymmetric key exchange mechanisms. Both the RSA operation library and RSA public keys should be stored in an embedded device's non-volatile memory. Therefore, the TLS based key exchange is not suitable for constrained devices such as Class-0. In this study, we utilized the pre-shared keys stored in an external (ATAES132A) chip. The keys used for both authentication and encryption and integrity of MQTT message payload between senders such as IoT Gateway and receiver such as the broker.

Through the proposed approach, cost-effective, secure IoT gateways/devices can be developed. The proposed approach offers the utilization of the external chip for two cases 1) authentication and key exchange, 2) authentication and payload encryption. The first method can be preferred where the processor has not physical I^2C. Through the proposed approach, keys are securely stored in the external chip. The session key is generated from a true-random seed. Software implementations such as mbedTLS or hardware cryptographic accelerators such as CRC, MD5, and AES are utilized, and payload integrity is satisfied. Shared keys are stored in the chip's secure area. Therefore, keys cannot be obtained through memory dump or side channel attacks. Chip has a true-random generator, which is one of the requirements for secure communication and FIPS based AES implementation.

In the second method, simultaneous encryption and publish can be succeeded in the utilization of physical I^2C. The only difference from method 1 is external chip can be utilized for encryption and decryption of the payloads, especially processor's hardware cryptographic accelerators are limited, such as ARM Cortex-M3 (STM32F207VGT6). AES encryption algorithm is not located in flash memory or RAM of the device. Therefore, more memory is reserved for the application. Since encryption operations are performed in the chip, more process power is saved for the applications in the IoT gateway. The chip is secure against the side channel attacks. Most of the AES algorithm software implementations suffer from the side channel attacks.

Modbus data can include 245 bytes maximum. Therefore, encryption with mbedTLS can be selected for software implementation in both embedded devices. However, when small queries are sent to the gateway or gateway sends sensor values periodically, the external chip can be evaluated efficiently with physical I^2C of ARM Cortex-M3.

As future work, we will evaluate other software implementations and SPI connection interfaces of ATAES132A for more efficient communication between the processor and the chip. ARM Cortex-M3 (STM32F217VGT6) that include cryptographic accelerators such as AES, MD5, SHA-1 can be utilized.

4. Acknowledge

The author thanks TÜBİTAK (Türkiye Bilimsel ve Teknik Araştırma Kurumu) for its support to this study under TUBITAK 1505 University-Industry Collaboration Grant Program project number 5170033 and EMKO Electronic A.Ş located in Bursa, Turkey.

References

ActiveMQ. (2003). Retrieved May 12, 2017, from http://activemq.apache.org/

Andy, S., Rahardjo, B., & Hanindhito, B. (2017). Attack scenarios and security analysis of MQTT communication protocol in IoT system. In *2017 4th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)* (pp. 1–6). http://doi.org/10.1109/EECSI.2017.8239179

ATAES132A. (n.d.). Retrieved from http://ww1.microchip.com/downloads/en/DeviceDoc/ATAES132A-Data-Sheet-40002023A.pdf

Banks, A., & Gupta, R. (n.d.). MQTT Version 3.1.1. Retrieved from https://www.oasis-open.org/news/announcements/mqtt-version-3-1-1-becomes-an-oasis-standard

Bassham, L. E. (2002). The Advanced Encryption Standard Algorithm Validation Suite (AESAVS). Retrieved from http://csrc.nist.gov/groups/STM/cavp/documents/aes/AESAVS.pdf

Bormann, C., Ersue, M., & Keränen, A. (2014, May). Terminology for Constrained-Node Networks. RFC Editor. http://doi.org/10.17487/RFC7228

Choi, S. K., Yang, C. H., & Kwak, J. (2018). System hardening and security monitoring for IoT devices to mitigate IoT security vulnerabilities and threats. *KSII Transactions on Internet and Information Systems*, *12*(2), 906–918. http://doi.org/10.3837/tiis.2018.02.022

Chowdhury, F. S., Istiaque, A., Mahmud, A., & Miskat, M. (2018). An implementation of a lightweight end-to-end secured communication system for patient monitoring system. In *2018 Emerging Trends in Electronic Devices and Computational Techniques (EDCT)* (pp. 1–5). http://doi.org/10.1109/EDCT.2018.8405076

Digikey. (n.d.). Retrieved December 20, 2018, from https://www.digikey.com

Dworkin, M. (n.d.). NIST Special Publication 800-38C: Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality. Retrieved from https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38c.pdf

Eclipse Paho. (n.d.). Retrieved from https://www.eclipse.org/paho/

Ettercap. (n.d.). Retrieved December 20, 2018, from https://www.ettercap-project.org/

Fathy, A., Tarrad, I. F. I. F., Hamed, H. F. A. H. F. A., & Awad, A. I. A. I. (2012). Advanced Encryption Standard Algorithm: Issues and Implementation Aspects. In *Communications in Computer and Information Science*. http://doi.org/10.1007/978-3-642-35326-0

FIPS 197: Announcing the ADVANCED ENCRYPTION STANDARD (AES). (2001). Retrieved from http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

Fusesource MQTT Client. (n.d.). Retrieved from https://github.com/fusesource/mqtt-client

Huitsing, P., Chandia, R., Papa, M., & Shenoi, S. (2008). Attack taxonomies for the Modbus protocols. *International Journal of Critical Infrastructure Protection*, *1*, 37–44. http://doi.org/10.1016/J.IJCIP.2008.08.003

Ionescu, V. M. (2015). The analysis of the performance of RabbitMQ and ActiveMQ. In *2015 14th RoEduNet International Conference - Networking in Education and Research, RoEduNet NER 2015 - Proceedings* (pp. 132–137). Craiova Romania. http://doi.org/10.1109/RoEduNet.2015.7311982

ISO/IEC 19464:2014: Advanced Message Queuing Protocol (AMQP) 1.0. (2014). Retrieved from http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=64955

Katsikeas, S. (2016). *A lightweight and secure MQTT implementation for Wireless Sensor Nodes*. *Technical University of Crete*. Technical University of Crete.

King, J., & Awad, A. I. (2016). A distributed security mechanism for Resource-Constrained IoT Devices A Distributed Security Mechanism for Resource-Constrained IoT Devices, *40*(June), 133–143.

MbedTLS. (n.d.). Retrieved from https://tls.mbed.org

Modbus. (n.d.). Retrieved November 21, 2018, from http://www.modbus.org

Mosquitto. (n.d.). Retrieved December 19, 2018, from https://mosquitto.org/

Naik, S., & Maral, V. (2018). Cyber security - IoT. *RTEICT 2017 - 2nd IEEE International Conference on Recent Trends in Electronics, Information and Communication Technology, Proceedings*, *2018–Janua*, 764–767. http://doi.org/10.1109/RTEICT.2017.8256700

Oliveira, C. T., Moreira, R., de Oliveira Silva, F., Miani, R. S., & Rosa, P. F. (2018). Improving Security on IoT Applications Based on the FIWARE Platform. In *2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA)* (pp. 686–693). http://doi.org/10.1109/AINA.2018.00104

OWASP IoT Vulnerabilities. (n.d.). Retrieved from https://www.owasp.org/index.php/OWASP_Internet_of_Things_Project#tab=IoT_Vulnerabilities

Petit, C., Standaert, F.-X., Pereira, O., Malkin, T., & Yung, M. (2007). A Block Cipher based PRNG Secure Against Side-Channel Key Recovery. In *AsiaCCS* (pp. 1–22). Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.74.4352%5Cnhttps://eprint.iacr.org/2007/356.pdf

Schwabe, P., & Stoffelen, K. (2016). All the AES You Need on Cortex-M3 and M4. *IACR Cryptology EPrint Archive*, *2016*, 714.

TinyCrypt. (n.d.). Retrieved from https://01.org/tinycrypt

Urbina, M., Astarloa, A., Lázaro, J., Bidarte, U., Villalta, I., & Rodriguez, M. (2017). Cyber-Physical Production System Gateway Based on a Programmable SoC Platform. *IEEE Access*, *5*, 20408–20417. http://doi.org/10.1109/ACCESS.2017.2757048

Vrettos, G., Logaras, E., & Kalligeros, E. (2018). Towards Standardization of MQTT-Alert-based Sensor Networks: Protocol Structures Formalization and Low-End Node Security. In *2018 IEEE 13th International Symposium on Industrial Embedded Systems (SIES)* (pp. 1–4). http://doi.org/10.1109/SIES.2018.8442109

Wardhani, R. W., Ogi, D., Syahral, M., & Septono, P. D. (2017). Fast implementation of AES on Cortex-M3 for security information devices. In *2017 15th International Conference on Quality in Research (QiR) : International Symposium on Electrical and Computer Engineering* (pp. 241–244). http://doi.org/10.1109/QIR.2017.8168489

Whiting, D., Housley, R., & Ferguson, N. (2003). Counter with CBC-MAC (CCM). United States: RFC Editor.