

# Trident

A M2M communication solution for IoT devices using blockchain fused  
MQTT and PUF based authentication scheme

Authored by

Khalida Anika Tabassum, 160041050

Afifa Hossain, 160041063

Md. Hafizur Rahman, 160041075

Supervised by

Md. Hamjajul Ashmafee

Lecturer

Dept. of Computer Science and Engineering

March, 2021



A thesis submitted in partial fulfillment of the requirements for the degree of B.Sc.

Engineering from Department of Computer Science and Engineering,

Islamic University of Technology

# Declaration of Authorship

This is to certify that the work presented in this thesis is the outcome of the analysis and simulations carried out by **Khalida Anika Tabassum**, **Afifa Hossain**, and **Md Hafizur Rahman** under the supervision of **Md Hamjajul Ashmafee**, Lecturer, Department of Computer Science and Engineering (CSE), Islamic University of Technology (IUT), Dhaka, Bangladesh. It is also declared that neither of this thesis nor any part of this thesis has been submitted anywhere else for any degree or diploma. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

*Authors*



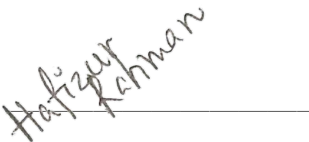
---

Khalida Anika Tabassum, ID:160041050



---

Afifa Hossain, ID:160041063



---

Md. Hafizur Rahman, ID:160041075

*Supervisor*



---

Md. Hamjajul Ashmafee,

Lecturer,

Dept. of Computer Science and Engineering,

Islamic University of Technology (IUT)

# Acknowledgement

At the outset, we express utmost gratitude to Almighty Allah for His blessings which allowed us to shape this research into reality and give it a form. This thesis owes its existence to a lot of people for their support, encouragement, and guidance. We would like to express our gratitude towards them.

We are very grateful to our supervisor **Md. Hamjajul Ashmafee**, lecturer, Department of Computer Science and Engineering, Islamic University of Technology (IUT), for his supervision, knowledge and support, which has been invaluable for us.

Finally, we seize this opportunity to express our profound gratitude to our beloved parents for their love and continuous support both spiritually and mentally.

# Abstract

The Internet of Things (IoT) is a network of interconnected computing devices, mechanical and digital machines that can move data without needing human or computer intervention. In this context **MQTT** was developed. MQTT is a lightweight **publish/subscribe messaging protocol** designed for M2M (machine to machine) telemetry in low bandwidth environments. MQTT stands for MQ Telemetry Transport but previously was known as **Message Queuing Telemetry Transport**.

The main purpose behind developing MQTT was to make it lightweight, so that it can be used in case of resource constrained devices. MQTT doesn't specify security mechanisms and instead, it only ensures delivering of data since it is a message oriented protocol, which is clearly an issue from the security standpoint. Therefore, encryption, authentication, reliability needs to be implemented as separate features in the application layer or for instance via TLS, which on the other hand might increase overhead. Overall, security is an ongoing endeavor for MQTT, and it is most likely the most critical one provided that MQTT is one of the most commonly deployed and mature communication protocol solutions. In contrast to other available options, addressing the problems and establishing a solution architecture will give MQTT a major and significant benefit.

In this paper we provide you **Trident**, a blockchain based structure for secured communication through MQTT with the help of physically unclonable function (PUF).

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Overview . . . . .	2
1.2	Motivation . . . . .	4
1.3	Problem Statement . . . . .	5
1.4	Objective and Organization of this thesis . . . . .	7
<b>2</b>	<b>Background Study</b>	<b>8</b>
2.1	Related Terminologies . . . . .	8
2.1.1	Message Queuing Telemetry Transport (MQTT) [19] . . . . .	8
2.1.2	Blockchain [10] . . . . .	10
2.1.3	Physically Unclonable Function (PUF) . . . . .	13
2.2	Related Works . . . . .	14
<b>3</b>	<b>Proposed Mechanism</b>	<b>16</b>
3.1	Client registration phase by broker . . . . .	17
3.2	Publish/Subscribe phase from client . . . . .	18
3.3	Authentication phase by broker . . . . .	19
3.4	Acknowledgement phase by broker . . . . .	20
3.5	Storing phase by broker . . . . .	20
3.6	Broker to Broker communication . . . . .	21
<b>4</b>	<b>Simulation and Observation</b>	<b>22</b>
4.1	Discussion on our works: . . . . .	23
<b>5</b>	<b>Future Works and Conclusions</b>	<b>24</b>
	<b>References</b>	<b>24</b>

# Chapter 1

## Introduction

### 1.1 Overview

The Internet of Things (IoT) is the most recent technological breakthrough. The Internet of Things (IoT) has the potential to have significant and far-reaching impacts in a variety of industries. It connects billions of everyday devices to the internet, ranging from smart watches to large-scale industrial machinery. The Internet of Things (IoT) is bringing the physical and online worlds together in new ways, presenting new opportunities and challenges for businesses, companies, governments and consumers. Over the last decade the term Internet of Things (IoT) has attracted attention by projecting the vision of a global infrastructure of networked physical objects and by enabling anytime, anyplace connectivity for anything. [7].

The Internet of Things is expected to pervade almost every sector and sphere. Data can be gathered from almost every source imaginable. It is considered as a substantial frontier that has the potential to enhance almost all aspects of our lives. Most devices that haven't been connected to the internet before will now be networked and respond in the same way that smart devices do. By 2020, the world is set to be completely IoT oriented. The Internet of Things can also be considered as a global network which allows the communication between human-to-human, human-to-things and things-to-things, which is anything in the world by providing unique identity to each and every object [1].

One may ask about the benefits which come with this technology. IoT allows efficient utilization of resources. In certain facets of life, it eliminates human activity. Enabling IoT

would lower manufacturing costs while rising returns. It increases the speed and precision of analytical decisions. It increases product promotion in real time. Develops the user service. It ensures high-quality data as well as reliable processing.

However we should bear in mind that the more the devices are connected to the internet, the more the data are passed in between, and the more the chance of being hacked or snitched. So the protocol, which these devices will use to communicate is a vital factor for their security, for our security. Also, another thing to consider while developing the protocol is that most of these devices are very constrained in terms of memory, CPU power etc. Because of the very limited computing performance, most of the resource constrained devices cannot handle most of the security approaches, notably the mechanisms which have heavy computation such as a running TLS for transport security [2].

Considering these circumstances, MQTT was established. At present it is one of the most commonly used protocols in IoT projects [12]. It's a basic messaging protocol that uses publish/subscribe operations to transmit and receive data between clients and servers. Furthermore, the protocol's compact scale, low power consumption, small data packets, and simplicity of execution makes it suitable for "machine-to-machine" or "Internet of Things" implementations. MQTT has unique characteristics, such as:

- Lightweight protocol which makes it easy to implement and provides fast data transmission.
- A messaging technique which is at the core of this protocol.
- Short data packets. Hence, low network usage.
- Low power usage. As a result, it saves the connected device's battery.
- It's real time! This is specifically what makes it perfect for IoT applications.

considering all the conditions, MQTT is one of the best solutions to be used in IoT devices[3].

## 1.2 Motivation

Even though MQTT is widely used for M2M communication in IoT environments, it has scarcity in providing necessary security. The standard itself states that “as a transport protocol, MQTT is concerned only with message transmission and it is the implementer’s responsibility to provide appropriate security features”[19].

As of today, 1st March, 2021, we did a Shodan search to find the number of MQTT servers connected to the internet. The results show that a total of 147,000 brokers are connected worldwide, which proves the popularity of the protocol.



Figure 1.1: MQTT servers online

From figure 1.1 we can see that around 146,313 brokers which is more than 99%, uses port 1883 which is the default port for MQTT that doesn’t use TLS mechanism for security purpose. And that makes it vulnerable to any kind of cyber-attack.



### 1.3 Problem Statement

Since MQTT lacks in the provision of security measures explicitly, TLS/SSL services can be used for this purpose. Unfortunately, due to finite computing performance, most of the resource constrained devices cannot handle most of the security approaches [6], notably heavy computation mechanisms like TLS for transport security. A simulation performed by HiveMQ demonstrates how TLS usage increases the CPU utilization.

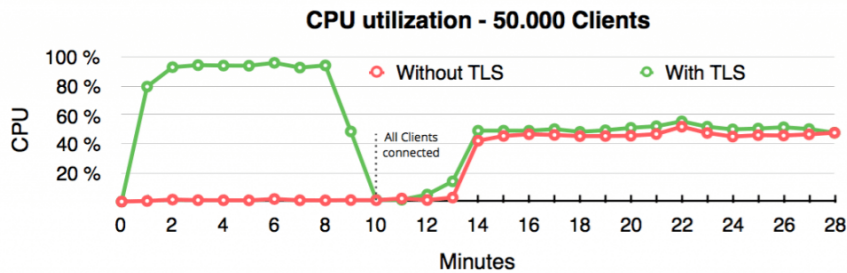


Figure 1.2: CPU utilization

Now, let's take a look at the performance of MQTT in terms of communication security:

- **Authentication:** No authentication criteria is provided. In few cases it uses user-name/password verification, which can't provide necessary security.
- **Data confidentiality:** The power to render a letter incomprehensible to any possible intruder or spy, and hence unavailable. The contents are only available to the message's author and legitimate recipients. Cryptography is essentially used to guarantee secrecy. MQTT, on the other hand, does not have this because it does not use any form of cryptography.
- **Data integrity:** It cannot guarantee whether a message has been altered or not during contact, either inadvertently or deliberately, between the time of sending and the time of receiving.
- **Non-repudiation:** The right to certify that the sender of a message cannot deny transmitting it or claiming not to be the actual sender, and that the receiver of the message cannot deny receiving it. MQTT is also missing in this region.
- **Time stamping:** This applies to the need to include the exact date (usually to the second) on which a message was received. MQTT lacks a timestamp.

Also, MQTT is a centralized system which makes it vulnerable to few points:

- Unable to scale up vertically at a certain point – After a certain point, increasing the server node’s hardware and software resources would not dramatically boost performance.
- When traffic spikes, bottlenecks will occur because the server will only have a small number of available ports on which it can listen for connections from client nodes. As a consequence, when there is a lot of traffic, the server may be exposed to a Denial-of-Service or Distributed Denial-of-Service attack.
- Since there is only one central node, the system is heavily reliant on network communication. If the nodes interrupt connectivity, the system will crash. There will be no incremental weakening of the system; instead, the whole system will collapse suddenly.
- There are fewer opportunities for data backup. When a server node crashes and there is no backup, the data is destroyed automatically. Database management is difficult – There is just one cloud node, and shutting the server down for servicing is inefficient and unprofessional due to capacity issues. As a result, upgrades must be conducted on-the-fly (hot updates), which is challenging and dangerous.

## 1.4 Objective and Organization of this thesis

our primary objective is to provide necessary mechanism to ensure communication security for MQTT by

- Achieving A distributed Broker System
- Enabling light weight authentication of a device
- Adding encryption mechanism
- Providing Confidentiality
- Non-repudiation
- Providing message integrity

To do so, we can take help from two other technologies of the industry, Blockchain and Physically Unclonable Function (PUF). We will introduce them shortly.

The thesis work is organized as follows. In chapter 2, we introduce all the domain topics related to Trident and do background study on the existing solutions and perform literature review. In chapter 3, we provide our original architecture i.e mechanism. In chapter 4 we discuss the simulation with relative pseudo codes and perform evaluation. In 5 we do we draw a conclusion to our work and explore future work opportunities.

## Chapter 2

# Background Study

### 2.1 Related Terminologies

#### 2.1.1 Message Queuing Telemetry Transport (MQTT) [19]

MQTT is a publish/subscribe architecture designed to link devices with minimal bandwidth and power over wireless networks. It's a simple and lightweight protocol that uses TCP/IP or Network Sockets to communicate. The publish/subscribe architecture allows messages to be pushed to client computers without the need for the user to pull the server on a daily basis.

MQTT in general consists of 4 core components. They are: **Publish/subscribe, the message, topics, broker.**

#### **Publish/Subscribe:**

The publish and subscribe system is the first thought. A device should either publish a message on a subject or subscribe to a topic to receive messages in a publish and subscribe system.

#### **Message:**

Messages are the bits of information you want to send between your gadgets, whether it's a command or a piece of data.

## Topics:

Topics are another crucial notion. Topics are how you indicate your interest in incoming messages or where you want the message to be published. Strings separated by a forward slash are used to denote topics. A subject level is indicated by each forward slash. Here's an example on how you would create a topic for a lamp in your home office: *Home/Office/Lamp*. [Note: topics are case-sensitive]

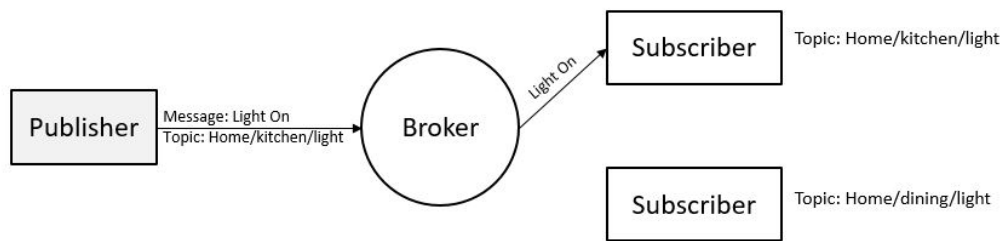


Figure 2.1: MQTT

## Broker:

The broker is in charge of receiving all messages, filtering them, determining who is interested in them, and then releasing the message to all subscribed customers, making it the focal point of communication. Any device that connects to the broker and may publish or subscribe to subjects in order to access the information is referred to as a client. Each client who wishes to send messages does so to a specific topic, and each client who wishes to receive messages does so to a specific topic. All messages with the same theme are sent to the relevant customers by the broker.

In figure 2.1 we can see a general use case scenario where a publisher publishes a message "Light on" to the topic "Home/kitchen/light". And we have two subscribers subscribing to the topics "Home/kitchen/light" and "Home/dining/light". After the message is published on topic 1, only the subscriber subscribed to topic 1 will get the message.

### 2.1.2 Blockchain [10]

In the most fundamental form, blockchain is nothing more than a sequence of blocks, but not in the conventional context. When we use the phrases "block" and "chain" in this sense, we're referring to digital data (the "block") that is contained in a shared archive (the "chain").

In simple term we can also call it a distributed ledger. In which the true state of the ledger is a agreed upon condition by all the nodes in the network at any point of time.

The core blockchain architecture components are:

**Node** - user or computer within the blockchain architecture (each has an independent copy of the whole blockchain ledger).

**Transaction** - The lowest unit of a blockchain system (records, information, etc.) that serves as the blockchain's intent.

**Block** - a data structure used for keeping a set of transactions which is distributed to all nodes in the network.

**Chain** - a sequence of blocks in a specific order.

**Miners** - specific nodes which perform the block verification process before adding anything to the blockchain structure.

**Consensus (consensus protocol)** - a set of rules and arrangements to carry out blockchain operations.

Let's have a closer look at what is a block in a blockchain. Each blockchain block consists of:

- **certain data**
- **the hash of the block**
- **the hash from the previous block**

The data stored inside each block depends on the type of blockchain. For instance, in the Bitcoin blockchain structure, the block maintains data about the receiver, sender, and the amount of coins.

A hash is like a fingerprint (long record consisting of some digits and letters). A cryptographic hash algorithm(SHA 256) is used to produce each block hash. As a result, it is much simpler to identify each block in a blockchain system. A hash is immediately added to a block when the block is created, and any modifications to a block affect the hash's change as well. Simply placed, hashes assist in the identification of a block.

The hash from a previous block is the block's final variable. This establishes a sequence of blocks which is the secret to the stability of the blockchain architecture. Block  $n$ , for example, points to block  $n - 1$ . Both verified and authenticated blocks are extracted from the genesis block, which is the first block in a sequence.

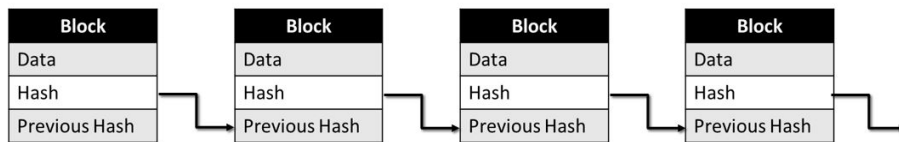


Figure 2.2: block structure

Any corrupt attempts provoke the blocks to change. All the following blocks then carry incorrect information and render the whole blockchain system invalid.

On the other hand, with the help of the powerful computer processors, people may potentially cause all of the blocks to be modified. However, there is a solution known as proof-of-work that prevents this probability. This helps the user to slow the formation of new blocks. In the Bitcoin blockchain architecture, deciding the necessary proof-of-work and adding a new block to the chain takes about 10 minutes. Miners, who are unique nodes inside the Bitcoin network system, do this function. As a reward, miners get to keep the transaction fees from the block that they checked.

Each new consumer (node) who joins the blockchain peer-to-peer network receives a complete copy of the system. After a new block is created by any node, it is broadcast to each node in the blockchain system. After that, each node verifies the block and guarantees that the data it contains is right. If all checks out, the block is attached to each node's local blockchain.

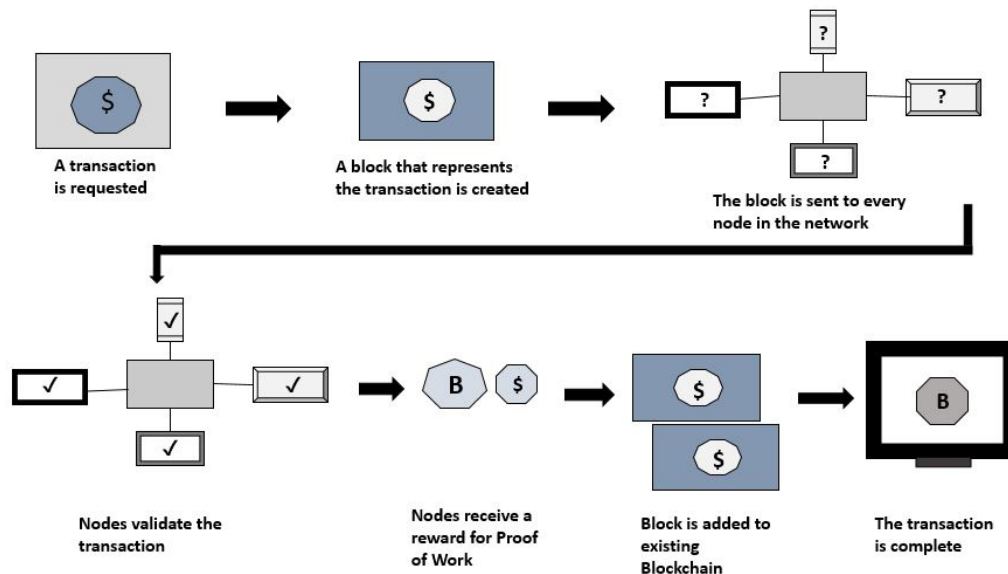


Figure 2.3: How blockchain works

A consensus protocol is generated by all the nodes in a blockchain architecture. A consensus mechanism is a series of network rules that become self-enforcing within the blockchain if anyone follows them. The Bitcoin blockchain, for example, has a consensus rule that every 200,000 blocks, a transaction number must be cut in half. This means that if a block produces a verification reward of 10 BTC, this value must be halved after every 200,000 blocks.

Furthermore, since the protocol defines a limit of 21 million BTC in the Bitcoin blockchain scheme, there can only be 4 million BTC left to mine. Until the algorithm is modified, the supply of Bitcoins will run out after the miners unlock this much.



### 2.1.3 Physically Unclonable Function (PUF)

A Physical Unclonable Function (PUF) is a hardware protection primitive that functions as a function embedded on hardware, in the sense that it should always generate the same output, referred to as a response, for a given input, referred to as a challenge.

In reality, responses are normally noisy; as a result, a fuzzy extraction scheme is used to stabilize each response while also translating it into a cryptographic token, such as a unique ID. Furthermore, each PUF instance implements a distinct feature, in the sense that no two PUF instances can yield similar responses.

Different hardware features, such as a memory's start-up values, can be used to enforce a PUF. PUFs based on inherent device components have the advantage of having no external hardware to build or run. These PUFs are classified as intrinsic PUFs, and they include memory-dependent PUFs like SRAM, DRAM, and Flash PUFs, which are based on the particular characteristics of each of these memory components.

**What is expected from A PUF:** For the same challenge to the same device with a difference in time and surroundings, the device will always return a same response. And in case of different device for the same challenge the response will be different. The expectation is clearly shown in the figure below:

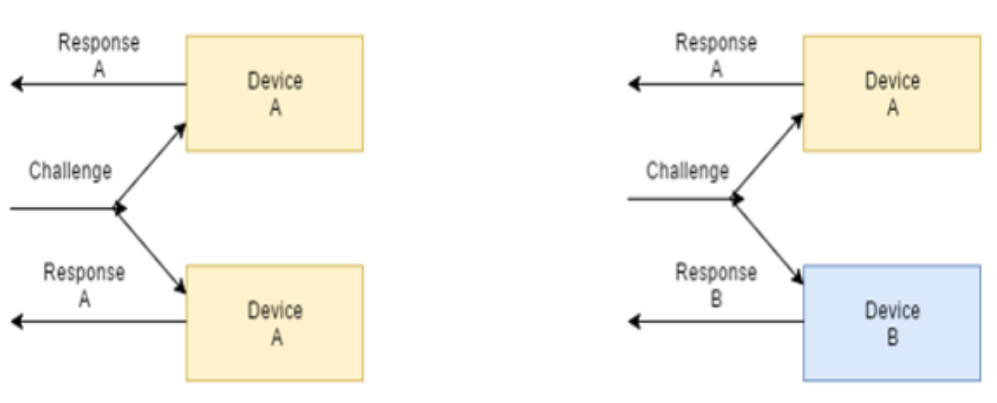


Figure 2.4: Expectations from A PUF

## 2.2 Related Works

In today's IoT applications, several protocols are used as communication protocols. Hypertext Transfer Protocol (HTTP) [5] is one of the most widely used protocols in the Internet of Things (IoT). Constrained Application Protocol (CoAP) [17], Extensible Messaging and Presence Protocol (XMPP) [15], MQ Telemetry Transport Protocol (MQTT) [19] and Advanced Message Queuing Protocol (AMQP) [20] are some more. While choosing a protocol suitable for IoT devices we have to consider energy efficiency, performance, resource usage and reliability [9]. Moreover, when we consider advanced functionalities, qos(quality of service), reliability, and multi casting ability, MQTT protocol is one of the best options [3].

However due to limitations in security measures MQTT is vulnerable to a good number of attacking scenarios. To mitigate limitations of this protocol quite a few number of works have been done before. Singh et al. [18] has proposed ECC based security mechanism which can provide data confidentiality using less resources. Also a study is performed to protect data confidentiality and non-repudiation using RSA and ECC [8]. Niruntasukrat et al. [13] have made a security mechanism that focuses on authentication and authorization of the devices to the broker. Though they have tried to provide security measures, none of them could meet all the necessary requirements.

S Sen et al. [16] have developed an architecture which is scalable. This architecture uses a common shared queue and cache which can be accessed by any broker, at any time. Whenever the CPU limit of a broker is reached, a new broker is deployed using the common cache and traffic is distributed. But they suffer from single point of failure. GS Ramachandran et al. [14] have proposed a distributed architecture using blockchain which distributes the messages exchanged to ensure non-repudiation. Every message published on the broker is broadcast to every other broker and using BFT consensus algorithm new block is added to the blockchain. Authentication is still a concern, and among existing solutions, S Eiroa et al. [4] shows how PUF can be used to authenticate a device in the hardware level. Different types of PUF exist, it is up to the implementer to choose which type to use, as PUF is highly hardware specific. L Negka [11] also have used PUF to detect counterfeit IoT devices, the means to authenticate devices in hardware level. After

a device is manufactured using PUF response, it is registered in the network, so that client can later check to find out if the device is original or not. As talked earlier, no solutions were provided which ensures all security measures at the same time. That is why we needed a new architecture or solution that can provide all necessary features to improve MQTT.

## Chapter 3

# Proposed Mechanism

Trident is proposed to ensure secured M2M communication between constrained devices without any requirement of the services from SSL/TLS layer. As the end devices or clients are resource constrained devices, we decided to make the broker a heavy powered device. Which will be the base of the whole architecture. Trident uses a distributed architecture of MQTT where the brokers work as the nodes and the miner of a blockchain network. We also use PUF to generate unique key for both encryption and decryption in the system. Primary properties of our architecture are:

- Each client/end-device will generate unique output/response using PUF. This response will be used as encryption/decryption key, device identifier.
- Each broker will be a permitted node in the blockchain which will ensure that only authorized brokers can add a block about data or device authentication values like PUF response and device ID.
- The system will consist of two chains. One to store the devices' PUF response and device id (Chain A). Another to store the data transmitted among the devices (Chain B).

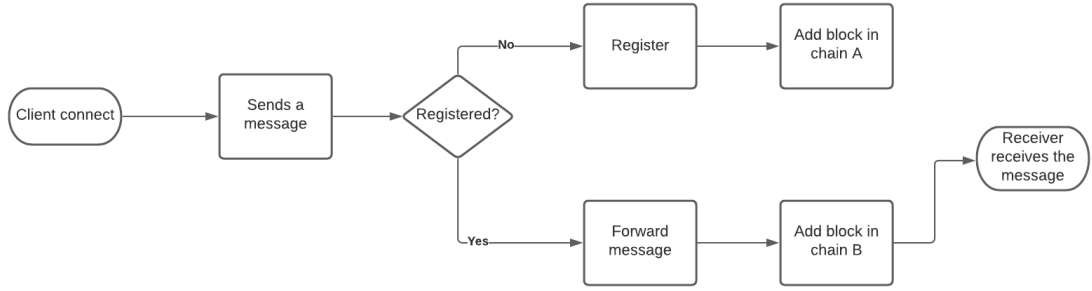


Figure 3.1: Message flow chart

Our proposal ensures that the clients go through two phases to complete the full circle to communicate with other clients. These two phases can be further divided into a total of 6 steps according to sequence of actions. They are:

1. Client registration phase by Broker
2. Publish/subscribe phase from client
3. Authentication phase by broker
4. Acknowledgement phase by broker
5. Storing phase by broker
6. Broker to Broker communication

Next, we introduce you to each step and discuss them in details.

### 3.1 Client registration phase by broker

Before installing a device (client) in the system any of the broker will register the device manually using the architecture of physical unclonable function (PUF).

The broker will pass a PUF challenge,  $PUF_c$  and a serial number called client serial number,  $S$ . Serial number can be any sequence number provided by the broker. Client will then process the challenge  $PUF_c$  and generate the response  $PUF_r$ . Client will also store its serial number and  $PUF_r$  as cryptography key in it's memory for further use in future. And also send  $PUF_r$  to the broker. The broker then will create a new block to add in chain A with value of serial number  $S$  and PUF response  $PUF_r$ .

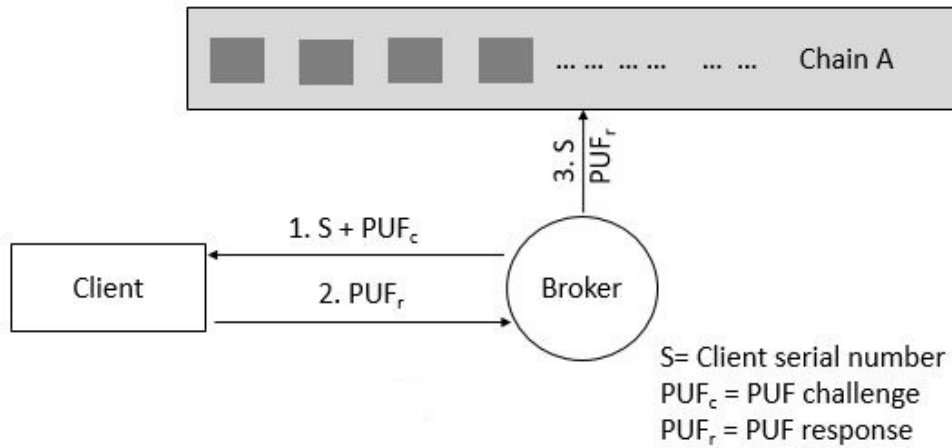


Figure 3.2: Registration phase

Block (A)
index
Timestamp: block creation time
PUF <sub>r</sub>
Serial number
Preceding hash
Hash
nonce

Figure 3.3: Structure of a block in chain A

So, we can understand that for each client, a block will be created and added to the chain A. Completing this step will make the client an authorized entity to the system.

As a distributed system it is not compulsory for a device to register with the specific broker under which it is going to operate. And that makes the system more suitable for implementation in remote cases.

### 3.2 Publish/Subscribe phase from client

In this phase a client will publish a message on a topic or subscribe to a topic. This is the phase where a client tries to communicate or pass a message or sets up itself to receive a message from another client.

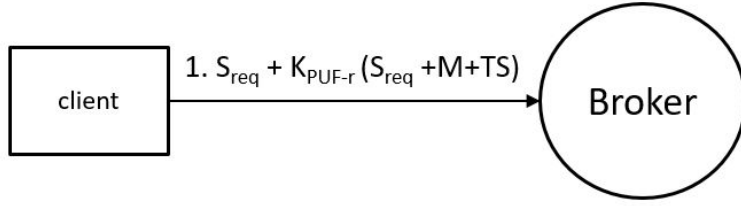


Figure 3.4: Publish/subscribe phase

The client will send a data packet containing its previously saved serial number  $S$  as  $S_{req}$  along with an encrypted message of its serial number  $S_{req}$ , the message  $M$  and timestamp of when the packet was sent,  $TS$ . The encryption will use its previously saved PUF response  $PUF_r$  as a key, in this case we call it  $K_{puf-r}$ . The data or message  $M$  contains the information about publish/subscribe.

### 3.3 Authentication phase by broker

This is the phase in which we mainly authenticate a client. After receiving the packet the broker now has client's serial number  $S_{req}$  and the encrypted message. Broker will then access chain A and parse through the blocks to find the  $PUF_r$  associated to that  $S_{req}$ .

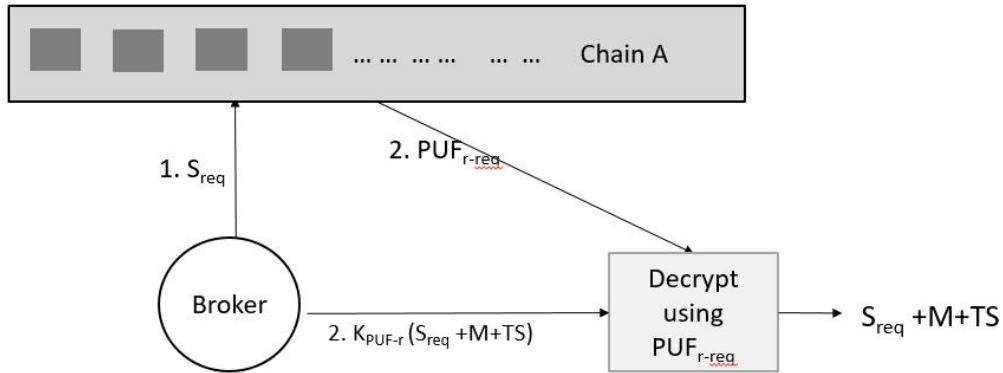


Figure 3.5: Authentication phase

Then using  $PUF_r$  got from the chain, broker will decrypt the client message and find  $S_{req} + M + TS$ . If both of the  $S_{req}$  matches and  $TS$  is valid, we consider the client to be authenticated.

### 3.4 Acknowledgement phase by broker

Only after a successful Authentication phase the broker completes publishing the message on a topic or subscribing the client to a topic and also sends back an acknowledged frame to the client.



Figure 3.6: Acknowledgement phase

### 3.5 Storing phase by broker

Then, If the initial request was a publish request, the broker stores the decrypted data passed by the client in chain B. And that ensures non-repudiation. Otherwise if the initial request was a subscribe request the broker updates his own subscribers list.

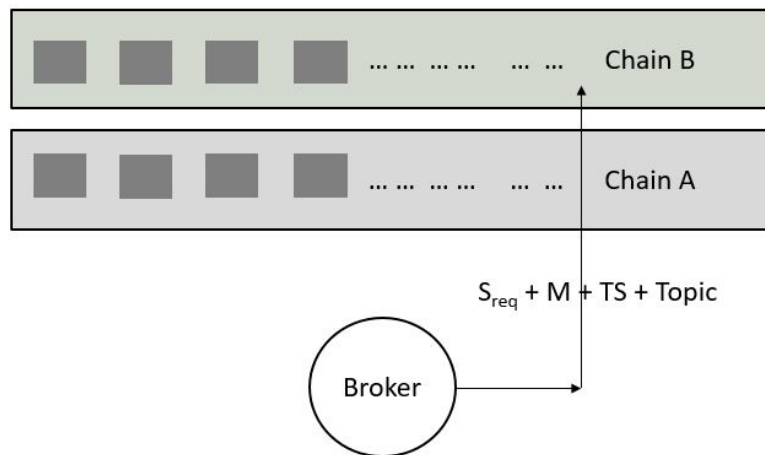


Figure 3.7: Storing phase



The broker stores information of client serial number  $S_{req}$ , message  $M$ , Timestamp  $TS$ , and the *Topic* on which the message was published in the block of chain B.

Block (B)
index
Serial number
Message
Topic
Timestamp
Preceding hash
Hash
nonce

Figure 3.8: Structure of a block in chain B

### 3.6 Broker to Broker communication

What we need to consider is, a case might be that two clients from different brokers need to exchange data. This is allowed through the blocks of chain B. The underlying structure of blockchain makes sure that whenever *nodeA* is trying to add a new block to the *chainA*, *nodeA* has to broadcast the new block to all of its neighbour nodes. This allows data being exchanged among clients from different brokers.

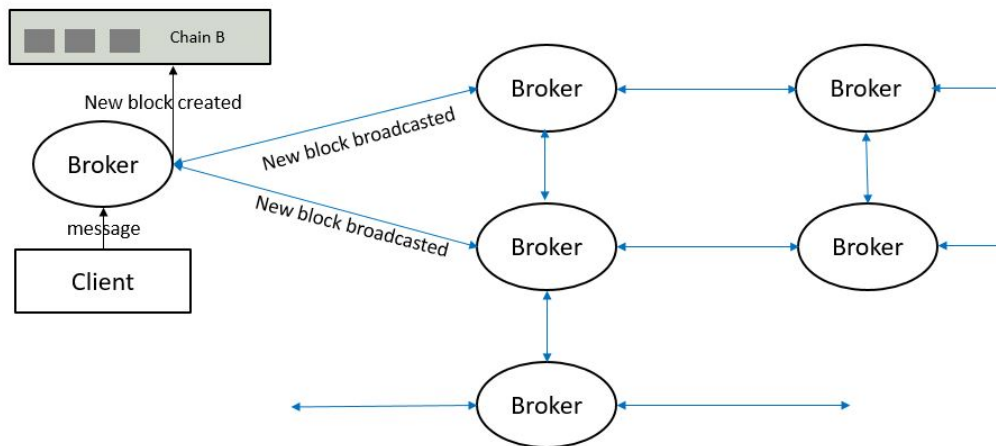


Figure 3.9: Broker to Broker communication

## Chapter 4

# Simulation and Observation

We developed Trident using node.js which is a widely used JavaScript framework. As mentioned earlier, every client encrypts the message using its key before publishing the message on a topic as shown in figure 4.1.

```
plain text message to be published: pub This is a plain text message to be sent
encrypted message to be published: U2FsdGVkX180d+nQ4svmgS9HoCdbuR8aQmvat8w/L/554xdZ1Hbp3qty6Yd+9d9NTTYbcyUkF7GNoviBwKffaw==
published
```

Figure 4.1: Publisher encrypting message before publishing

After encryption, the encrypted message is published. This encryption here helps to ensure **authentication** and **confidentiality** eventually.

On the other hand, after receiving a new publish message the broker decrypts the message to ensure **authenticity** of the message as in figure 4.2. Then the broker creates a new block in the chain with information from this publish message [figure 4.3]. This new block ensures **non-repudiation**.

```
encrypted message received by the broker: U2FsdGVkX180d+nQ4svmgS9HoCdbuR8aQmvat8w/L/554xdZ1Hbp3qty6Yd+9d9NTTYbcyUkF7GNoviBwKffaw==
message decrypted by the broker: pub This is a plain text message to be sent
```

Figure 4.2: Broker decrypting the published message

```

{
  "index": 1,
  "timestamp": "01/06/2020",
  "data": {
    "sender": "mqttjs_918705e1",
    "message": "pub This is a plain text message to be sent",
    "topic": "MQTT_TOPIC"
  },
  "precedingHash": "f556af85a25f2883d26b011d041ab717f3a82f6c9e2d8bd897510c13ec14e61f",
  "hash": "0000250276b9f2a0c832ba73a9dd9b91121575d7b601508af0aed253087d7c07",
  "nonce": 116251
}

```

Figure 4.3: Broker adding a new block

Finally, the subscriber receives the message in its encrypted form [figure 4.4]. Thus, our simulation meets all of our objectives and goals.

```

received by target: U2FsdGVkX1+wXg4tnHrjrEDkXLw9Mg7JhtmDL3Lqdre9V/0wj00hgsHvyrEFUXLV2bfkV6Crs5rL4k74PSHgVQ==

```

Figure 4.4: Subscriber receiving the message

## 4.1 Discussion on our works:

At the very beginning we chose MQTT to work on as it is one of the most promising protocol if all the limitations can be mitigated. We kept looking for existing solutions provided by the others and found that no solution can individually solve all the limitations. So, there we got our target i.e. providing all the solutions in the same architecture. Then we looked for the lightweight key generation system without using help from any external entity. And we used PUF to do that. Then we designed our architecture. After that, we developed our architecture using Node.js and did the simulations.

## Chapter 5

# Future Works and Conclusions

In the future we are planning to modify trident based on different cases. Right now we have proposed more of a general version of trident which can be implemented in any sector for use. We plan to modify it according to specific sector's requirements. And then test those modified versions in depth by implementing in real life scenarios. To do so, we might need a large amount of research funding. We hope to get it from the concerned representatives in near future. Also as we have motioned earlier, PUF implementation is highly device and circuits specific. So we have to find the specific PUF challenge and criteria for specific sensors and devices in the future.

# References

- [1] Renu Aggarwal and Manik Lal Das. “RFID Security in the Context of ”Internet of Things””. In: *Proceedings of the First International Conference on Security of Internet of Things*. SecurIT '12. Kollam, India: Association for Computing Machinery, 2012, pp. 51–56. ISBN: 9781450318228. DOI: 10.1145/2490428.2490435. URL: <https://doi.org/10.1145/2490428.2490435>.
- [2] Syaiful Andy, Budi Rahardjo, and Bagus Hanindhito. “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)*. IEEE. 2017, pp. 1–6.
- [3] Niccolò De Caro et al. “Comparison of two lightweight protocols for smartphone-based sensing”. In: *2013 IEEE 20th Symposium on Communications and Vehicular Technology in the Benelux (SCVT)*. IEEE. 2013, pp. 1–6.
- [4] Susana Eiroa et al. “Using physical unclonable functions for hardware authentication: A survey”. In: (2010).
- [5] Roy Fielding et al. *Hypertext transfer protocol–HTTP/1.1*. 1999.
- [6] James King and Ali Ismail Awad. “A distributed security mechanism for resource-constrained IoT devices”. In: *Informatica* 40.1 (2016).
- [7] Evangelos Kosmatos. “Integrating RFIDs and Smart Objects into a Unified Internet of Things Architecture”. In: *Advances in Internet of Things* 01 (Jan. 2011), pp. 5–12. DOI: 10.4236/ait.2011.11002.

- [8] Abdessamad Mektoubi et al. “New approach for securing communication over MQTT protocol A comparaisn between RSA and Elliptic Curve”. In: *2016 Third International Conference on Systems of Collaboration (SysCo)*. IEEE. 2016, pp. 1–6.
- [9] Dae-Hyeok Mun, Minh Le Dinh, and Young-Woo Kwon. “An assessment of internet of things protocols for resource-constrained applications”. In: *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*. Vol. 1. IEEE. 2016, pp. 555–560.
- [10] Satoshi Nakamoto. *Bitcoin: A peer-to-peer electronic cash system*. Tech. rep. Manubot, 2019.
- [11] Lydia Negka et al. “Employing blockchain and physical unclonable functions for counterfeit IoT devices detection”. In: *Proceedings of the International Conference on Omni-Layer Intelligent Systems*. 2019, pp. 172–178.
- [12] A. Niruntasukrat et al. “Authorization mechanism for MQTT-based Internet of Things”. In: *2016 IEEE International Conference on Communications Workshops (ICC)*. 2016, pp. 290–295. DOI: 10.1109/ICCW.2016.7503802.
- [13] Aimaschana Niruntasukrat et al. “Authorization mechanism for MQTT-based Internet of Things”. In: *2016 IEEE International Conference on Communications Workshops (ICC)*. IEEE. 2016, pp. 290–295.
- [14] Gowri Sankar Ramachandran et al. “Trinity: A byzantine fault-tolerant distributed publish-subscribe system with immutable blockchain-based persistence”. In: *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE. 2019, pp. 227–235.
- [15] Peter Saint-Andre et al. “Extensible messaging and presence protocol (XMPP): Core”. In: (2004).
- [16] Souranil Sen and Aruna Balasubramanian. “A highly resilient and scalable broker architecture for IoT applications”. In: *2018 10th International Conference on Communication Systems & Networks (COMSNETS)*. IEEE. 2018, pp. 336–341.
- [17] Zach Shelby, Klaus Hartke, and Carsten Bormann. “The constrained application protocol (CoAP)”. In: (2014).

- [18] Meena Singh et al. “Secure mqtt for internet of things (iot)”. In: *2015 fifth international conference on communication systems and network technologies*. IEEE. 2015, pp. 746–751.
- [19] OASIS Standard. “MQTT version 3.1. 1”. In: *URL <http://docs.oasis-open.org/mqtt/mqtt/v3.1>* (2014).
- [20] Steve Vinoski. “Advanced message queuing protocol”. In: *IEEE Internet Computing* 10.6 (2006), pp. 87–89.