

FlyTrap: Decentralised Blockchain Security & Auditing Architecture for IoT and MQTT Brokers

Konrad M. Dryja

A dissertation submitted in partial fulfilment
of the requirements for the degree of
Master in Science
of the
University of Aberdeen.



Department of Computing Science

2020

Declaration

No portion of the work contained in this document has been submitted in support of an application for a degree or qualification of this or any other university or other institution of learning. All verbatim extracts have been distinguished by quotation marks, and all sources of information have been specifically acknowledged.

Signed:

Date: April 5, 2020

Abstract

An expansion of the title and contraction of the thesis.

Acknowledgements

Much stuff borrowed from elsewhere

Contents

1	Introduction	8
1.1	Overview	8
1.1.1	Internet of Things	8
1.1.2	Security of data	9
1.1.3	MQTT	9
1.1.4	FlyTrap	9
1.2	Motivation	10
1.2.1	MQTT	10
1.2.2	Blockchain	10
1.2.3	Legislature	10
1.3	Goals	10
1.4	Report Structure	11
2	Background & Related Work	13
2.1	MQTT	13
2.1.1	Message persistence	14
2.1.2	Implementation	14
2.1.3	Publishing	15
2.1.4	Subscribing	16
2.2	Blockchain	18
2.2.1	Architecture	18
2.2.2	Consensus Algorithms & Proof-of-Work	19
2.2.3	Proof-of-Stake	20
2.2.4	Proof-of-Authority	21
2.2.5	Ethereum	21
2.3	IoT, Hyperledger and GA	22
3	Requirements & Architecture	23
3.1	Requirements	23
3.1.1	User stories	23
3.1.2	Use-case Scenarios	24
3.1.2.1	Scenario #1: Air Quality study in the UK	24
3.1.2.2	Scenario #2: Data breach in an oil drilling facility	24
3.1.2.3	Scenario #3: Unsatisfied Customer	25

3.1.2.4	Scenario #4: Information monetization	25
3.1.2.5	Scenario #5: Securing access to the broker	25
3.1.3	Functional Requirements	26
3.1.4	Non-functional Requirements	26
4	Design	28
4.1	Secure Proxy	28
5	Implementation	30
6	Evaluation & Testing	31
7	Discussion	32

Abbreviations

ACL Access Control List.

CCPA California Consumer Privacy Act.

GDPR General Data Protection Regulation.

IoT Internet of Things.

MQTT Message Queuing Telemetry Transport.

PII Personal Identifiable Information.

PoA Proof-of-Authority.

PoS Proof-of-Stake.

PoW Proof-of-Work.

QoS Quality of Service.

RFID Radio-Frequency Identification.

TCP Transmission Control Protocol.

TLS Transport Layer Security.

Chapter 1

Introduction

1.1 Overview

1.1.1 Internet of Things

Internet of Things, also known as IoT, is a growing field within technical industries and computer science. It is a notion first coined in Ashton [1] where the main focus was around RFID (radio-frequency identification) tags - which was a simple electromagnetic field usually created by small-factor devices in the form of a sticker capable of transferring static information, such as a bus timetable or URL of a website (e.g. attached to a poster promoting a company or an event). Ashton argued the concern of data consumption and collection being tied to human presence at all times. In order to mine information, human first was required to find relevant data source, which then could be appropriately evaluated. However, as it was accurately pointed out, people have limited resources & time, and their attention could not be continuously focused on data capture. Technologist suggested delegating the task to the machines themselves; altogether remove the people from the supply chain. A question was asked, whether “things” could collect data from start to finish. That paper is known to be the first mention of IoT and building stone, de facto defining it as an interconnected system of devices communicating with each other without the need for manual intervention.

With time and ever-expanding presence of smartphones, personal computers and intelligent devices, the capabilities of those simple RFID tags were also growing beyond just a simple static data transmission functionalities. Following the observation by Moore et al. [17], the size of integrated circuits was halving from year to year, allowing us to put more computational power on devices decreasing in size. They were now not only capable of acting as a beacon, but actively process the collected information (for example, temperature) and then pass it along to a more powerful computer which then could make decisions on whether to increase or decrease the strength of radiators at home - all without any input from the occupants. Eventually, IoT found their way to fields and areas such as households (smart thermostats or even smart kettles), physical security (smart motion sensors and cameras) or medicine (smart pacemakers). This number is expected only to grow in the future. CISCO white paper [8] speculates that we might see 50 million of those devices by the end of 2020.

1.1.2 Security of data

The growing presence of smart-devices significantly increased the convenience and capabilities of “smart-homes” - at the same time, IoT also started handling more and more sensitive data - especially considering the last example from the previous paragraph. Scientists from the University of Massachusetts successfully performed an attack on a pacemaker [10], reconfiguring the functionality, which - if performed with malicious intents - could have tragic consequences. Nevertheless, even less extreme situations, such as temperature readings at home, are nowadays heavily regulated by data protection laws. Examples being the General Data Protection Regulation (GDPR) introduced by European Commission [7] or California Consumer Privacy Act by California State Legislature [5]. Collection of data is required to be strictly monitored and frequently audited in case of a breach - which also includes restrictions on the collection of Personal Identifiable Information (PII, as per GDPR). Those and more put an obligation on every company willing to exchange user data to govern the data appropriately and ensure its security - which includes data collected by Internet of Things devices.

1.1.3 MQTT

IoTs are usually low-power with limited computational power - mostly to decrease the required maintenance and ensure long-lasting life, without the need of replacing the power source (which is often a fixed battery) - meaning that only minimum amount of work should be performed on the “thing” itself, instead of sending it off to a centralised structure (e.g., a server hosted on the cloud) for further processing. One of the popular choices includes an intermediary, a broker, relaying communication between clients connected to it. That way, Peer-to-Peer connection is not required and can be wholly delegated to separate backend server. A popular choice for the broker is MQTT (Message Queuing Telemetry Transport)¹ standard defining the exact shape and form of TCP packets, handling unexpected timeouts & reconnects along with distributing channels of communication onto different topics containing separated information. From there, clients can either subscribe (i.e. consume) or publish (which can also be used for issuing commands) the data. Unfortunately, the OASIS standard introduces limited security capabilities (offering only username/password authentication) and no auditing or logging.

1.1.4 FlyTrap

This project will be aiming to develop a novel approach - further referred to as **FlyTrap** - for handling security in systems utilising MQTT brokers and their implementations, focusing on platform-agnostic solution hosted within a containerised environment. It will not depend on the specific software implementing the broker but instead will aim to work with any broker that fully implements MQTT v5.0 standard. Furthermore, to ensure decentralised operation resistant to data breaches, downtime and full transparency, Ethereum² platform would be used as a data layer: capturing relevant interaction as publicly available transactions. In order to limit the quantity of data put on the blockchain (as computational and storage power there is limited), I will also introduce several rules dictating logging of only specific events. The system’s purpose is to incorporate **Authentication, Authorisation and Accountability** (AAA) framework to IoT devices communicating through MQTT.

¹<https://mqtt.org/>

²<https://ethereum.org/>

1.2 Motivation

1.2.1 MQTT

MQTT v5.0 (as per the specification³) does not dictate nor specify any requirements regarding the security. It does offer an option of restricting some topics only to specific users, defined in access control lists (ACLs). The users then are required to provide a password when initiating a connection with the broker. Although, the basic username/password authentication is known to be cumbersome, only offering limited security. This also puts a burden on system administrators to maintain those ACLs in some centralised system, which then again is at risk of breaches or leakage. Moreover, placing the burden on a singular MQTT broker creates a single point of failure, where system downtime could halt the entire architecture.

1.2.2 Blockchain

By decentralising the data layer of the AAA framework and in the process, placing it on a distributed ledger, I can ensure maximised uptime and complete transparency of performed transactions. Events such as permission changes, failed authentication attempts will be recorded as a separate transaction which then could be audited by anyone knowing the public address of the system. This then could be handed over to authorities or auditing corporations to ensure that data is passed lawfully. Utilising Blockchain technologies also opens an opportunity to require payment (in the form of cryptocurrency) from potential consumers of data effectively expanding the business model.

1.2.3 Legislature

The rise of awareness of the necessity of data protection also encouraged governments to introduce legal requirements (such as GDPR or CCPA) of data governance and face heavy fines in case of non-compliance. MQTT standard and their implementation at the moment would be considered non-compliant, due to effectively no way to trace past operations. General Data Protection Regulation requires entities handling user data to maintain proper retention of data and purge if requested by the data owner. MQTT at its current state is not capable of either, as messages are removed from the broker as soon as they are consumed (with small exceptions), leaving no trace of “who” accessed “what” (not to mention questions such as “why” they accessed it).

1.3 Goals

The project can be divided onto four main goals and two extras, leaving some field for manoeuvring in case of roadblocks or difficulties resulting from the challenges faced in the dissertation. By having soft targets, I will be able to stop sooner in case of overestimating the schedule, or carrying on with extra work, should I find myself meeting the targets quicker than expected.

Main Goals:

1. Design structure of blockchain network, relevant data models that would be placed on the blockchain and deploy on the Ethereum platform, capable of recording transactions and allowing for modification of ACLs, i.e. which wallets are permitted to access specific resources on the MQTT brokers.

³<https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html>

2. Design rules that would be used for capturing the transactions. For example, a rule stating that if the client makes more than five consecutive, failed authentication attempts would be placed on a blacklist and that action would be added onto the blockchain as a transaction.
3. Design containerised software acting as a secure proxy between brokers and connecting clients. This will handle both authentication and log performed action as an immutable transaction on a blockchain network. Logging will only be performed if the requested operation triggers some pre-defined rules.
4. Perform evaluation of the designed solution using an off the shelf MQTT broker and a range of experimental scenarios with a simulated network of MQTT clients.

Extra Goals:

1. Create public API for the auditors to freely access the contents of blockchain and thus transactions containing information about suspicious operations.
2. Generalise the implementation of the framework so it can be deployed with any broker following the MQTT standard.

What project is NOT trying to be:

- Design a new blockchain platform from scratch. Rather existing solution - Ethereum - is going to be used.
- Write / modify operating system of IoT devices.
- Design a new MQTT Broker. The system is going to be built on top of MQTT layer.

1.4 Report Structure

The dissertation is going to be divided onto seven chapters, each describing the following aspects of the project:

- Chapter 1 **Introduction** chapter will outline the main motivation behind the project and introduce the notions used a building block in the design. It will also list goals and no-goals defining success.
- Chapter 2 In **Background & Related Work**, similar research and state of the art will be described along with outlining the differences between them and this project. Furthermore, a thorough explanation of used software will also be attached, such as what is blockchain, Ethereum, MQTT.
- Chapter 3 **Requirements & Architecture** will include analysis of both functional and non-functional requirements, main use-cases that are driving the project and high-level overview of the architecture explaining how each element addresses each of the requirements.
- Chapter 4 **Design** will be an expansion to architecture, providing an explanation on how each of the elements connects to another.

Chapter 5 **Implementation** will talk about the process of implementation of the design into software. It will include notions such as followed processes, used frameworks and sample code snippets.

Chapter 6 Inside **Testing & Evaluation** a comparison between state-of-the-art software, vanilla and FlyTrap will be performed. Tests checking for performance impact and whether the common attack can be detected/stopped will also be run.

Chapter 7 **Discussion & Future Work** will include conclusions of the project, elements that were left-over, but beneficial for future iteration and all blockages encountered throughout.

Chapter 2

Background & Related Work

In this section, I will list all technologies that are used in this project along with discussing other papers which were trying to address security with IoT devices by also trying to include blockchain technology.

2.1 MQTT

When designing architecture with the main target being the communication of many (even couple of thousands a second) clients continually exchanging data, scalability and availability needs to be kept in mind. The first and obvious solution would be to directly connect data consumers and data produces, by making them communicate in Peer-to-Peer fashion, removing the need for any extra infrastructure. This might work perfectly fine with small systems (disregarding issues such as dynamic DNS or static IP). However, as the number of clients requesting access to data increases, the total capacity of the sensor would eventually be capped - since IoT usually are of limited power and computation capacity. Imagine a scenario where a single temperature sensor continually getting bombarded with requests for current readings. It might be able to cope up to 5 incoming requests every second, everything else would cause malfunction or significantly slower response times.

Then there is also an issue of security. By allowing clients to connect to our IoT devices, we are opening a new attack vector. What if the client does not want only to access the temperature readings, but perhaps inject a worm which would intercept other sensors (such as cameras). Recently “smart nannies”, responsible for alerting the parents when the child is crying and also relieving the adults from having to be always nearby, gained popularity. A direct camera feed could be accessed via a smartphone, no matter where. This eventually led to exploitation, as it was found that many of those devices were vulnerable to remote access by third parties[20].

MQTT aims to address those issues (and not only), by moving the communication to a separate entity, which operates in a publish-subscribe fashion. This would mean that IoT devices only have to publish information that is available to them (e.g. temperature readings), allowing to altogether remove remote access, effectively mitigating this particular attack vector. Furthermore, the MQTT brokers can be further placed behind load balancers and such to enhance their availability further.

In short, MQTT, fully expanded to Message Queuing Telemetry Transport is an open protocol, certified by OASIS and ISO[2], responsible for the publisher-subscriber architecture. It is important to point out that MQTT is not a piece of software or a server, but rather a set of standards

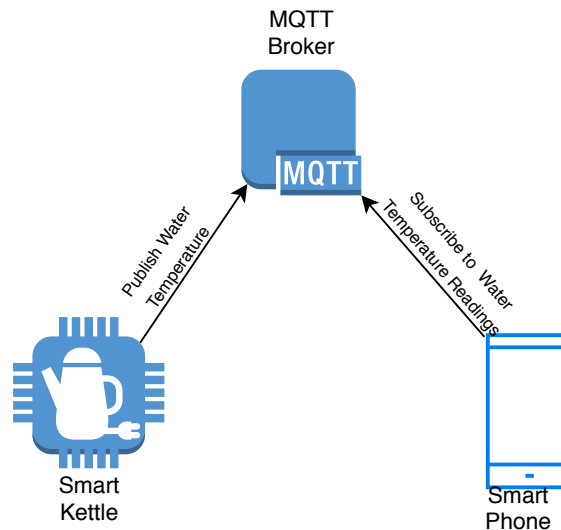


Figure 2.1: MQTT Broker Architecture

defining what potential clients can expect (what kind of responses and data) while connection to brokers following the standard. Figure 2.1 briefly shows how MQTT-compatible broker can relay information between clients. Smart Phone and Smart Kettle do not have to be online at the same time in order to receive information, nor Smart Phone is even permitted to initiate a direct connection to Smart Kettle. The broker's responsibility is to track connected subscribers (which must specify the topic of their interest) and maintain the connection until subscribe advertises session termination or abruptly disconnects (e.g. loss of power or unreliable connection).

In MQTT architecture, Client ID identifies each of the connecting entities (publisher/subscriber) and topic identifies a bridge between publishers and subscribers connected to the same topic. For example, a Smart Kettle could be publishing temperature readings under a topic called "UK/Aberdeen/Kettle" - then, a smartphone would need to request the same topic to receive those readings.

2.1.1 Message persistence

This will be discussed in depth when I will describe the process of publishing and subscribing, but it is worth pointing out that by default, the messages are not saved nor cached on the broker. That is if Kettle publishes the temperature reading, but no subscribers are listening to this information, the message will perish. This is not ideal, for a situation where a smart device could wake up only every couple of minutes and then go to low-power mode again. To address this, MQTT Messages can be enriched by "Retain" flag. If such a flag is present, the broker will keep the message and send it straight away to any new subscribers requesting given topic. This is also useful for issuing commands to IoT devices. For example, a phone could send a command to turn off the lights with "Retain" flag set. Then, the smart light switch could check for retained messages every couple of minutes, removing the need for constant connection.

2.1.2 Implementation

MQTT by itself is only a collection of standards instructing implementors on what patterns should be followed and the structure of particular messages. Thus it is not shipped with any piece of software. It assumes operation on TCP layer of the network (although newer versions also allow

for WebSocket support [16]), thus also allowing for encrypted connection via Transport Layer Security. Every exchanged message is a TCP packet, following a strict convention- which in case of deviation is discarded as corrupted.

Two of the implementations that I have considered during this project are Mosquitto¹ by Eclipse and Moquette². The former written in C and the former in Java, although there is many, many more. In a paper by de Oliveira et al. [6], scientists compare Moquitto and RabbitMQ, arguing their choice by the offered cloud infrastructure with more significant scalability opportunities. Moreover, some solutions are paid, whereas the considered approaches are free and open-source, allowing for a better understanding of operations. The paper is concluded with the finding that hardware and network latency has a far more significant impact on the performance, rather than the choice of the individual broker, which leaves the decision mostly down to offered extra features.

Mosquitto also offers a Docker container [15] in which the broker can be run, allowing for further isolation and removal of extra dependencies.

2.1.3 Publishing

The most popular method of passing MQTT messages is still under the Transport layer, as TCP packets. This allows for slightly higher freedom (compared to stricter protocols, such as HTTP), at the cost of more sophisticated parsing. MQTT standard is composed of several message types with the most important being:

- CONNECT - used to initiate the connection
- PUBLISH - used by the client to publish messages and by the broker to publish messages to subscribers
- SUBSCRIBE - used by the client to request a subscription to a given topic
- UNSUBSCRIBE - used by the client to request removal of subscription to given topics
- Along with relevant *ACK counterparts (e.g. CONNACK) used to indicate the successful transmission of the message

As shown in figure 2.2, the publishing flow starts with the CONNECT messages. Inside, there are several flags included, such as Quality of Service requested (MQTT can periodically send heartbeat ping to clients to check if they are still alive), requested version of MQTT protocol (at the moment, v5.0 and v3.1). This part is also referred to as “Variable header”. The second part, known as “Payload” consists of the client ID.

Then, once the client has established its identity to the broker, the broker responds with CONNACK message, which contains bit informing whether a further connection is allowed or not. From this point, the client is cleared to start publishing session.

Usually, for every message to be published, there is one PUBLISH packet. A newer version of MQTT allows for spreading larger messages across multiple packets, although this will not be covered in this paper. The PUBLISH packet contains mostly two properties - topic to be

¹<https://mosquitto.org/>

²<https://github.com/moquette-io/moquette>

published on and the actual payload. Each of the properties is prepended with 8 bytes indicating the length. From this fact, we can derive the maximum possible size of individual payload - 65535 characters (pure ASCII, no Unicode, which may take more than 1 bytes per character). Same as with CONNECT, each message is responded to with PUBACK, acting as a receipt for receiving the payload.

The client can continue to publish new messages without having to connect again, as long as the TCP session has not been terminated. Should the client want to disconnect, it should follow standard TCP flow, i.e. issue FIN/ACK packet to the broker. For situation, where the connection has been terminated abruptly, there are options such as Will flag (message to pass in case of sudden disconnection) or Keep Alive (to indicate how long should the connection be kept alive for before assuming the client has lost connection).

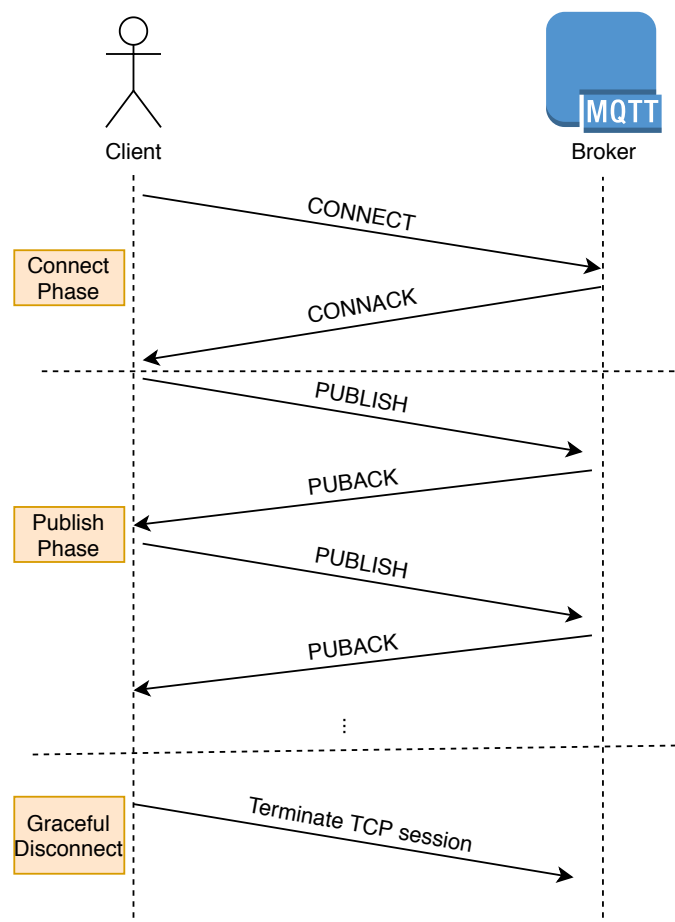


Figure 2.2: Publishing flow with MQTT

2.1.4 Subscribing

Subscribing flow is quite similar to Publishing, with some minor differences. Following figure 2.3, first and foremost, a connection needs to be established by instating standard TCP/TLS session and then sending CONNECT packet. The contents follow the same standard, i.e. containing information such as Client ID or even optional parameters in the form of “key: value” (particularly useful for this project).

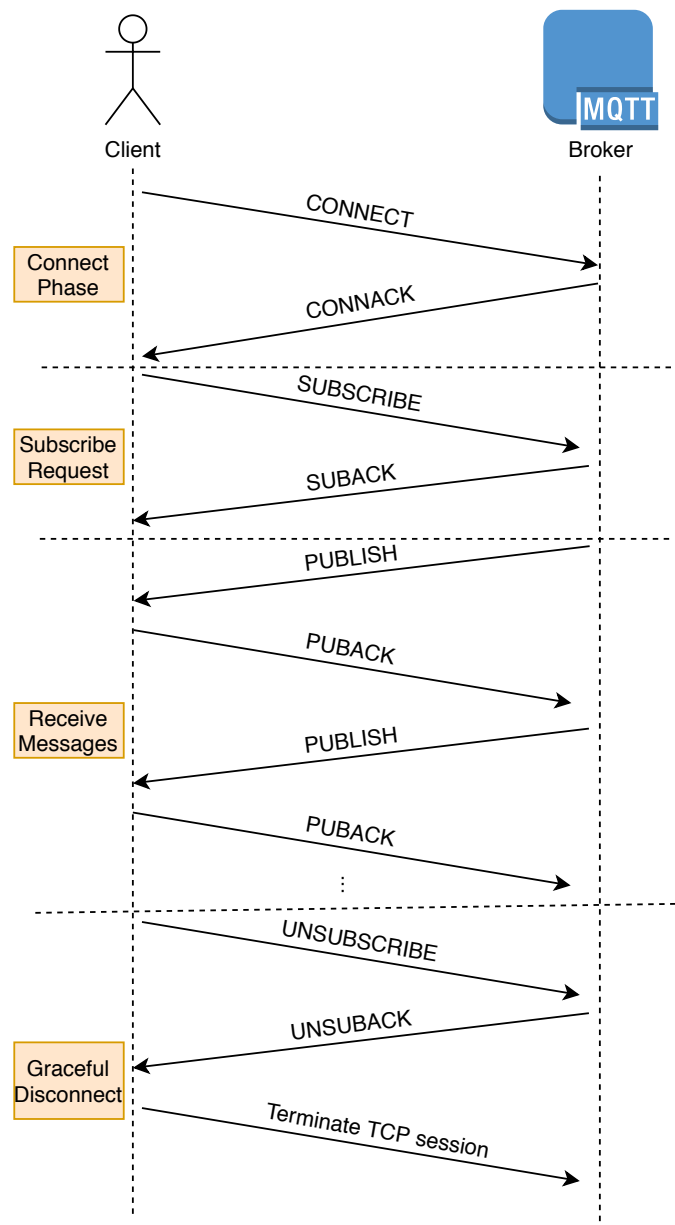


Figure 2.3: Subscribing flow with MQTT

After a successful connection, the client can proceed to send a request for subscription. Similar with `PUBLISH` packet, the client specifies the type of the packet in the variable header and then requested topic for subscription in the payload. The extra element is the QoS flag - Quality of Service. MQTT has three levels of QoS:

1. 0 - No response to `PUBLISH` messages
2. 1 - `PUBLISH` messages will be followed by `PUBACK`
3. 2 - More granular control over `PUBLISH`, with extra packets such as `PUBREC` (Publish Received), `PUBREL` (Publish Release) and `PUBCOMP` (Publish Complete).

Once the `SUBSCRIBE` message has been processed and approved by the broker, it will issue `SUBACK` message and remain connected to the client. From this point, any message that is

82	04	00	01	00	07	F	L	Y	T	R	A	P	00
1	2	3	4	5	6	7	8	9	10	11	12	13	14

Table 2.1: Example SUBSCRIBE to topic FlyTrap packet

published on the topic specified in SUBSCRIBE packet will be published (as PUBLISH packet) to every client currently subscribed to it. Of course, depending on requested QoS, the broker might then await for PUBACK message (or even issue other messages such as PUBREC, PUBREL, PUBCOM). The diagram demonstrates a simple exchange with QoS set to 1.

To close off MQTT, I also wanted to overview an example packet and dissect it byte by byte to demonstrate exactly what kind of information is included - this can be seen in table 2.1

- 1 Control field, specifies the type of the message (CONNECT, SUBSCRIBE etc.)
- 2 Remaining length of the message. Can be expanded to 2 bytes.
- 3-4 Packet ID
- 5-6 Payload length
- 7-13 Payload. Corresponding hex encoding of characters, replaced with actual characters for clarity
- 14 Requested QoS

2.2 Blockchain

Lots of concepts in this paper involve blockchain methodologies, which, by itself, is an expansive area. As part of this section, I will be only covering the most relevant topics necessary to understand the design choices taken within my project, but further reading is strongly encouraged.

2.2.1 Architecture

Blockchain often goes by its infamous name of simply overly complicated linked-list, and in fact, it is not very far away from being true. The concept was first introduced and popularised by Nakamoto et al. [18] in a paper introducing a highly controversial notion of digitalising and decentralising currency, by moving it into a structure called a blockchain. Blockchain network was meant to operate on a peer-to-peer basis, with different peers validating each other's transactions and holding a copy of the entire block. This removes the need for a central authority governing the currency (for example, central banks), by placing a copy of all records on every participant's computer - one problem remained, and that was trust. How do we trust other peers that they do not inject fraudulent transactions? However, before I answer this question, let us focus on figure 2.4, which outlines the difference between distributed and centralised ledgers. With the current economic model, usually there exist some central authority (in this example, a central bank) which is responsible for tracking, verifying and authorising all transactions between participants. Compare it with a decentralised ledger, where there is no such central entity. Instead, each participant verifying all transactions that happen between nodes. They no longer have to trust Central Bank to do their job currently, as they are free to confirm the authenticity of all transactions themselves. Nevertheless, again, we get back to the same question - how does the authentication happen?

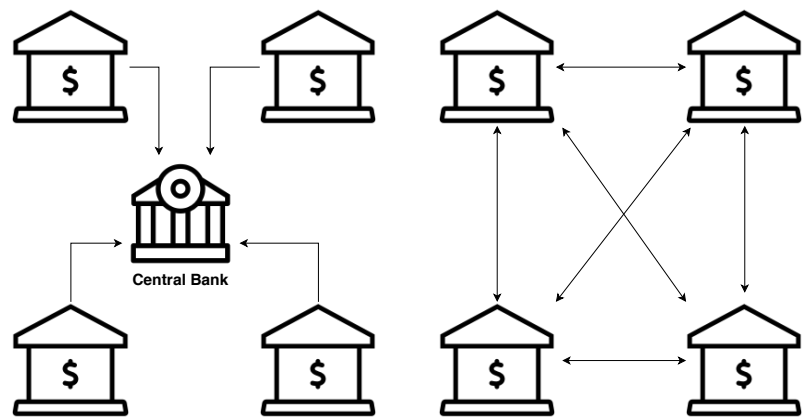


Figure 2.4: Centralised Ledger (on the left) vs Decentralised Ledger (on the right)

2.2.2 Consensus Algorithms & Proof-of-Work

Before a participant can add their transaction (“block” from blockchain) to the public records (“chain” from blockchain), we need a cryptographically secure mean to verify whether this particular participant can add this block. Establishing trust between participants on a blockchain is often referred to as “consensus”. Thus, several consensus algorithms exist. Currently, perhaps the most popular one, it is proof-of-work. In fact, PoW dates even before the paper by Satoshi Nakamoto, all the way back to Jakobsson and Juels [12]. It utilises one of the most critical properties of hash functions, that is pre-image resistance. The blockchain will offer a cryptographic puzzle to the participant willing to add a new block. This puzzle would be based on reversing a hash, i.e. a hash would be generated, and the participant would be tasked with reversing it, thus getting the original value. This “puzzle” is also often referred as mining a new block, that is, finding a value that after passing through specified hash function would produce expected output (also known as cracking hashes) - that is also part of the reason why modern mining requires much computational power.

Then, everyone starts a race towards reversing this hash. The first person to achieve the target is rewarded with a possibility to add new a block to the network (along with the found value). In the future, any peer can verify the authenticity by feeding the attached value through the hash function and verifying whether the obtained value matches the expected hash. All of it is possible since computing hashes is relatively fast and not a very computationally expensive operation. At the very end, when attaching the new block to the chain, the miner is usually rewarded with cryptocurrency, which can then later be exchanged with other participants.

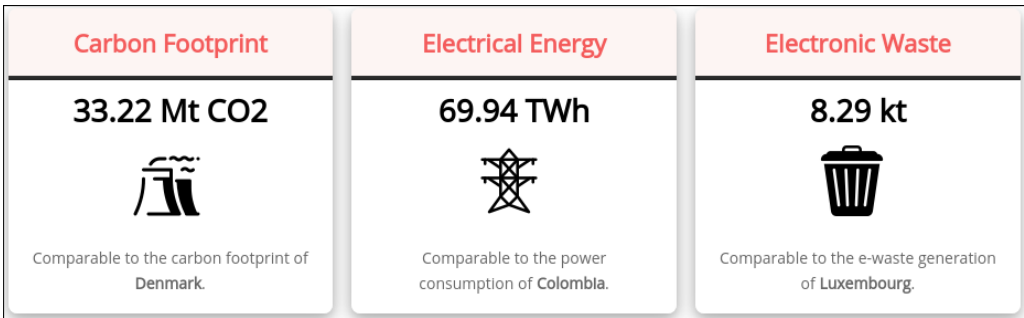


Figure 2.5: Annualized Total Footprint of Bitcoin network [11]

Of course, this approach has several downsides. First of all, all the computational power is effectively wasted to this cryptographic puzzle, with no real end-use - especially if you take part in the race to crack the next hash and someone ends up being faster than you - all your effort went for nothing. This was widely discussed by scientists [9], who currently point out a negative impact on the environment. As reported by portal Digiconomist [11], as of 2020, the annualised carbon footprint of Bitcoin network can be compared with the carbon footprint of the entire country of Denmark, with extra samples such as electrical consumption or electronic waste in figure 2.5

Another problem that Proof-of-Work algorithms create is a 51% attack. Nowadays, setting up your Bitcoin node and starting to mine is not very feasible since people with higher hash rates (the speed at which a person can crack hashes) usually form organisations, that share this power amongst each other and then once they can crack the individual hash, reward each of the members with only a small portion. This might sound good for individuals since now they are guaranteed a payout (rather than risking taking part in the race and losing, winning nothing), but it effectively defeats the decentralised concept of blockchain. If one organisation holds more than 51% of the hash rate of the entire blockchain, it can start authorising fraudulent blocks and adding them to the chain. Since they hold the majority of the network's hash rate, nobody can defy them.

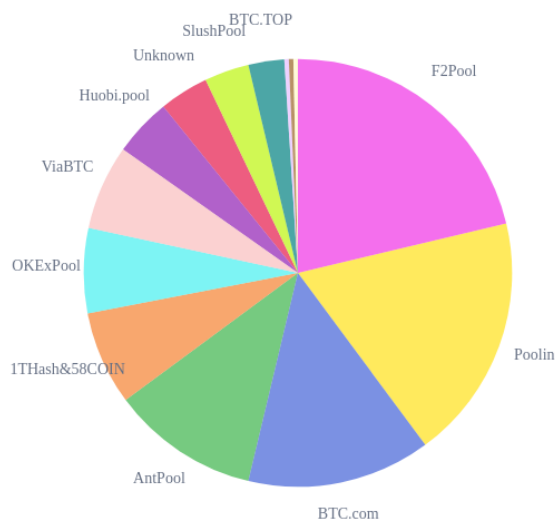


Figure 2.6: Summary of Mined Blocks as of 2020-03-30, per blockchain.com

Figure 2.6 shows the approximate split of total mined blocks in the past 48hrs since March 30th. Now, imagine a situation where organisations BTC.com, Poolin and F2Pool started collaborating, taking over 51% of the market. They would be able to add arbitrary blocks, self-verifying them - and since they hold the majority, nobody could oppose it.

2.2.3 Proof-of-Stake

A slightly different approach to verifying the transaction is called proof-of-stake, first introduced by [13] - which does not involve cryptographic puzzles nor requires high computational power and is all based on some pre-defined amount of cryptocurrency that is being put on hold, while delegates are selected. Every participant can bet any amount of crypto - which is returned to them after the validator is selected. The process can be outlined in the following steps:

1. Participants define their stake.
2. Network selects one participant that is going to be acting as a validator for the next block. The higher the stake, the higher the chance of getting selected. Losers get their stake back.
3. validator goes ahead and verifies the next block gets added to the chain, there is no block reward, although they receive network fees for the transaction.
4. After a couple of days (once other participants verify the transaction was not fraudulent), the stake is released and goes back to the validator.

This partially eliminates the issue of 51% attack mentioned above. First, because hash rate nor computational power no longer matters (and thus forming organisations loses the point) and secondly, even if the fraudster held more than half of the entire markets crypto, they would still be at risk of losing the stake, as their chance of getting selected as a validator is not 100%.

Sadly, this approach also is not free of any issues. Contrary to what I mentioned above, it creates a bias towards participant with more significant wealth, which can put more value on stake. This might create situations where rich get richer - though there is always a non-zero chance of getting selected. Furthermore, while they might not have malicious intents, they would be at a higher chance of getting selected as validator and thus collecting more network fees.

As this field is still expanding, more work is published, suggesting refined approaches. At the current day, both Bitcoin and Ethereum use Proof-of-Work, though the latter aims to move towards Proof-of-Stake in the future iterations [21].

2.2.4 Proof-of-Authority

However, what if we do not care about full decentralisation and want to avoid extra operational costs through proof-of-work or proof-of-stake algorithms? A simpler solution, called proof-of-authority [19] can also be used. This approach offers no rewards for adding new blocks to the chain, so it is not used in public blockchains. The validators are pre-selected and are responsible for vetting new blocks. This has more uses in situations where data does not have to remain secret, and we do not mind the lack of decentralisation. In fact, if the validators become compromised, they would be able to start allowing malicious blocks.

2.2.5 Ethereum

Proof-of-Work proved itself to be a tremendous waste of energy and resources, with Bitcoin using it solely for authorising the transaction and nothing beyond it. That particular period was also a time when a lot of different currencies started showing up, as Bitcoin's source code was open, everybody was allowed to host their network. Ethereum was one of them, but it was also the first to introduce a concept known as smart-contracts - a way to put the proof-of-work energy to some use (though still a lot of was wasted), introduced in 2015 by Buterin et al. [3]. The network also gave birth to so-called decentralised applications (or Dapps for short), through smart-contracts. Smart-contract can be understood as pieces of code which can get executed on the blockchain, written in a specialised language called Solidity inside EVM (Ethereum Virtual Machine). The transactions were no longer limited to the information about transferring currency between accounts, but also could execute code and act as a persistent database, which could not be altered by anyone and change history was publicly available.

2.3 IoT, Hyperledger and GA

Attempts at combining IoT authorisation with blockchain has been made in the past. One of the examples is a recent work by scientists from Khon Kaen University in Thailand. In that paper [14], researchers look into Authorization Architecture for IoT (using MQTT broker as an intermediary entity). They are arguing about the benefits of combining any solutions for low-power devices and distributed architecture, which ultimately enables much better scalability and removes the single point of failure.

They are also utilising Hyperledger Fabric - another blockchain-based ledger. Compared to Ethereum, Hyperledger [4] is used mostly for Business-to-Business scenarios, as it does not feature any reward for mining, i.e. adding extra blocks to the chain. Transparency is also limited, as the information is no longer placed on a publicly available platform but rather depends on trusting the nodes connect to the network although I will spend some more time discussing differences in implementation and discussion chapters of this paper.

Moreover, the focus of that paper is at finding optimised consensus algorithm, such that any latency caused by permission lookup is minimised. Scientists suggest using Genetic Algorithms to compose Optimal Consensus. Their experiments were executed on Kafka MQTT[22]. Thai researchers were able to achieve the performance of their solution called GA Kafka improved by 69.43% compared to standard Kafka.

Although this paper does not take into consideration other parts of the AAA framework, focusing solely on Authorisation. It has no mention of authenticating connecting clients or making them accountable by keeping audit crumbs of the most sensitive operations conducted on the chain. In my work, I will also be less focus on the performance of the blockchain network itself, leaving this down to the blockchain itself. As mentioned in the previous section, Ethereum has had some rapid movements in terms of improving their consensus algorithms and moving away from Proof-of-Work instead aiming to implement Proof-of-Stake.

Chapter 3

Requirements & Architecture

In this chapter, I will outline base requirements for the project along with sample stories that would later dictate the workflow. In the second part, I will also include an overview of the architecture proposed for the system, correlating each element with relevant requirement and explaining how they would address the use-cases.

3.1 Requirements

3.1.1 User stories

1. As a government regulator, I would like to overview access history to specific MQTT topics, to make sure the data is handled in GDPR-compliant manner.
2. As a government regulator, I would like to verify why / when / who accessed given resource at a specific time, such that I can issue fines for potential non-compliance and inspect data breaches.
3. As a topic owner, I would like to restrict people that can publish/subscribe to them, to maintain their confidentiality.
4. As a topic owner, I would like to collect payments from people willing to access my data.
5. As a topic owner, I would like to block access to my information from requests coming outside the requested country, to comply with GDPR requirements.
6. As a broker owner, I would like to collect payments from people willing to publish their data on my system, to keep the system profitable.
7. As a broker owner, I would like to secure a distributed network of brokers (with varied implementations), to increase the system's availability.
8. As a broker owner, I would like to block access to the system to malicious clients performing denial-of-service attacks, to avoid system downtime.
9. As a data consumer, I would like to publish/subscribe my messages on low-power devices, such that I can utilise my IoT sensors.
10. As a data consumer, I would like to access the broker from over a hundred parallel sensors, each publishing data independently.

3.1.2 Use-case Scenarios

Below I have listed a collection of stories which relate to real-life scenarios (though they are not real, any similarities are coincidental) and present various problems faced in different areas - both in industrial and commercial environments. FlyTrap is looking to address the issues surfaced in those short stories. In the Evaluation chapter of this paper, each of them will be referred to and tested on whether it is indeed helpful and how the discussed solution solves the problems.

3.1.2.1 Scenario #1: Air Quality study in the UK

Scenario: Robert is working as a Research Fellow at a University located in Manchester. The research aims at issuing air quality IoT sensors to staff across University, intending to capture information such as pollution or carbon dioxide level to analyse contents of air in the state. Each sensor is issued to an individual taking part in an experiment (e.g. member of staff, lecturer, PhD student), which is based in a specific room on campus. Robert needs to be able to track the inventory, and thus every sensor must be traceable down to a person.

The budget allows to issue up to 1000 sensors, and Robert would like to use MQTT broker to receive the data from the IoT devices. Additionally, he would like to share the dataset with researches across the country; thus, he makes the MQTT broker public. As per GDPR, such data, containing full name, office location and detailed temperature readings, is fully protected and needs to adhere to various governance requirements within the European Union. This information should also be stored only within the jurisdiction, that is, only within EU, to ensure enforceability of GDPR. Robert needs to make sure that only researchers that are located in Europe can access the data and everyone else (for example, people from the USA or Russia) will be denied access.

What problem is addressed here: Data containing personal information of European citizens needs to be handled in GDPR compliant way. This cannot be ensured if the information is accessible outside EU, as there would be no jurisdiction or enforceability once the data leaves the region. The person responsible for data storage needs to ensure that no people outside EU can access it.

3.1.2.2 Scenario #2: Data breach in an oil drilling facility

Scenario: Bob is a Chief Information Technology in a company Chell handling processing of oil and gas in Scotland. Bob's company also contracts many smaller companies which provide staffing and direct drilling services. Many sensors are used in the company, which are responsible for collecting data such as air pressure, humidity, occupancy on drilling platform or temperature. Those IoT sensors are utilising MQTT Broker, which is restricted only to authorised Chell employees.

Unfortunately, due to unrelated reasons, access to the broker has been compromised and thus allowing third parties to peek into the data flowing through potentially. Bob is approached by Judy, who works with the team responsible to trace the severity of leakage and determine the potential scope & damages.. Judy asks Bob to outline who might have had access to the leaked information and what the leaked information contained. Judy also instructs Bob to inform all people and contractors that might have been affected by the breach.

What problem is addressed here: Companies that use MQTT brokers to store information which are confidential and contain trade secrets of high commercial value want to determine the range

of leak that happened. This is part of disaster recovery procedure and is important for business continuity; i.e. determine how much of data has leaked and what kind of information was exposed to malicious actors.

3.1.2.3 Scenario #3: Unsatisfied Customer

Scenario: Mary recently purchased a smart assistant, which comes with several smart sensors to be placed around the house. Unfortunately, Mary decided to return all the sensors and cease further usage, she reaches out to Moogoo's representative - Matt.

Matt knows that Moogoo is using MQTT brokers to connect their smart sensors and then use the phone app to issue commands back to them through the broker. Although the phone app is not the only piece of software that has access to the data from the smart sensors. Analytics teams also consume those in order to help Moogoo create better products. Those servers have their own data storage capabilities and replicate the data that they consume. Matt is now tasked with identifying which internal analysis services might have accessed Mary's sensors in order to erase this information since it is on of GDPR requirements, also called "right to be forgotten".

What problem is addressed here: Again, GDPR comes into action here, in particular Article 17 - Right to erasure. Moogoo needs to permanently erase all trail coming from Mary's sensors, that includes any analytics datasets. Since those services are using MQTT brokers, there is no access trail and without proper infrastructure, impossible to go in the past and track which services were accessing the data.

3.1.2.4 Scenario #4: Information monetization

Scenario: Frank owns several farm fields in Scottish Highlands. As part of the ongoing digitalisation, he decided to purchase some IoT sensors to learn more about the environment his farms are located in. Some of the sensors include specialised soil acidity meters capable of measuring pH or determine how moist it is.

He decided to connect all sensors to one MQTT broker for ease of access and started to consume the information through provided mobile application. As the equipment was quite expensive, Frank would like to also sell access to the sensors to the interested parties, such as other land owners or University researchers looking to learn more about the soil quality in the area. No personal information is included, though the data is highly dependant on the location (which currently only Frank can access to) and thus might be also useful for other people.

What problem is addressed here: Selling data is a very lucrative business, especially if the data is valuable and has potential audience. MQTT does not have provisions to accommodate such requirements - anything like that would need to either be stored in some persistent location (and then processed for commercial purposes) or routed through extra layer of verification.

3.1.2.5 Scenario #5: Securing access to the broker

Bob works for a start-up company contracted by the local council to deploy smart sensors across the city to measure various statistics, such as air quality, traffic intensity, parking spot occupancy. All of those sensors are connected to the MQTT broker, which later can also be accessed by third

parties - providing they have a legitimate and approved reason. The MQTT broker is accessible by people that might not have overlapping permissions, for example company A can access only data set Y and company B can access only data set X - both X & Y are published on the same MQTT broker.

Bob is aware of the vanilla implementation of username/password authentication for MQTT brokers, but is concerned about the insufficient security in case of leakage. He also already is using Ethereum within the company to hold internal data, so can issue more wallets - if necessary.

What problem is addressed here: Security capabilities of vanilla MQTT Brokers can be found lackluster and not sufficient for many needs. Using symmetric password authentication can be very damaging if the said password leaks to third parties. For people that are already using related blockchain technologies, the transition to FlyTrap would be even simpler, as it can make use of any and every wallet compatible with ETH.

3.1.3 Functional Requirements

The user stories can then be further formulated into the following functional requirements:

- (FR1) The system will provide an interface to manage access to the topics along with inspecting the audit trails.
- (FR2) The system can connect to any Ethereum node, be it a public endpoint or a locally running, closed network. This will provide the flexibility of either using transparent and with 100% uptime resource or a closed node with reduced costs.
- (FR3) The system should provide a way to collect payments in ETH from clients attempting to gain access to relevant resources. This payment would then in the process, be transferred to the resource owner's Ethereum wallet.
- (FR4) The system should offer an option to specify an exact amount of ETH required to publish or subscribe - with the possibility of separating the costs and also setting the cost to 0 (=free).
- (FR5) The system should be capable of fending off primitive denial-of-service attacks by blocking continuous, failed attempts to connect.
- (FR6) The operations performed by clients will be of limited complexity, such that they can be executed on devices with limited computational power.
- (FR7) The system can answer crucial GDPR questions, such as who accessed given resource, why did they have access, when they accessed it and what exactly was accessed.
- (FR8) The system should offer an option to restrict the client's country that can access the resource, which will be verified using GeoIP lookup, as various countries have various data protection laws.

3.1.4 Non-functional Requirements

In addition to the functional, it is also vital to mention the following non-functional requirements, as the system is intended for end-users (potentially non-technical) and due to incorporation with blockchain can introduce performance overhead.

- (NFR1) The system should provide **an overhead of no more than 2 seconds cumulative** per MQTT session. This is important, as the intention is to provide an add-on on top of the existing MQTT brokers. This might further compromise the current efficiency, so the system should aim to minimise the added latency
- (NFR2) The system should be agnostic of the used MQTT broker, as long as the broker **fully implements MQTT v5.0 standard**. As pointed out earlier, there is a variety of brokers available to use, such as Mosquitto or Moquette. FlyTrap should not rely on the implementation of a broker, but rather only on the standard utilised.
- (NFR3) The system should be capable of extending any MQTT broker with **Authentication, Authorisation, Accountability** framework. This is to ensure that data can only be accessed by authenticated entities, which are authorised to access requested resources - and in case of a breach or other disaster, keep them accountable to their actions.
- (NFR4) The system should only be based on **Free and Open-Source Software**. Since the ultimate aim is to provide increase security, keeping the source open would allow any potential users to inspect its operation. Furthermore, third party security audits can happen without the system owner's intervention.
- (NFR5) The system should be capable to run inside **virtualised container**, to ensure that it's platform agnostic.

Chapter 4

Design

4.1 Secure Proxy

In order to enable FlyTrap to make decisions on whether the requests for publishing or subscribing should be accepted or denied, a secure proxy needs to be established between the clients and the MQTT Broker. As the communication between the broker and the consumers happens on Transport Control Layer, it is possible to insert a middleman who would be capable of inspecting the packets flowing through, dissecting it for relevant information and finally make a decision about their future journey - all without the client ever knowing that someone has intercepted the connection.

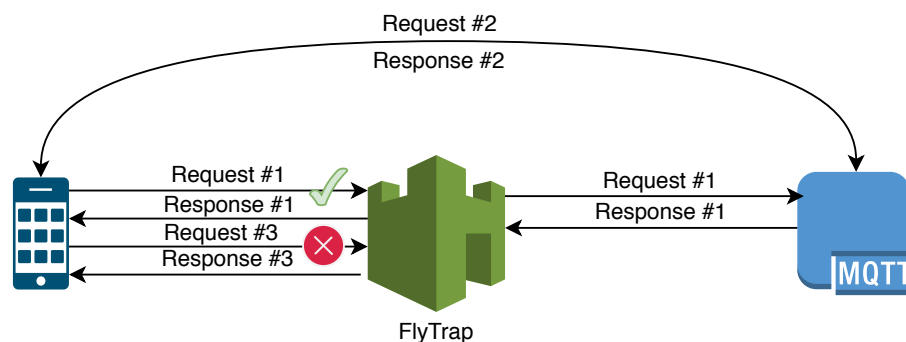


Figure 4.1: FlyTrap acting as a proxy

Figure 4.1 demonstrates all 3 possibilities when client attempts connection to a broker. In the Request #1, FlyTrap will dissect the packet and confirm that the phone indeed can be allowed to access specific topic and then start bidirectional proxy with the broker, passing the TCP packets between two. Request #2 shows that the same packet can be used for vanilla MQTT Broker without FlyTrap, thus decoupling the client and secure proxy, as the former can be used without the need to change the latter. Finally, for the third request, it is found that the client cannot access the requested resource and will be presented with CONACK response, with access denied flag set, terminating the connection.

Although this solution enough will not be sufficient, as quickly as FlyTrap can tap into the connection, the same can be assumed for potential malicious actors, which could be listening on the flowing through packets. The solution will support an extension to standard TCP - Transport Layer Security, or TLS for short, responsible for encrypting the TCP packets, significantly reducing the threat of man-in-the-middle attacks.

TLS sessions can be summarized in the following steps:

1. Initiate standard TCP session
2. ClientHello with client's cypher capabilities
3. ServerHello and exchange of the cypher suite, along with server's certificate
4. Key exchange and change of cypher spec
5. Encrypted session starts

It's important to point out, that due to step 3 requiring server's certificate, FlyTrap will need to either obtain a copy of broker's certificates or generate a new pair, ensuring that the connecting clients will trust it.

Chapter 5

Implementation

Chapter 6

Evaluation & Testing

Chapter 7

Discussion

Bibliography

- [1] Ashton, K. (1999). An introduction to the internet of things (iot). *RFID Journal*.
- [2] Banks, A., Briggs, E., Borgendale, K., and Gupta, R. (2019). Mqtt version 5.0. *OASIS Standard*.
- [3] Buterin, V. et al. (2014). Ethereum: A next-generation smart contract and decentralized application platform. URL <https://github.com/ethereum/wiki/wiki/%5BEnglish%5D-White-Paper>.
- [4] Cachin, C. et al. (2016). Architecture of the hyperledger blockchain fabric. In *Workshop on distributed cryptocurrencies and consensus ledgers*, volume 310, page 4.
- [5] California State Legislature (2018). Ab-375 privacy: personal information: businesses.
- [6] de Oliveira, D. L., Veloso, A. F. d. S., Sobral, J. V., Rabêlo, R. A., Rodrigues, J. J., and Solic, P. (2019). Performance evaluation of mqtt brokers in the internet of things for smart cities. In *2019 4th International Conference on Smart and Sustainable Technologies (SpliTech)*, pages 1–6. IEEE.
- [7] European Commission (2018). 2018 reform of eu data protection rules.
- [8] Evans, D. (2011). The internet of things: How the next evolution of the internet is changing everything. *CISCO white paper*, 1(2011):1–11.
- [9] Gervais, A., Karame, G. O., Wüst, K., Glykantzis, V., Ritzdorf, H., and Capkun, S. (2016). On the security and performance of proof of work blockchains. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 3–16.
- [10] Halperin, D., Heydt-Benjamin, T. S., Ransford, B., Clark, S. S., Defend, B., Morgan, W., Fu, K., Kohno, T., and Maisel, W. H. (2008). Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 129–142.
- [11] Index, B. E. C. (2020). Digiconomist.—[electronic resource]. *Mode of Access:* <https://digiconomist.net/bitcoin-energyconsumption>.
- [12] Jakobsson, M. and Juels, A. (1999). Proofs of work and bread pudding protocols. In *Secure information networks*, pages 258–272. Springer.
- [13] King, S. and Nadal, S. (2012). Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *self-published paper, August*, 19.
- [14] Klaokliang, N., Teawtim, P., Aimtongkham, P., So-In, C., and Niruntasukrat, A. (2018). A novel iot authorization architecture on hyperledger fabric with optimal consensus using genetic algorithm. In *2018 Seventh ICT International Student Project Conference (ICT-ISPC)*, pages 1–5. IEEE.
- [15] Light, R. (2017). Mosquitto: server and client implementation of the mqtt protocol. *Journal of Open Source Software*, 2(13):265.

- [16] Mijovic, S., Shehu, E., and Buratti, C. (2016). Comparing application layer protocols for the internet of things via experimentation. In *2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI)*, pages 1–5. IEEE.
- [17] Moore, G. E. et al. (1965). Cramming more components onto integrated circuits.
- [18] Nakamoto, S. et al. (2008). A peer-to-peer electronic cash system. *Bitcoin*.—URL: <https://bitcoin.org/bitcoin.pdf>.
- [19] Network, P. (2017). Proof of authority: consensus model with identity at stake.
- [20] Pultarova, T. (2016). Webcam hack shows vulnerability of connected devices. *Engineering & Technology*, 11(11):10–10.
- [21] Saleh, F. (2020). Blockchain without waste: Proof-of-stake. *Available at SSRN 3183935*.
- [22] Waehner, K. (2019). Iot and event streaming at scale with kafka & mqtt. <https://www.confluent.io/blog/iot-with-kafka-connect-mqtt-and-rest-proxy/>. Accessed: 2020-03-15.