Security in Smart Cities

Mike Castro Lundin

December 2016

1 Introduction

This report will analyse security concerns in smart cities. In particular, this report focuses on the implementation of a sensor network which gathers data from trash bins, communicates with a gateway, which then forwards the data to The Things Network (TTN), where all the data is merged and can be extracted into applications. The implementation described in [1] is taken as a starting point, but the model can be extended to support also controlling actuators remotely via a similar interface.

This report will identify the security/privacy issues in smart city applications such as these, considering how a potential violation in security goals can affect the companies behind these smart city solutions, cause economic damage or even put lives at risk.

It will also analyse how technologies such as TTN and the LoRaWAN protocols deal with working in a potentially hostile environments, to prevent, deter, detect and recover from security incidents. Specifically how they attempt to enforce confidentiality, integrity and availability.

The report will point out potential vulnerabilities in smart city applications using these technologies, as well as suggest counter measures that can be implemented for prevention, deterrence, detection and recovery.

Finally the report will suggest an access control model for extending the platform developed in [1] into a software as a service (SaaS) platform supporting any application that can fit within the established architecture.

2 General Smart City Architecture

The general architecture as used in [1] is shown in figure 1. There are multiple nodes which can be sensors or actuators, which connect via radio signals to a

nearby gateway. This is done using the LoRaWAN network protocol. The information is forwarded by the gateway to TTN which stores the data temporarily. Then an external application can using the MQTT messaging protocol receive data and send commands to the nodes.

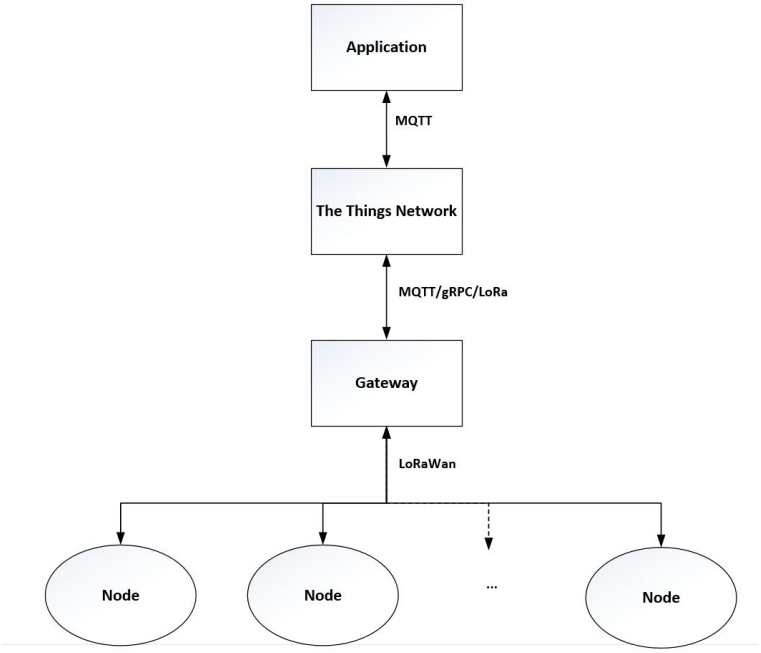

Figure 1: LoRaWAN/TTN smart city application architecture

In order for security and privacy concerns to be addressed, each block in the architecture must be analysed for vulnerabilities, as well as how the blocks connect to each other.

3 Vulnerabilities and Security Goals

The first step to analyse the security in such a system is to identify which are the security goals that are expected, as well as identifying vulnerabilities and their possible consequences.
The goals that this report proposes are:

1. Confidentiality: Data should only be available to the target nodes (f.e the data should only be available in the blocks that need it)
2. Integrity: Integrity of data must be verified, and unauthorised modification should be detected. Both origin and data integrity should be protected.

3. Availability: The system should implement mechanisms to deal with Denial-of-Service (DoS) attacks.

4. Accountability: Data should be signed by then node that generated, ensuring that at the top of the architecture, faulty nodes can be detected. This is goal is an extension to origin integrity

An additional consideration in terms of functionality is that the system should be able to securely add new nodes or reconnect nodes to the network without compromising security.

3.1 Vulnerabilities and threats

The main threats in a system like this are:

1. Deception: Hostile node being added to the network (adding hostile data), data being added using a compromised node, forging a node signature (identity deception), hostile gateways, hostile applications.

2. Disclosure: A violation in confidentiality, for example data being read by an unauthorised party

3. Disruption: DoS by flooding a gateway (hostile nodes connecting to a gateway), gateway Internet connection attacks, TTN website/service, saturating RF to avoid node communication (since they listen before speak).

The consequences of attacks are discussed in [8] for SCADA systems, so they have been slightly modified to apply to the system described in this report.

1. DoS: System downtime and loss of operations. If for example dealing with traffic control can cause traffic lights to not work, assuming no protocols are setup if for it to work independently (f.e switch every minute).

2. System files deletion in Gateway: System downtime and loss of operations. if the gateway does not work, nodes will be isolated, same applies to unavailability of the application or the TTN services.

3. Trojan/Key-logger to steam application authentication credentials: Hostile complete control of system. Can force undesired states on the system (f.e make all traffic lights red).

4. Data confidentiality lost: Loss of Corporate Competitive Advantage, privacy concerns.

5. Change data: Loss of data or even causing erroneous actions in the system

It should be clear that a violation of the security goals can lead to very severe consequences, and security should be taken very serious in many of the smart

city applications.

Specifically we must verify that nodes are reliably authenticated, and data must be sanitised/verified. Gateways must also be authenticated, and TTN must ensure that applications cannot get access to nodes they do not own. Also data should remain secret from the moment it is sent until it reaches its end destination. As mentioned before there must also be mechanisms in place to deal with DoS attacks.

The following sections will analyse each part of the architecture in figure 1, and show which measures are implemented to ensure the goals and prevent/deter the threats.

4 Security in the node level

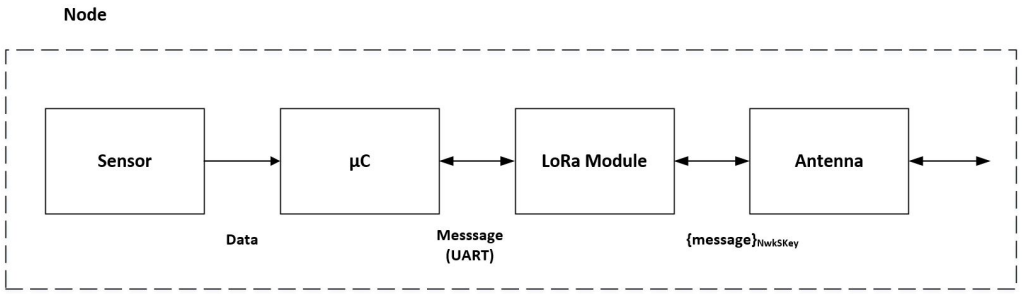

Figure 2: Node Architecture

The report will consider security in a node that has the architecture shown in figure 2. Data is gathered by the sensor and sent to a micro-controller (μC). The data is optionally processed into a message which is sent to the LoRa module, which handles the encryption and packaging of data. Finally the encrypted message is sent to the antenna so it can be broadcast to a gateway.

The main concern in the node is the confidentiality and integrity of data. Nodes can be located at hostile environments, and there are multiple possible attacks on such an architecture if the attacker has physical access to the hardware. This can include all of the attacks in figure 3, by replacing components or signals or just disconnecting the node. These attacks can be deterred by designing the hardware with this in mind, and detecting any unusual behaviour in the data being transmitted.

A more interesting attack would be for example carrying out side-channel analysis on a node to in the worst case extract the AppKey from the node. This would mean the node is completely compromised, since the attacker will be able to forge the nodes signature. Thus it is important that in the implementation of nodes, AppKey's are unique for each node (so only one node is compromised), and unusual behaviours such as repeated connection messages are detected, and the nodes are removed from the applications and blocked by gateways. More about AppKey and the LoRaWAN protocol will be described in next section.

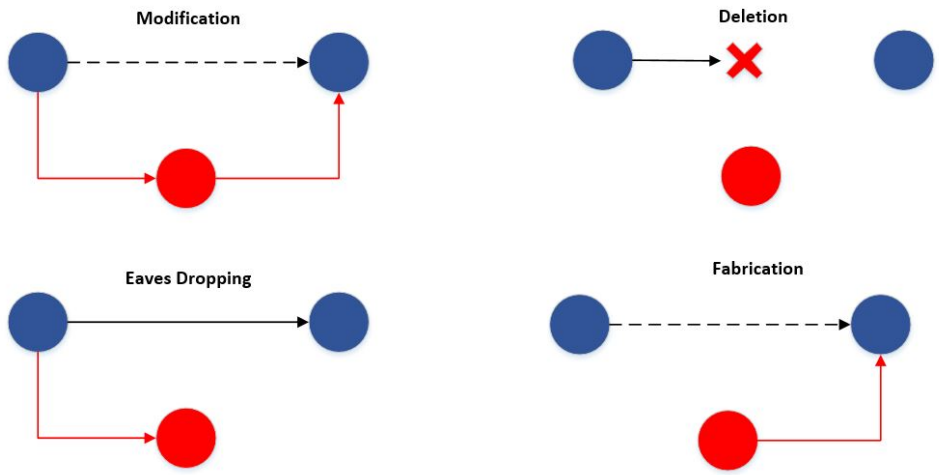

Figure 3: Attack Types

Thus to conclude the node analysis, since attackers can gain physical access to nodes in many applications, the hardware must be designed to deter the modification of the devices, and unusual behaviour in nodes should be detected and the relevant nodes should be blocked.

5 Security in the LoRaWAN protocol

The LoRaWAN protocol is used for communication between nodes and the gateway, as well as for enrolling/activating nodes. The description of the protocol is based on the LoRa specification [6]. The packet structure is shown in figure 4.

The whole packet is encrypted using NwkSKey (Network Session Key) and

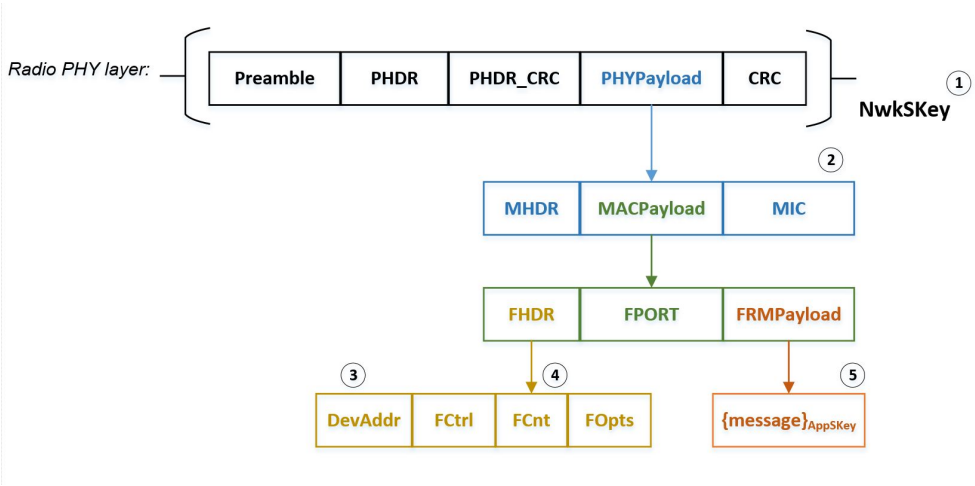

Figure 4: LoRa Packet structure

128-bit AES in counter mode (CTR). This key (marked 1 in figure 4) ensures confidentiality when sending a message from a node to a gateway, since it is known by these two parties (and the TTN broker, which is authenticated using its certificate, and is considered trusted by default). However one of the goals we established was that only the end-point should have access to data, and thus the gateway should have access to the LoRa packet except the message (FRMPayload, marked 5 in figure 4). This goal is ensured by encrypting the message using AppSKey which is not known by the gateway, allowing what is called federated network access by the specification, meaning the gateway cannot eavesdrop on the data, and thus it is somewhat safe to use a gateway that the solution does not own (authenticity is verified when enrolling/activating as described later, though after that, the gateway could potentially stop all messages unless there is some undocumented mechanism inbuilt in LoRa to ensure the gateway is actually forwarding the encrypted data, and not DoS attacking the nodes).

Confidentiality is ensured by cryptography, assuming the keys are secret, since nodes have a unique key for the gateway and a unique key for the TTN service, and authenticate each other using these keys. Generation of keys will be described and analysed later. Authentication of the gateway to the TTN service is done inside the gateway program, and thus is not described in this report, but assumed to be done correctly, and authentication of the TTN service is probably done using their certificate.

There are three mechanisms that prevent replay attacks, or reusing of old messages (and thus help ensure availability). The gateway can detect resent messages by looking at the FCnt bits (4 in figure 4), which should change from

message to message. Thus, even if the same message is sent, these bits should still change. This can be used as a mechanism for detection of strange behaviour. Similarly the gateway knows who is the original sender of the message by looking at the DevAddr bits (3 in figure 4), and thus only forward messages sent by nodes assigned to it by the authentication involved in node enrolment. The third mechanism is the use of the Message Integrity Code (MIC, 2 in figure 4), so that TTN can validate the message using the AppKey that only the node and TTN know. Inside MIC is the original message, so the application can verify it has not been changed.

In order to verify the assumption regarding keys being secret, both enrolment protocols are analysed:

5.1 Over-the-air Activation (OTAA)

OTAA is one of the protocols used by LoRaWAN to generate the NwkSKey and AppSKey, which will be used for one session. This way compromising of a key will only lead to loss in one node and only for one session. The protocol relies on the fact that the gateway has a secure authenticated session with TTN using TLS for example, and each node has common knowledge with TTN, an AppKey. A simplified version of the OTAA protocol is shown below, translated into AnB notation as taught in the data security course:

```
1  Protocol: OTAA (simplified)
2  # Communication between G and s goes through a TLS channel
3  #       where both are authenticated (sk(G, s))
4  # Removed the Transmission specification terms (f.e Rx delay),
5  #       thus does not describe full packet structure
6
7  # N is a node, G is a gateway, s is TTN
8  Types: Agent N,G,s;
9         Number AppEUI, DevEUI, DevNonce, MHDR, NetID, DevAddr,
10                AppNonce, pad, NetworkData, AppMessage;
11        Function sk,mac
12
13 #sk(N, s) is the AppKey
14 #sk(G, s) is TLS encryption
15 Knowledge: N: AppEUI, DevEUI, sk(N, s), mac;
16             G: s, mac, sk(G, s);
17             s: G, sk,DevAddr, pad
18
19       # unencrypted OTAA join request
20 N->G: AppEUI, DevEUI, DevNonce,
21       # Message Integrity Code (MIC)
22       mac(sk(N,s), MHDR, AppEUI, DevEUI, DevNonce)
23 # G asks s which is identified with AppEUI
```

```
24 G->s: mac(sk(G,s), AppEUI, DevEUI, DevNonce,
25 mac(sk(N,s), MHDR, AppEUI, DevEUI, DevNonce))
26 # If s recognizes DevEUI in the App
27 s->G: mac(sk(G,s), mac(sk(N,s), AppNonce, NetID, DevAddr),
28           # NwkSKey so gateway can communicate with node
29           mac(sk(N, s), 0x01, AppNonce, NetID, DevNonce, pad))
30 # N now has all information to generate AppSKey and NwkSKey
31 G->N: mac(sk(N,s), AppNonce, NetID, DevAddr)
32 # Example Communication: AppMessage end-to-end encrypted
33 N->G: {| NetworkData, {| AppMessage |}mac(sk(N, s), 0x02,
   AppNonce, NetID, DevNonce, pad) |}
34         mac(sk(N, s), 0x01, AppNonce, NetID, DevNonce, pad)
35 G->s: {| AppMessage |}
36         mac(sk(N, s), 0x02, AppNonce, NetID, DevNonce, pad)
```

Thus the node verifies that the gateway is authorised by receiving the a valid message encrypted using AppKey with the necessary information to generate the session keys (a nonce, network information and an address). The server verifies that DevNonce is not reused by remembering old values, to prevent any possible reusing. One concern is that in line 27, the encryption using sk(N,s) is done using ECB mode, which is vulnerable if the structure of the plaintext is known. This is why MIC is important and should be verified before the data is considered valid. A possible exploit is shown in figure 5 as specified in [5].

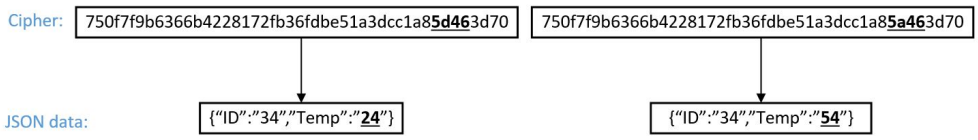

Figure 5: LoRa cipher vs data

This data could easily pass input sanitation, since they are both numbers, however the fact that it has been modified will be detected by MIC which acts as a CMAC.

In conclusion given the assumptions that the keys are secret, and the gateway and TTN are authenticated to each other (and are both trusted), this protocol will successfully create new keys for each node whenever required.

5.2 Activation by Personalisation (ABP)

This protocol is not described in depth in the specification. It is an activation protocol that instead of generating a new NwkSKey and AppSKey each session,

has a constant key which is only known by TTN and the node. Thus if a node uses a gateway, it will likely get the NwkSKey from TTN, and work after only one message if TTN recognises the node. The downside of this is that the keys will always be in memory in the node, vulnerable to attacks, and if the key is compromised, then the node will be hostile until it is reprogrammed with a new key.

6 Security in The Things Network (TTN)

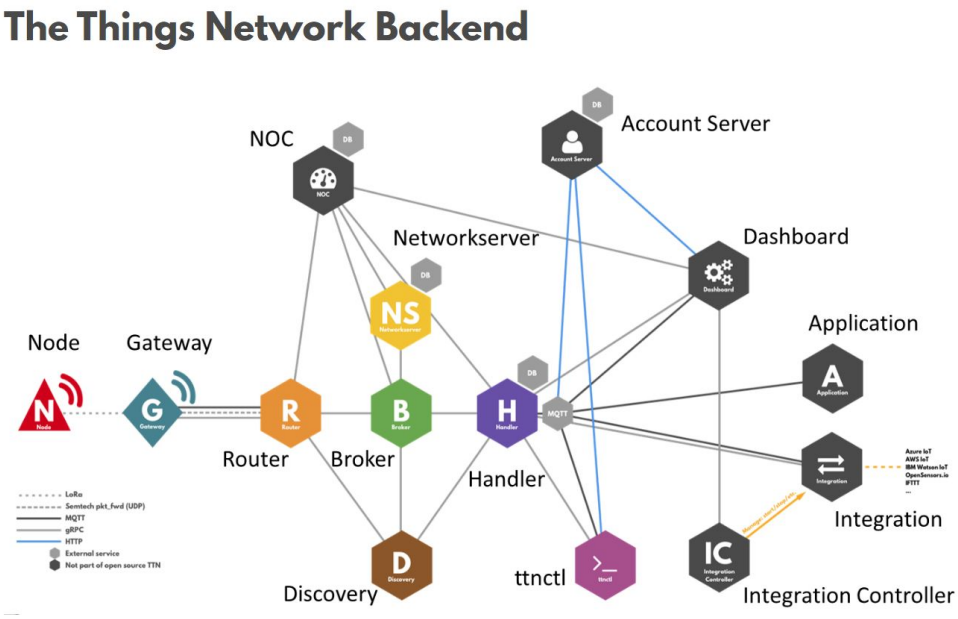

Figure 6: TTN architecture
(Source: https://www.thethingsnetwork.org/wiki/Backend/Home)

For this report we will assume that every node that is inside the TTN service (and thus out of the instance developers control) is safe. Safety in the node, transmitting data from the node to the gateway, as well as activating a node has been discussed in previous sections. Thus the remaining blocks and connections are the gateway, and the connections between the gateway and the router, as well as the connection from the handler to the application.

6.1 Gateway

The gateway is perhaps one of the most vulnerable parts of the system, due to the fact that it is an Internet facing component that does might not have

a lot of computationally power. The gateway is potentially also in a hostile enviroment, and vulnerable to similar attacks as a node, but it contains keys for every single node (NwkSKey), which if using ABP as disscussed before, will violate privacy by leaking metadata (all the information in figure 4 except the encrypted payload). It must be ensured that keys are only available for decrypting node messages, and not by any other process.

It is common that the gateways are for example a Raspberry Pi, and thus it is vital to deactivate remote access/control if possible, or ensure that it is done safely (for example using SSH). It is also important that the gateway runs only secure and updated software so that known exploits cannot be used on it. Further enhancements include firewalls that block all IP's except the TTN service.

6.2 Gateway-Router

The gateway connects to the router using software provided by TTN, and thus it is not publicly known which protocols are used or how they authenticate each other. As mentioned before, the router can be authenticated by the TTN service certificate, however authenticating the gateway, and identifying if it is a malicious gateway is the more complex task.

6.3 Handler-Application

The application is a user written program which receives the data from the nodes via TTN. The MQTT messaging protocol is used for communication. It is possible (and highly recommended) to implement MQTT over TLS, increasing the overhead, but thus authenticating TTN by its certificate. The application authenticates using a username and a password, and after successful authentication can receive the data sent by the nodes it owns.

This log-in is also used in the ttnctl service in figure 6 to add nodes to the application. Creating a new node returns a unique AppKey, so that it can be programmed into the node, thus avoiding duplicates. Even if someone would guess a valid 128 bit key, it would still require knowledge of the matching AppEUI and DevEUI to use it to forge a node.

Since the application is now authenticated, then TTN can perform access control, and only serve requests for nodes that the application owns (for example an access control list mapping user to nodes). The problem with the system is that many applications cannot read from the same node, thus an access control solution is required on top of the application to allow more complex access patterns (such as allowing multiple applications to access data from the same node). This will be described on the Application section.

The MQTT 3.1 specification [7] specifies threats and necessary mechanisms required, which are the following:

Threats: [7]

1. Devices could be compromised
2. Data at rest in Clients and Servers might be accessible
3. Protocol behaviors could have side effects (e.g. "timing attacks")
4. Denial of Service (DoS) attacks
5. Communications could be intercepted, altered, re-routed or disclosed
6. Injection of spoofed Control Packets

Mechanisms: [7]

1. Authentication of users and devices
2. Authorisation of access to Server resources
3. Integrity of MQTT Control Packets and application data contained therein
4. Privacy of MQTT Control Packets and application data contained therein

In the previous blocks of the architecture, detecting compromised devices has been discussed, as well as mechanisms to prevent it (cryptography, MAC...). Regarding data resting, this will only happen in the TTN servers and eventually in the application (depending on the developer). Timing attacks are possible in AES, for example analysing cache-timing [2] [4]. Counter measures for DoS attacks have been discussed both in terms of protocol mechanisms to detect replaying of messages, as well as filtering only valid traffic in the gateway. Interception and alteration of communications have been handled using cryptography and CMAC methods. Injections of spoofed packets have also been dealt by eliminating reused packages.

In terms of the mechanisms, every node authenticates to the other nodes by either 128-keys (node-gateway, node-TTN), user/password (application-TTN) or some not specified mechanism (gateway-TTN). Access authorisation is done by TTN, by associating nodes to applications with an ownership relation. The integrity and privacy of MQTT packets is ensured by TLS.

TTN implements client (node) to broker (TTN) encryption, but if End-to-End (E2E) encryption is desired, then the node should know an additional key, and encrypt the payload before sending it to the LoRa module. This key will have

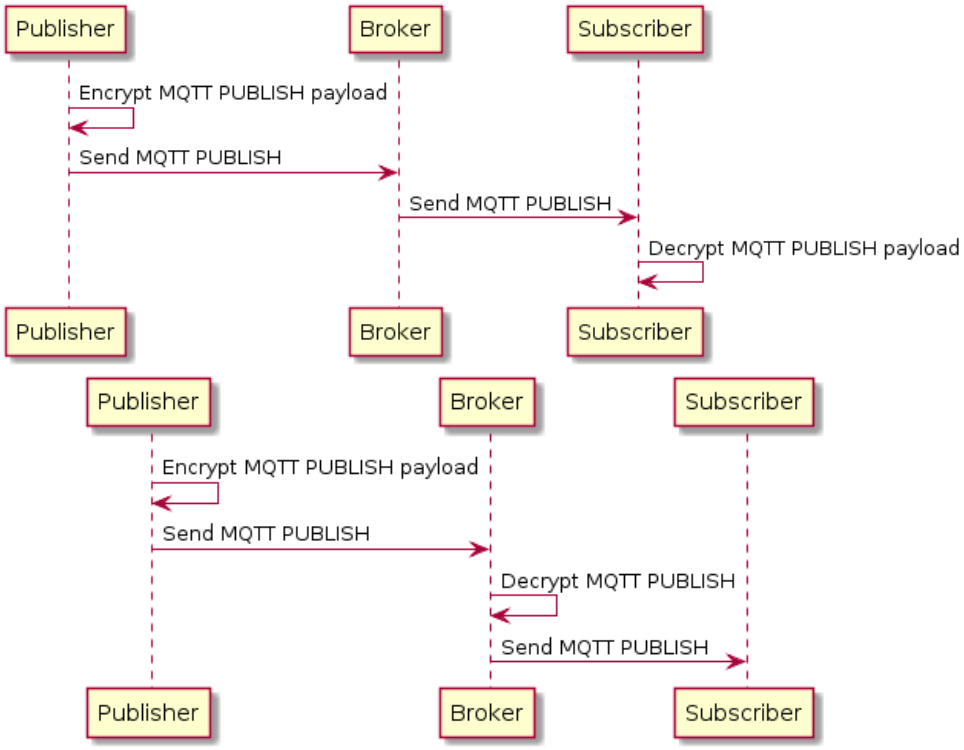

Figure 7: E2E and Client-to-Broker Encryption
(Source: http://www.hivemq.com/blog/mqtt-security-fundamentals-payload-encryption)

to be constant though, and thus vulnerable to the same exploits as the other keys in the node. These two encryption scenarios are shown in figure 7.

7 Security in the application

As discussed in chapter 6.3, the application is authenticated by the username and password. This report will now propose an authentication and access control model to build on top of an application, allowing it to become a generic SaaS platform.

7.1 Authentication

Allowing many users, requires that they can be identified and this identity must be verified in order to disclose data. Thus a username/password model could be

built on top of the application. This could follow the same design as suggested in the authentication report, and shown in figure 8. The print server would then be replaced with a data providing service, and the commands would be requesting data or sending commands instead of printing.

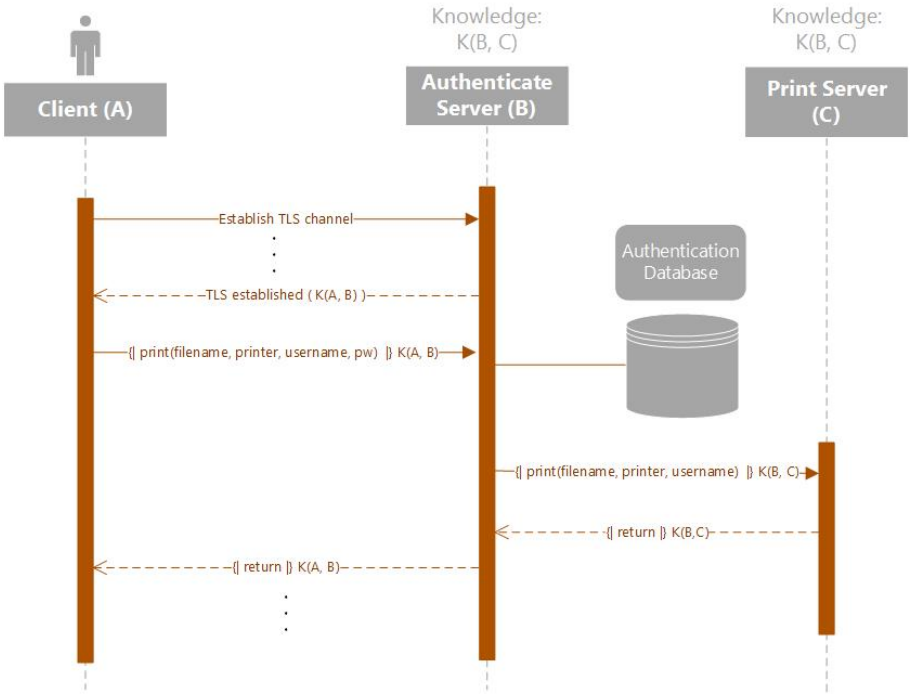

Figure 8: Authentication as sugested in report 2

Since different services might require access to data from different nodes, and we want to implement least privilege, then we will need some access authorisation mechanism. This can be done using access control lists as in figure 9 or using role based access control as in figure 10

Building such a system on top of TTN allows more complex models, where more than one client can have "ownership" of data. This flexibility comes at the cost of increased risk, by adding a new layer to the architecture, and thus a new attack point. Also storage of data in this system must be analysed and be implemented safely. Of course it also requires a system administrator to define policies.

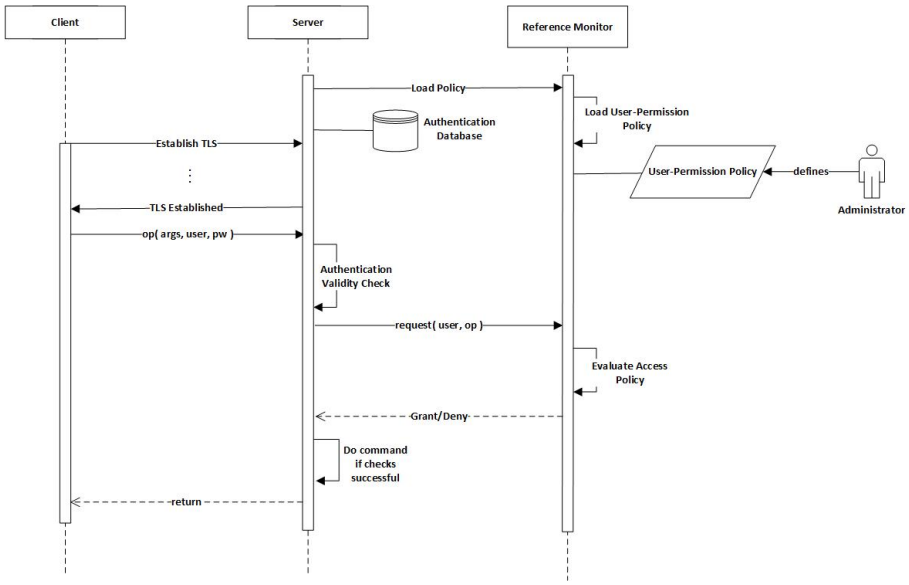

Figure 9: Access Control Lists in application

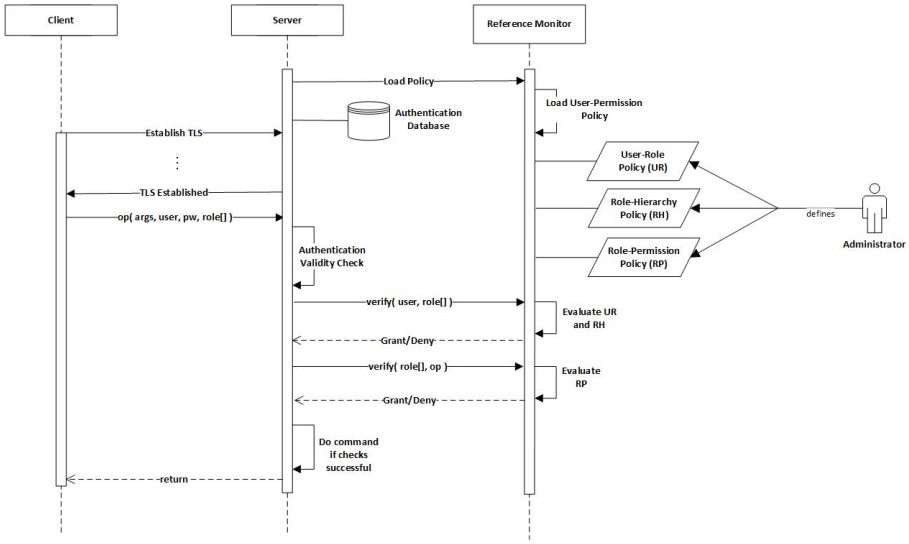

Figure 10: Role Based Access Control in application

8 Conclusion

To summarize the security of the analysed system we are interested in analysing confidentiality and integrity of data, availability of data and services and accountability to detect malicious or hijacked nodes and lock them out.

Some of the considered threats have been compromising of devices by side-channel analysis or unauthorised modification, data extraction at the node level, DoS attacks, modified/disclosed communications and injection of spoofed messages. These threats have been dealt with in relevant architecture levels, so for example data confidentiality (and thus communication disclosure) is ensured cryptographically at every level assuming the 128-bit keys are secret. The threats to extract the keys have been analysed, specially as a consequence of compromised devices. DoS and spoofed message injection are dealt at each communication link between architecture layers by either TLS connections, or using the MIC and FCnt bits in LoRa.

Threats on initialisation of the system have also been discussed, in particular vulnerabilities arising from the OTAA protocol.

Vulnerabilities in the gateway have been deterred by close control of the communication in the gateway from and to it via the Internet. Security in TTN has been either assumed to be correct because of lack of public disclosure of how protocols work, or is assumed to communicate safely via TLS and authenticated users.

Finally, a safe implementation of an extension of this system to allow more complex access control model has been suggested. Thus the application has been extended to be a SaaS platform.

It is evident from the findings that there are clear vulnerabilities in the system. The main ones found where using physical access to the nodes to infer confidential information, as well as possible weaknesses in the LoRa protocol because of ECB mode. Security in the gateway has been increased by taking counter measures, however ensuring that the gateway is completely invulnerable is a difficult task, and thus it is a matter of making intrusion hard, thus deterring attacks.

Some other angles of attack which have not been considered have been attacks on TTN, either the data they have (linking all applications to their devices and their AppKey, AppSKey and NwkSKey) or as a DoS. While nodes can use any gateway available, the TTN infrastructure seems to be the single-point of failure, since only it can receive data from gateways. The application level is not a single point of failure since any application that can authenticate can extract data.

Comparing the architecture to the SCADA model, it is possible to see that the addition of a hostile environment in the node/gateway level means that communication at every level must be assumed to be compromised, and thus requires extensive verifying. Authentication between layers is also required as an extension. Some vulnerabilities pointed out for SCADA systems are also very relevant for the architecture analysed in this report. The most important vulnerabilities are [3]:

1. Non rigorous access policies: If the TTN username/password is badly implemented in real life (f.e written in a paper in a public space in an office).
2. Rare or null update policies (security patches): Many parts of the system (in particular the gateway), require having the latest software to be safe.
3. Communication Protocol Vulnerabilities: Vulnerabilities in LoRa, the implementation of TLS or communication between the gateway and TTN.

Similarly, the countermeasures suggested are also very relevant [3]:

1. Firewalls: Blocking ports and IP addresses. Only use secure protocols. Can also use a DMZ.
2. System hardening: Removing unused software, disable all unused ports, use of removable devices restricted as much as possible...
3. Password policy: The TTN password and application passwords should follow proper password policies (f.e avoid most common passwords).
4. Software Management: Update to recover/avoid security issues.

References

[1] Jakob Schuldt-Jensen Andre Castro Lundin, Ali Gurcan Ozkil. Smart cities: A case study in waste monitoring and management. *Hawaii International Conference on System Sciences (HICSS)*, 50, 2017.

[2] Daniel J. Bernstein. Cache-timing attacks on aes. *Department of Mathematics, Statistics, and Computer Science, The University of Illinois at Chicago.*

[3] Igor Nai Fovino. *SCADA System Cyber Security*, pages 451–471. Springer New York, New York, NY, 2014.

[4] Anthony Tonge Mairéad O'Hanlon. Investigation of cache-timing attacks on aes. *School of Computing, Dublin City University.*

[5] Robert Miller. *LoRa Security. Building a Secure LoRa Solution.* MWR Labs.

[6] T. Eirich (IBM) T. Kramp (IBM) O.Hersent (Actility) N. Sornin (Semtech), M. Luis (Semtech). *LoRaWAN^{TM} Specification.* LoRa Alliance, 2015.

[7] Richard J Coppen Raphael J Cohn. *MQTT Version Version Version 3.1.1 Plus Errata 01.* OASIS Open, 2015.

[8] National Communications System. Supervisory control and data acquisition (scada) systems. *Technical Information Bulletin*, 04(1):42–45, 2004.