Project acronym: **IoTrust**

Project title: Secure trust bootstrapping and peer-to-peer network connections in

the Internet of Things

Third Party: **XXX**

[Insert LOGO of the Third Party]

Deliverable D.1

IoTrust Architecture Design

|  |  |
| --- | --- |
| **Deliverables leader:** | **Digital Worx GmbH, Germany** |
| **Authors:** | Rohit Bohara |
| **Due date:** | 31-12-2020 |
| **Actual submission date:** | 29-12-2020 |
| **Dissemination level:** | Public / confidential |

Abstract: It is a deliverable document designed and developed under the IoTrust project. It gives extensive information about the IoTrust architecture. This document will serve as a reference document for the future deliverables of the IoTrust.

**Disclaimer**

The sole responsibility for the content of this publication lies with the authors. It does not necessarily reflect the opinion of the European Commission. The European Commission is not responsible for any use that may be made of the information contained therein.

**Copyright**

This document may not be copied, reproduced, or modified in whole or in part for any purpose without written permission from the NGI Consortium. In addition, an acknowledgement of the authors of the document and all applicable portions of the copyright notice must be clearly referenced.

All rights reserved.

This document may change without notice.

**Table of contents**

[1 Introduction 1](#_Toc26358819)

[2 Activities carried out to complete the deliverable 1](#_Toc26358820)

[3 Technical description 1](#_Toc26358821)

[4 Conclusions and next steps 1](#_Toc26358822)

[Appendix 2](#_Toc26358823)

Nb: The deliverable structure below is only provided for guidance and you may adapt in a free form manner the structure to fit your needs.

# Introduction

The deliverable **D.1 IoTtrust Architecture Design** fulfils the objective **O1** which aims to design a human-centric and open IoTrust solution to increase the use trust and application of secure IoT networks in worldwide sectors like Smart Cities, Industry 4.0 etc. The deliverable D.1 is the output of the task **T.1 IoTrust Architecture Design**. The task T.1 was completed in the duration of month M1 to M6. The DW was the leader of the task. The milestone **MS2 Enhanced final version of IoTrust architecture** was achieved by D.1. The milestone MS2 is the advanced version of the MS1.

# Activities carried out to complete the deliverable

The user-centric requirement analysis was performed in the task T.1 to deliver deliverable D.1. It was an iterative process in which requirements of end users and other stockholders such as internet developers were taken in to consideration in designing the IoTrust architecture.

The task T.1 was performed based on Agile SCRUM[[1]](#endnote-31037) methodology. Each SCRUM sprint cycle was of 2 weeks. At the start of each sprint cycle requirements were gathered from end users and patterners. These requirements were analysed and an IoTrust architecture draft was designed and developed based on them. At the end of the cycle, this draft was verified and validated against the requirements. This process was performed iteratively throughout the lifecycle of the task T.1.

In a project like this, where a final product is shaped according to the end user and external stakeholders' requirements, some unforeseen issues might arise in the areas such as requirement gathering, changing and unclear requirements, functional requirements verification and validation criterion etc. These problems were identified and solved using the iterative SCRUM cycle before they could occur and hinder the project. The requirement gathering and analysis were continuous process like the designing the architecture.

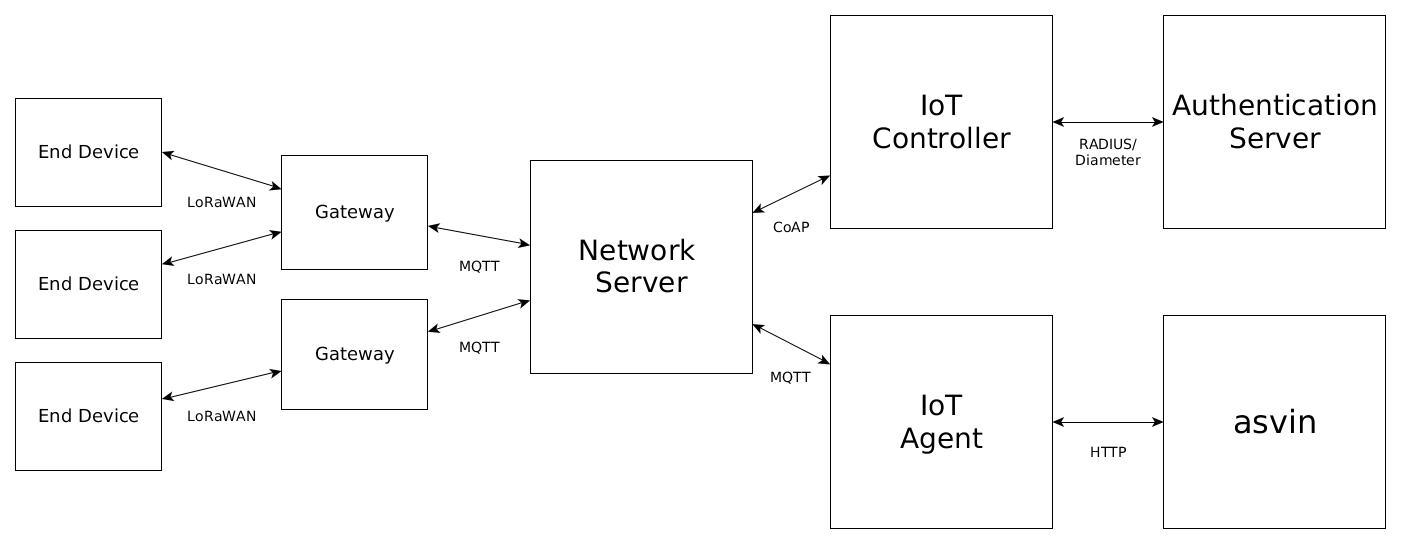
There were some unforeseen technical issues also addressed and fixed in the task T.1. The project is going to employ LoRaWAN[[2]](#endnote-1) protocol to send data packets between a LoRa[[3]](#endnote-2) node and gateway using radio communication in the 868 MHz ISM band. There are a large number of development boards available with different LoRa modems such as SX1276/77/78/79[[4]](#endnote-3) from [Semtech](https://www.semtech.com/), RFM95/96/97[[5]](#endnote-4) from [HopeRF](https://www.hoperf.com/), RN2483[[6]](#endnote-5) from [Microchip](https://www.microchip.com/) etc. We identified early enough that some libraries used for SX127X chips can send a packet of maximum 51 bytes which is not desirable for this project. It would have been a blocker in the project if we had not identified it at the start of the project.

# Technical description

Describe briefly the key technical characteristics of the deliverable and explain how they are related to the final results expected to be achieved by the project.

You can choose to include or annex relevant documents, mock-up, weblinks, screenshots, etc).

The core aim of the deliverable D.1 was to prepare an advanced IoTrust architecture design with an iterative process. There are many aspects to the architecture design. We have analysed and prepared it with the details of hardware, software stack, communication protocols, DevOps, user interface, customer experience, API end points etc. This architecture design will serve as a reference for further deliverables. Although the core attributes of the architecture will be the same, there might be some minor changes as we reach to the next milestones. The Figure 1 [fig ref] illustrates the overall IoTrust architecture.



The architecture components are described as follows.

## End-Device

It is a small form-factor hardware which sits on the edge of an IoT network. It consists of microcontroller, memory, input/output peripherals, communication protocol etc. These end-devices are put in to work for a specialized task. They are more suitable than the conventional computing devices for small, repetitive tasks because of their small form-factor and lower power consumption.

In the IoTrust architecture, an end-device will be used to collect, format and send sensor data to a server. It is paramount to authenticate an end-device before it connects to the server using a critical network. Because if the end-device is compromised than it opens the flood gate to the critical network infrastructure. The authentication, authorization and key management tasks will be performed by a secure bootstrapping protocol, peer to peer and distributed ledger technologies.

The Smart Everything (SME) Lion[[7]](#endnote-6) [[8]](#endnote-7) development board will be employed as an end-device for the IoTrust project. It is designed and developed by [Arrow](https://www.arrow.com/). It is packed with Atmel SAMD21[[9]](#endnote-28725) microcontroller based on the ARM Cortex M0+ architecture, Microchip RN2483 LoRaWAN module, Telit Jupiter SE868-A GPS module, Microchip RN4871[[10]](#endnote-20231) BLE module, Atmel AT24C256C 32Kx8 Bits EEProm and Atmel ATECC508A[[11]](#endnote-32483) crypto authentication chip. An end-device will use LoRaWAN protocol for communication. It will send LoRa packets using radio channels.

## Gateway

A gateway is the last component at the end of the LoRaWAN network infrastructure. This base-station serves as an intermediary between an end-device and network server. Basically, a gateway is a multi-channel high performance transceiver module which can receive, process and send several LoRa packets simultaneously using different spreading factors on various channels. An end-device will send data using LoRaWAN protocol. The LoRa packets will be received by all gateways in its proximity. It is often called LoRa gateway. It can handle LoRa packets from thousands of devices in the range of 1 to 10 kms.

The RHF2S208[[12]](#endnote-7083) will be utilized as a gateway. It is designed and developed by RisingHF. It comes with 4G LTE connectivity module, Ethernet, GPS, WiFi module, temperature monitor, RTC and power management unit. It also houses fully integrated 1 or 2 SX1301[[13]](#endnote-29609) LoRa core processor antenna with peak gain of 2.6dBi, and high performance ARM Cortex A53 microprocessor. The gateways and end-devices both will operate in the EU868 ISM band.

## Network Server

The Network Server is part of the LoRaWAN back-end infrastructure. It represents the central hub of all communications from and to LoRaWAN end-devices. It aims to hide the Physical (PHY) and Medium Access Control (MAC) layer details of the LoRaWAN protocol to the components that need to communicate with end-devices. The network server is in charge of collaborating with the end-devices to keep the overall network health, i.e., optimise the data-rate and overall energy consumption of the deployment site, as well as orchestrate what radio configuration parameters end-devices should employ in order to avoid packet loss or unnecessary retransmissions.

The IoTrust project will employ the ChirpStack.io open source LoRaWAN Network Server Stack [cite chirpstack]. This project is popular a Free Open-Source Software (FOSS) implementation of the LoRaWAN network server that provides several operation and administrative facilities in order to deploy a network of end-devices. All the components are licensed under the MIT license. Therefore, modifications and improvements can be made commercially available. Its architecture employs several operation and administrative end-points common in the IoT application scenario. These include, a web interface dashboard, standardised protocol event-based broker using MQTT [cite MQTT], and a REST[[14]](#endnote-3399) API over secure HTTPS connections. Therefore, its integration with other IoT libraries and networking components is relatively easy.

Overall, the LoRaWAN network server is a unique component. There is only one single instance per deployment and provides high-level abstraction of end-device communications. This is, applications and users are presented with a high-level abstraction end-point to send and receive messages to and from end-devices. These end-points may be a REST API, an MQTT broker or other customizable solutions. The network server will manage all the low-level details in order to guarantee secure and reliable delivery of messages to and from the LoRaWAN infrastructure.

## IoT Controller

The IoT Controller plays the role of authenticator in the Authentication, Authorisation, and Accounting (AAA) architecture (cite AAA). End-devices perform a bootstrapping process when they are deployed for the first time. This process includes an authentication and key agreement stage. The device credentials and ID information needs to be previously configured in an Authentication Server. While the end-device

Typically, end-devices transmitting over non-constrained networks perform the bootstrapping by directly addressing any authentication server connected to an IP network. This exchange usually employs a standardised protocol such as RADIUS or Diameter (cite RADIUS and Diameter) to carry Extended Authentication Protocol (EAP) (cite EAP) messages over regular IP networks. However, RADIUS and Diameter require an exchange of relatively large messages with a large number of transmissions. This only exacerbates the problem of energy consumption and radio bandwidth usage due to header overhead for constrained radio technologies such as LoRaWAN.

Therefore, a lightweight Low-Overhead EAP over CoAP (LO-CoAP-EAP) [cite LO-CoAP-EAP] protocol is chosen instead. LO-CoAP-EAP employs the novel Constrained Application Protocol (CoAP) [cite CoAP] and a set of efficient primitives to significatively reduce the header overhead of transmitting authentication EAP messages over a constrained network. The IoT Controller includes the LO-CoAP-EAP protocol logic that parses the upstream messages transmitted by the end-devices, and forwards its contents to an authentication server that employs typical AAA protocols such as RADIUS or Diameter to carry EAP payloads. Likewise, when the authentication server answers with the new downlink EAP messages, the IoT Controller generates a new LO-CoAP-EAP packet and forwards it to the end-device.

## Authentication Server

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris facilisis convallis viverra. Nunc sagittis non dolor id vestibulum. Sed diam mauris, pretium ac sollicitudin sed, feugiat in dui. Vestibulum bibendum neque eros, nec lobortis nisi finibus non. Donec viverra, dolor nec bibendum tincidunt, mi diam semper sapien, quis vestibulum dolor felis eu justo. Sed sollicitudin pharetra ipsum, et finibus libero fringilla nec. Mauris semper augue sed nibh rutrum, non rutrum turpis interdum. Sed at nibh id turpis varius tincidunt eget ut augue.

## Data Flow

* Include details about data exchange steps that take place among the different architecture components described above.
* Maybe include a figure.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris facilisis convallis viverra. Nunc sagittis non dolor id vestibulum. Sed diam mauris, pretium ac sollicitudin sed, feugiat in dui. Vestibulum bibendum neque eros, nec lobortis nisi finibus non. Donec viverra, dolor nec bibendum tincidunt, mi diam semper sapien, quis vestibulum dolor felis eu justo. Sed sollicitudin pharetra ipsum, et finibus libero fringilla nec. Mauris semper augue sed nibh rutrum, non rutrum turpis interdum. Sed at nibh id turpis varius tincidunt eget ut augue.

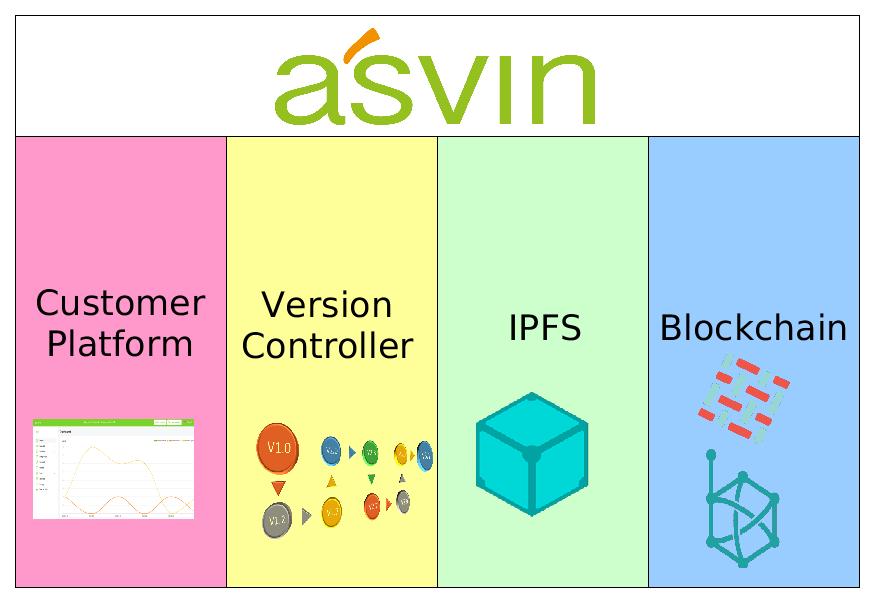
## IoT Agent

The IoT Agent is a MQTT client which subscribes to the topics exposed by the MQTT broker running in the Network Server. At the heart of the MQTT is the MQTT broker and clients. The data sent by the end-devices is received by the Network Server over LoRaWAN which is in turn dispatched using MQTT messages. Each message includes a topic. The IoT agent will subscribe to the topics to receive these messages. The topics will include device registration, device data, config data etc.

IoT Agent forwards the device metadata and sensor data to the asvin platform. It does it over HTTPs using REST API end-points. The IoT Agent acts as an agent between the network server and the asvin platform.

## asvin Platform

It is a Platform as a Service (PaaS) to facilitate over the air security patches for IoT devices using novel decentralized and distributed technologies. The asvin Platform[[15]](#endnote-14641) provides a complete solution for device, security patches and rollout management. It is comprised of 4 components as depicted the figure (figure ref).



Each component of the asvin platform is tailored to perform specific set of tasks efficiently. The Customer Platform provides an institutive web interface to give an overview and manage devices and their security patches. The Blockchain Server stores the critical device and security patches metadata information in a distributed ledger. The InterPlanetary FileSystem (IPFS)[[16]](#endnote-5368) server stores the security patches in a private peer-to-peer cluster. The Version Controller manages the information of new security patches rollout.

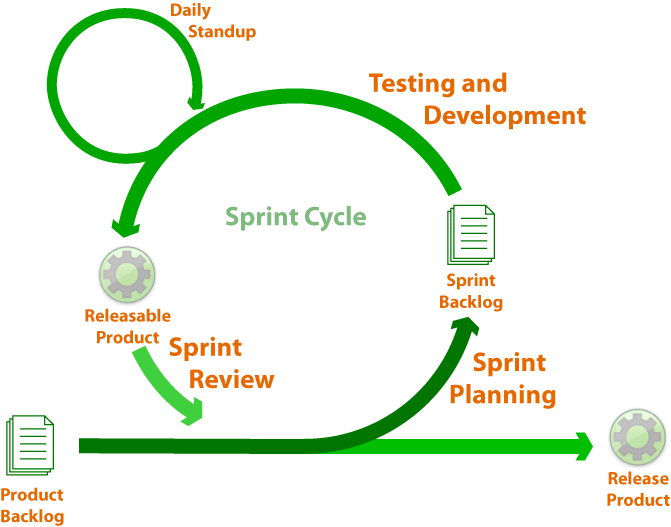
The asvin platform exposes its services using the REST API end-points. The other IoT services and platforms can interact with the asvin platform using these APIs. The IoT agent forward its data to the asvin platform using the respective API end-point. The asvin platform can send data to the network server directly as it also has REST API end-points or it can send data through the IoT agent.

# Conclusions and next steps

Outline any conclusions on the results achieved and any lessons learned for the next stage of the project.

Describe briefly the next steps in the project development and how you will build on this deliverable to complete the work.

The IoTtrust architecture is designed and submitted in the deliverable D.1. This deliverable gives comprehensive information about each component of the architecture. Every component of the IoTtrust architecture is designed after multiple rounds of discussions. It will be employed as a reference for the future deliverables.

[[17]](#footnote-13769)

Going ahead in the future, we will develop the components of the architecture using the SCRUM framework. We will plan one week sprint. At the start of the week, we will have a sprint planning meeting. It will include the tasks to be completed, acceptance criterion etc. We will complete the week with the sprint review meeting. All tasks are discussed in the sprint review meeting. The sprint review meeting includes the problems faced, solved and their status. The development backlogs are also examined. This process will be executed iteratively for the IoTrust development.

Appendix

References

1. https://www.scrumguides.org/scrum-guide.html [↑](#endnote-ref-31037)
2. https://lora-alliance.org/sites/default/files/2018-05/2015\_-\_lorawan\_specification\_1r0\_611\_1.pdf [↑](#endnote-ref-1)
3. http://wiki.lahoud.fr/lib/exe/fetch.php?media=an1200.22.pdf [↑](#endnote-ref-2)
4. https://www.mouser.com/datasheet/2/761/sx1276-1278113.pdf [↑](#endnote-ref-3)
5. https://www.hoperf.com/modules/lora/RFM95.html [↑](#endnote-ref-4)
6. http://ww1.microchip.com/downloads/en/devicedoc/50002346c.pdf [↑](#endnote-ref-5)
7. https://static6.arrow.com/aropdfconversion/5ff647cd30f423703234cbf85de7f2e794f2b199/smarteverythingasmelionuserguide.pdf [↑](#endnote-ref-6)
8. https://lorawan-hackathon.readthedocs.io/en/latest/lion.html [↑](#endnote-ref-7)
9. https://cdn.sparkfun.com/datasheets/Dev/Arduino/Boards/Atmel-42181-SAM-D21\_Datasheet.pdf [↑](#endnote-ref-28725)
10. https://ww1.microchip.com/downloads/en/DeviceDoc/RN4870-71-Bluetooth-Low-Energy-Module-Data-Sheet-DS50002489D.pdf [↑](#endnote-ref-20231)
11. https://ww1.microchip.com/downloads/en/DeviceDoc/20005928A.pdf [↑](#endnote-ref-32483)
12. https://fccid.io/2AJUZ-RHF2S208/User-Manual/User-Manual-4782465 [↑](#endnote-ref-7083)
13. https://www.mouser.com/datasheet/2/761/sx1301-1523429.pdf [↑](#endnote-ref-29609)
14. https://roy.gbiv.com/pubs/dissertation/top.htm [↑](#endnote-ref-3399)
15. https://asvin.readthedocs.io/en/latest/ [↑](#endnote-ref-14641)
16. https://ipfs.io/ipfs/QmR7GSQM93Cx5eAg6a6yRzNde1FQv7uL6X1o4k7zrJa3LX/ipfs.draft3.pdf [↑](#endnote-ref-5368)
17. https://carlleedrowsycat.wordpress.com/2014/09/28/what-is-an-agile-sprint-retrospective/ [↑](#footnote-ref-13769)