Software Requirement Specification

Muhammad Rameez

Table of Contents

2. Introduction

2.1. Purpose

This document is meant to serve as the first draft for the requirement specification of the Thesis project. It can be thought of as a summary of the brain storming session for requirements of the project. The requirements presented here are meant to be only indicative of what we want to build in a very abstract manner. This should not be taken as the final requirement specification document. The requirements presented in this document might not end up in the final requirement specification document. Some of these requirements might be too broad, others might be too narrow or might not be well defined at this stage. This document also presents some abstract design and implementation choices. These are presented as implementation alternatives after some requirements. The reason for discussing some design and implementation choices at such an early stage is to discuss and define the scope of the project as early as possible.

2.2. Scope

To be filled out later...

2.3. Definition and Acronyms

To be filled out later...

2.4. Overview

Provides overview of the remaining sections of this document. To be filled out later...

3. Overall Description

3.1. Product Perspective

The biggest portion of the system is the **Abstract IoT interface**. Our system has three main actors or entities namely IoT **sensor Nodes, Readers / smartphones and web portal**. Abstract IoT will be largest portion of the system and will encompass most of the actors depending on final requirements. IoT nodes will exchange sensor data with each other and with the readers. This data can be light, temperature and pressure data etc. The data will be communicated or transferred using secure protocol either Telehash or MQTT. Authorized readers or smart phones can request data from nodes using functions provided in the IoT interface. IoT nodes and their data are commodities which can be leased by external or internal actors. IoT interface provides methods for metering and selling usage or data. This is realized using micro transactions and payement channels such as described in raiden or perun networks. Conflict resolution is left to the blockchain. Web portal can use the IoT interface to gather data about the devices in the company's network e.g. number of devices, their state i.e. offline or online, state of micro transactions, balance etc.

3.2. Big Picture decisions

Before we move on to discussing individual requirements we need to discuss and settle some big picture design questions. These design questions are essential for defining achievable requirements. We will discuss these design decisions below along with pros and cons of each decision

- Autonomous sensor running smart contracts themselves vs Smart contracts on cellphones / Special nodes.
- Hybrid both autonomous and dumb sensor nodes

3.2.1. Smart Contracts on IoT nodes

This is the paradigm employed by Filament they have specialized javascript virtual machine running on their IoT nodes or taps which runs smart contracts on to the IoT nodes themselves. The devices are initialized with an initial smart contract at the time of manufacturing, this smart contract includes amongst other things a customer-specific community name which lets it discover other devices in customer's radio network and allows it to have a shared context. For public community networks such as smart cities they use proprietary block-chain based systems such as Blockstack, namecoin and Blockname. Further information about Filaments solution is given in I**nitializing devices with smart contracts section of** [1].

Benefits

This approach has some obvious benefits sensor nodes can be truly autonomous and are able to function in both public and private networks with minimum extra effort. They are easier to deploy and maintain over a longer period of time.

Prerequisites

However, having such a solution will require certain prerequisite requirements which need to be met. The IoT nodes themselves need to be able to run smart contracts this means either we need specialized nodes or smart contracts need to be ported to the existing hardware nodes. Both of these solutions will require significant time and effort in my opinion. Furthermore, some other requirements need to be met like tamper resistant firmware and firmware updates. Filament solution for making firmware of their taps tamper resistant is given in **Security Begins with Hardware** section of [1]. Another important requirement for this design is to have sensor data disclosed to only authorized devices such as tablets and smartphones. Filament uses a technique in which cryptographic credentials of an IoT device can be chained back to a root in this manner they determine authorized devices such as tablets or smart phones which are able to read data off of the IoT node. Read further **Access control** in [1]. Some other requirements that must be met are summarized below

- Tamper resistant firmware
- Access Control / Authorized devices only

3.2.2. Smart Contracts in Special nodes

A second solution would be to have two types of nodes basic nodes and special nodes. Basic nodes share or log sensor data. Sensor data is stored in local flash of basic nodes and is encrypted at rest and during transfer to special nodes. Special nodes take data from basic nodes and run smart contracts and mircotransactions. Please note that special nodes can be more then smart phones these could be an evolved form of basic nodes i.e. **Telos B** nodes etc. We need to do further research in order to determine if something like TelosB or Waspmote can meet my project requirements. For now, I am only considering special nodes to be smart phones or tablets etc. This approach like the first one has some prerequisites and benefits.

Benefits

We won't need specialized hardware for nodes. This design can be achieved using generic off the shelf components. Most of if not all of the design could be implemented within software.

Prerequisites

Some basic prerequisites for this approach are

- Access control / Authorized special nodes only
- Encrypting sensor data at rest and in flight
- Data from reprogrammed basic nodes should be rejected by special nodes and network portal
- In case of TelosB or Waspmote etc. special node should be able to run Smart contracts easily

3.3. Micro transactions in Special Nodes

In order to make IoT nodes autonomous economic actors they should be able to perform micro transactions with each other for e.g. a car can pay for parking itself. Filament uses something close to a bitcoin wallet on the device to enable micro transactions read **microtransactions** section of [1]. Their technique builds on Json Tokens with zero knowledge proofs. They argue payment channels with both parties signing the same transaction might be too expensive for resource constrained IoT devices. I think if memory and power requirements can be dealt with i.e. for always plugged in devices payment channels can work and for truly mobile devices we might need to look into zero knowledge proof or some other solution.

4. Specific Requirements

In this section I present some high level abstract requirements for different parts of our system.

4.1. Interface Requirements

4.1.1. Nodes

Nodes should be able to interface with each other and with organization web portal securely. Interface means that they communicate and exchange data and information. Data can be anything i.e. logged sensor data, micro transaction data i.e. state, keys, signing information etc. All exchange of information should be encrypted and achieved using some secure protocol such as MQTT or TELEHASH.

4.1.2. Web Portal

Organizations should be easily able to deploy a monitoring web portal for all the IoT nodes in its network. IoT nodes should have a mechanism to securely expose their transaction history and operational status to an organizations central portal i.e. all IoT devices read some special seed for the portal and present transaction and value data to the portal whenever possible. How this data is presented to the portal is largely dependents on implementation and organizational primitives i.e. could be with the help of special nodes i.e. cell phones that present data each time they read a basic node. Alternatively, special nodes automatically send data whenever a

basic node comes into contact with them which is new or whose last read data has exceeded some pre-determined threshold.

4.2. Communications interfaces

Communication between nodes can be achieved using one or more of the following Bluetooth, Wi-Fi, RFID /radio scans, NFC etc. Of course the actual transfer of data should be secure I.e. achieved using a secure protocol such as TELEHASH irrespective of the underlying transport medium used.

4.3. IoT Abstract Interface functional requirements

This section will mostly deal with functional requirements concerning the abstract IoT interface we are building.

4.3.1. Functional Requirement 1.1

ID: FR1

Title: Node Discovery

Desc: IoT nodes should be able to discover each other. IoT abstract interface should define methods and classes for nodes to discover each other securely.

Design alternatives: TELEHASH has native-local discovery. Alternatively, this can also be achieved using organizations running notaries or using something like an Ethereum naming service.

4.3.2. Functional Requirement 1.2

ID: FR2

Title: Data Sensing

Desc: IoT nodes should be able to sense and log sensor data. IoT interface should support methods for sensing temperature, pressure, and light data.

4.3.3. Functional Requirement 1.3

ID: FR3

Title: Data Encryption

Desc: IoT nodes should be able log sensor data. IoT interface should define methods for encrypting sensor data on flash memory at rest and during transfer between nodes.

4.3.4. Functional Requirement 1.4

ID: FR4

Title: Data Transfer

Desc: IoT nodes should be able to transfer data with each other and with special authorized personal. IoT interface should provide methods for securely accomplishing such data transfers. This will require IoT interface to use some secure protocol such as TELEHASH or modify other protocol such as MQTT.

4.3.5. Functional Requirement 1.5

ID: FR5

Title: Smart Contracts

Desc: IoT interface should support methods for enabling smart contracts in the system. Smart contracts can be run either directly on special nodes. Alternatively, Smarty contracts can be enabled using a web portal or central server which collects information and data from nodes.

4.3.6. Functional Requirement 1.6

ID: FR6

Title: Microtransactions

Desc: IoT interface should have methods for allowing micro transactions on the nodes or in the system.

4.3.7. Functional Requirement 1.7

ID: FR7

Title: Microtransaction Definition

Desc: Microtransaction Definition includes

- Rate of commodity to be sold
- Authorized entities who are allowed to transact with the node
- Transaction Context e.g. under which conditions this transaction allows access to commodity e.g. power access for 1 hour, location data for specific or full part of the journey, pressure and temperature data for specific operational timings.
- Microtransaction context and rate cannot be updated after they have been created.

4.3.8. Functional Requirement 1.8

ID: FR8

Title: Device Identity

Desc: Each IoT node should have its own unique cryptographic identity

4.3.9. Functional Requirement 1.9

ID: FR9

Title: Device Identity Management

Desc: Devices can be assigned Identities in groups or individually. The identities can be updated at any time. Only very special authorized entity whose identity is cryptographically proven should be able to assign or update device identities.

4.3.10.Functional Requirement 1.10

ID: FR10

Title: Device Identity

Desc: Each IoT node should have its own unique cryptographic identity

4.3.11.Functional Requirement 1.11

ID: FR10

Title: Authorized Entity

Desc: IoT interface should have a mechanism to define authorized entities who are permitted to use nodes.

4.3.12.Functional Requirement 1.12

ID: FR12

Title: Communicating Authorized Entity information with nodes

Desc: IoT interface should have a mechanism for communicating authorized entity information with all nodes in the system, this can be done either one time during start or a dynamic communication mechanism should be derived for exchanging this information regularly amongst nodes.

4.3.13.Functional Requirement 1.13

ID: FR13

Title: Device-Entity Relationship Context

Desc: IoT Node access is key and this access is the primary feature which can be sold. Micro transaction allows owners to sell this access however this should not be done in all or nothing basis. Relationship between entities and devices can be used to determine a context which is used to sell specific access. One context allows entity A to read pressure data from all nodes in the network another allows Entity B to have full access to all even numbered devices.

4.3.14.Functional Requirement 1.14

ID: FR14

Title: Updating Device-Entity Relationship Context

Desc: IoT interface should define function to update relationship context updates. These updates should only work in case no running smart contract is being effected or prematurely terminated by the update.

4.3.15.Functional Requirement 1.15

ID: FR15

Title: Auto update of micro transactions and Relationship Context

Desc: Preprogramed sell features for data and access should be removed, renewed or updated automatically once the smart contract related to them has finished or funds in the contract exhausted.

4.3.16.Functional Requirement 1.16

ID: FR16

Title: Monitoring primitives

Desc: IoT interface should define monitoring primitives for companies wishing to deploy the system i.e. define a new monitoring task and its primitives e.g. define a class called project, tracking etc with primitives such as add sensor node to this project, remove node, display data. The Project primitive is always associated with one or more smart contract and tracking primitive is tasked with bringing data into the smart contract.

4.3.17.Functional Requirement 1.17

ID: FR17

Title: Encryption keys

Desc: IoT nodes will have different encryption keys for encrypting data, signing transactions etc. Should there be a mechanism to update these keys either manually or automatically? If so should this be done automatically with each new monitoring task or project the node is added to? Is it sufficient to have static keys for nodes.?

4.3.18.Functional Requirement 1.18

ID: FR18

Title: Data and Access Categorization

Desc: should nodes have different data categories for data they expose to external entities? Data in public category can be read by all i.e. either by paying for it or for free, private data can be read by authorized personal/devices only i.e. through using some security mechanism. This also includes categories for node access i.e. special nodes can sell access to network or internet etc.

5. Design choices based on FR5 and FR6

Basic nodes job is just to report data to smart contracts on the block chain or the network either continuously or at fixed intervals i.e. weather when shipping changes hands etc. FR5 and FR6 lead to two different design choices which need to be discussed. It is important that we clarify which path needs to be taken in an earlier stage. This is assuming we don't use nodes which are capable of running smart contracts and micro transactions themselves.

Data Presented to Smart Contract at fixed Intervals

sensors need to log data and securely communicate the data to the smart contract before changing hands in case of our shipping analogy at which point smart contract goes through entire logged data to determine if a violation has been made. This requires that as soon as the current /primary responsible party in the system changes data is fed to the smart contract either using special nodes or fixed point machines at handover points.

Smart Contract only takes complete activity data

This will be useful for use cases where we can clearly define current activity end points or points where the current context switches. Continuing with our shipping analogy before each hand off the data portion is signed cryptographically with next shippers key. The data is presented to the smart contract in case of violation or after final delivery. Smart contract can then trivially determine who was responsible for violations based on cryptographic keys which serve as locks and classifier on portions of data.

6. Open Questions

6.1. Protocol choice

6.1.1. TELEHASH vs MQTT

Telehash is transport agnostic, supports end to end security and node discovery. I haven't looked into detail of MQTT yet also require more research into both these protocols. I suspect refined and more precise requirements will help in choosing the right protocol.

6.2. Node Hardware

So far haven't done much research in to this. I have very briefly looked at the following. I need to do further detail research into sensor Hardware and what is available and I suspect refined and more precise requirements will help me narrow and focus my research.

6.2.1. Waspmote

It is presented in [3].

Advantages:

- Off the shelf solution,
- supports encryption,
- ope source sdk,
- big sensor stack,
- supports multiple communication premitives XBee 802.15.4 • XBee 868MHz • XBee 900MHz • XBee Digimesh • XBee ZigBee • LoRa. 1 year claimed bettery timing without recharging

Disadvantages:

- Only supports certain encryption libraries
- not sure if allows to program custom encryption solutions,
- safe recommended operating temperature of -10 to 50c

6.2.2. TelosB Sensors

I haven't looked into this so far apart from reading material presented in [4] and [5]. I need to do further in depth research into the capabilities of these sensors.

6.2.3. Rasberry Pi based solution

Have a look at [6] and [7]

Advantages:

- very modular,
- open source,
- good community support,
- can build almost anything

Disadvantage:

- lots of manual work in programming and assembling components,
- may detract from the main project of focusing on block chain

7. Use Cases

Selling Sensor data

IoT nodes can sell their sensor data to authorized entities e.g. pressure, light, temperature and location data can be sold. IoT interface will allow owners to easily define pricing, authorization policies, and policies for data they want to sell or use.

GPIO Metering

Since all MCUs have general Input and output. IoT interface can provide some general pins for connecting external devices to our nodes. These will enable node owners to sell different services such as power, wifi access etc. Service connected to these pins is not important what is important is that customers can easily define rates at which these pins supply access and calculate payments in the form of micro transactions e.g. a power outlet can be connected to one of the pins. The node owner has the ability to define charging rates for power delivery either one off rate for continuous use i.e. 0.1 ether for say 24-hour period or define rate based on load or time etc. The user who wishes to use or draw power will use his smart phone to request access, a smart contract will be negotiated between user and device. IoT device will calculate fee based on payment interval / method and rate of the service offered connected to the I/O.

8. References

[1] https://filament.com/assets/downloads/Filament%20Security.pdf

[2] https://filament.com/assets/downloads/Filament%20Foundations.pdf

[3] http://www.libelium.com/products/waspmote/overview/

[4] https://www.advanticsys.com/shop/mtmcm5000msp-p-14.html

[5] https://telosbsensors.wordpress.com/

[6] https://tutorials-raspberrypi.com/raspberry-pi-sensors-overview-50-important-components/#wireless

[7] https://www.linux.com/news/21-open-source-projects-IoT