University of Zurich^UZH

Inventorying and Secure Life-Cycles of IoT Devices

Armin Richard Veres
Zürich, Switzerland
Student ID: 20-700-118

Supervisor: Dr. Eryk Schiller
Date of Submission: December 1, 2023

BACHELOR THESIS – Communication Systems Group, Prof. Dr. Burkhard Stiller

University of Zurich
Department of Informatics (IFI)
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland

ifi

Bachelor Thesis
Communication Systems Group (CSG)
Department of Informatics (IFI)
University of Zurich
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland
URL: http://www.csg.uzh.ch/

Abstract

Das ist die Kurzfassung...

Acknowledgments

Optional

Contents

Chapter 1

Introduction

1.1 Motivation

There are daily more and more Internet of Things, IoT, devices connected to the internet, with the need to gather and process massive amounts of real-time information, especially with 5th Generation networking, which allows for extensive information exchange.

Enhanced connectivity and adoption of IoT trigger cyber attacks, which are increasingly sophisticated and affect considerable amount of IoT-related infrastructures, raising security concerns with consumers, as well as businesses. This stresses the importance of appropriate IoT security management and enhancement of IoT life-cycle management. Considering the heterogeneity of IoT devices, the dynamism of the security landscape and number of IoT stakeholders make these tasks quite challenging, especially considering that a single insecure update can put a whole IoT system at risk.

As emphasized by the European Union Agency for Cybersecurity, ENISA, Cyber Security Act, CSA, the management of IoT infrastructures encompassing the entire life-cycle of products, as well as the continuous certification, are fundamental tools to guarantee a high level of security. [1] Also as pointed out by the Network and Information Security, NIS2, directive a pragmatic security framework must stimulate active collaboration between the IoT Stakeholders. [2]

Providing access to cybersecurity information is central for realization of a homogeneous perspective on cybersecurity. CSA and NIS promote strategic cooperation among stakeholders to support and facilitate information sharing, leading to an approach that helps respond to large-scale incident by creating more effective synergies against cybersecurity vulnerabilities.

1.2 Description of Work

This thesis will develop a service to support security information sharing between IoT stakeholders to support continuous security assessment throughout the IoT device lifecycle. We will consider the use of Distributed Ledger Technology, DLT, as a possible approach to facilitate a trustworthy and transparent platform for sharing cybersecurity information without a trusted third-party. [3] It is important to integrate the presence of several entities with different responsibilities and roles in sharing cybersecurity knowledge. So while performing security monitoring activities, the user, i.e., a device, may detect vulnerabilities that will be shared with the manufacturer for further investigation, prompting for mitigation and or resolutions. This requires the device to be re-configurable throughout its life-cycle according to the changing threat landscape and to the device manufacturers publishing of secure updates, i.e., patches, and device profiles.

Established approaches of secure IoT deployment and bootstrapping have significant challenges. Using pre-shared credentials for every device is the simplest approach, but it prevents the identification of specific devices and the verification, whether the device is corrected to the correct network. [4] This thesis will develop a bootstrapping service to provide a lightweight bootstrapping protocol, supporting different authentication methods, depending on the characteristics of the device and providing key management.

The infrastructure of the developed bootstrapping mechanism will enable the inventorying of IoT devices. Said infrastructure will keep track of all embedded IoT devices and their respective security levels. To ensure security throughout the life-cycle of a device, an update/patching mechanism will be developed, where manufacturers and software providers will provide fixes to a security issue after an attack or vulnerability detection.

As most updating proposals are based on centralized models using e.g., client-server architectures, this thesis will design a scalable and secure approach for disseminating software updates in scenarios with selected IoT devices. The design shall entail decentralization, robustness and efficiency, bringing the upgrading functionality closer to the end devices. Blockchain based technologies will be leveraged by providing a transparent ledger to manage different versions of software elements composing and IoT device or system and share relevant security aspects, such as vulnerabilities or device information. As interoperability is crucial, this thesis will analyze the use of Bifröst [5] and Interledger [6] approaches to interconnect different blockchain implementations.

Finally this thesis will consider mitigation for IoT devices using Threat Manufacturer Usage Description, MUD, proposed by NIST [7], which provides a flexible and dynamic way to alert about new threats and mitigation to apply before and update of patch is released. Threat MUD is intended as a complement to MUD file, dynamically reconfiguring a device in case of detection of a vulnerability.

1.3 Thesis Outline

This thesis has two main emphasis, Inventorying and Secure Firmware Updating.

IoT devices need to able to be uniquely identified ...

Chapter 2

Background

2.1 Distributed Ledger Technology

Distributed Ledger Technology, DLT, refers to the technology of the decentralized database, which provides control over the development of data between various entities through a peer-2-peer network. In order to achieve synchronicity across nodes of the network, consensus algorithms are used.

2.1.1 Blockchains

As defined by [8] a blockchain is a distributed data structuring organizing a growing list of transaction records into a chronologically linked sequence of data blocks, which is achieved by a decentralized peer-2-peer network, consisting of a cluster of participating nodes. Blocks are continuously appended to the blockchain through the *mining* process, executed by distrustful peers. The mining process includes the verification of broadcast transactions to specific nodes in the peer-2-peer network. Through a consensus mechanism the blocks are conjoined to the blockchain. The result is a chain of blocks that ensures the integrity of existing blocks, leading to a DLT without a need for a central authority.

There is a major distinction between *permissioned* and *permissionless* blockchains. Everyone may join a permissionless blockchain, given in some cases they fulfill certain requirements, in contrast to permissioned blockchains, where only approved members may join. [9] This requires a central authority to fulfill this step, but given some environments, such as enterprise, it is necessary, see Section 3.4.2.1.

Permissioned networks boast improved performance over permissionless on on e.g., transaction verification, as only approved members are in the network. To offset the absence of trust, in permissionless blockchains, typically mined cryptocurrencies and or transaction fees. Therefore in permissioned the cost of using a fault-tolerant consensus mechanism.

On that note a blockchain is a subset of the DLT. [8]

2.1.2 Smart Contract

Further defined by [8] a smart contract specifies a computer transaction protocol which is based on the execution of terms of the contract upon fulfillment of some pre-conceived condition(s). Relating back to blockchains, the smart contract is a part of some block, that is stored and verified there in the form of code until execution. Its state consists of the contracts balance and an internal storage, that is updated on each invocation of the contract. A user can send a transaction to the contract address, which triggers the state transition of the contract and the data being written to the internal storage. A smart contract may perform a predefined logic and also interact with other accounts by sending messages or transferring funds.

Most smart contracts apply the *order-and-execute* architecture, in which the consensus protocol first validates and orders transactions, then propagates it to its peer nodes and finally each peer executes the transaction sequentially. The order-and-execute architecture requires deterministic execution, otherwise no consensus consensus can be reached. To address non-determinism usually a Domain Specific Language, DSL, is used, such as *Solidity*.

2.1.3 Consensus Protocols

TODO CFT (crash fault-tolerant) or BFT (byzantine fault-tolerant)

2.2 Verifiable Credentials and Identity Management

2.2.1 Verified Credentials

In contrast to identification through physical means, such as passports and driver licenses, the *Verifiable Credential, VC*, model tries to achieve similar portability, alas in a digital wallet, be it on a phone or other edge device. [10]

Digital Identity is the digital reference to a person or subject [11], with which it/they in response to requests for digital identification, authentication or proofs of authentication is presented. Also of importance is that there is a unique identifier, that is connected to the digital identity. [12] In general identities come with *identifiers*, such as names, social security number, mobile number, date of birth etc. [13].

Support for verifiable credentials and digital wallets has been growing, e.g., in Canada [14], where Canada's Verifiable Organizations Network, VON, permits storage of credentials inside Hyperledger Aries [15]. VON allows the issuing of digital licenses, permits and registrations to legal entities using VC. The European Blockchain Services Infrastructure also uses VCs to issue documents from public institutions, such as digital diplomas and social security passes. [16]

An example VC Document can be seen in Code Listing 2.

2.2.2 Decentralized Identifiers

Decentralized Identifiers, DIDs, not to be confused with Decentralized Identities, are often used by decentralized identity protocols. [17] They are not issued by a centralized body and are stored on DLTs or on peer-2-peer networks, which makes DIDs globally unique, resolvable with high quality and cryptographically verifiable. [18] For example, an Ethereum account in itself is a valid DID and one can create as many of them as one wishes. [13].

According to [13] two main components make DIDs possible:

- **Public Key Infrastructure (PKI)**: Using a public and private key for an entity, one can authenticate user identities e.g., in a blockchain network such ash Ethereum and prove digital ownership of assets. The private key can decrypt and or signs, while the public key verifies or encodes.
- **Decentralized Datastores**: the blockchain as verifiable data registry is open, trustless and a decentralized repository of information and through the existence of public blockchains the need to store identifiers in centralized registries is eliminated.

Each DID is made up of three elements separated by colons as seen below [17]:

$$\underbrace{\text{did}}_{\text{Scheme}} : \underbrace{\text{method}}_{\text{Method}} : \underbrace{\text{1234567890abcdefg}}_{\text{Method-Specific Identifier}}$$

The scheme is fixed to be DID, while the method summaries and describes how the DID works with a specific blockchain and the specifications of DID deployment. Finally the method-specific identifier guarantees to be unique within the context of the DID method.

A DID document can be of format JSON or JSON-LD, which is a JSON based format to represented linked data [17]. An example can be seen in Code Listing 3. There are further optional identifiers and keys in such a document explained in depth in the W3C specification [17]. Mainly it consists out of the identifiers, the DID, which is required, a **controller** that may dispose over it and aliases through **alsoKnownAs**. The next section entails **Verification Methods**, which may list multiple ways of verifying interactions with the DID and other associated parties. The third Section **Verification Relationships** describes the relationship between a DID and its verification method(s). Finally the **Services** part describes what endpoints there are for the DID to interact or advertise, including decentralized identity management services for further discovery, authentication, authorization, or interaction.

Although DIDs are integral to managing decentralized identities, they are not enough on their own, as they do not provide meaningful identity attributes. In order to establish a trust-relationship between DID entities, the above mentioned VCs prove useful. [10]

2.2.2.1 DIDComm

TODO

2.3 Manufacture Usage Description

MUD has been developed by the International Engineering Task Force, IETF, with following goals and intents in mind: [19]

- Substantially reduce the threat surface on a device to those communications intended by the manufacturer.
- Provide a means to scale network policies to the ever-increasing number of types of devices in the network.
- Provide a means to address at least some vulnerabilities in a way that is faster than the time it might take to update systems. This will be particularly true for systems that are no longer supported.
- Keep the cost of implementation of such a system to the bare minimum.
- Provide a means of extensibility for manufacturers to express other device capabilities or requirements.

MUD does not entail address network authorization of general purpose computers, it simply creates a suggestion than can be followed. The architecture of Devices using MUD can be seen in Figure 2.1, which is the reference architecture by NIST [7] but it can be found in similar fashion inside the RFC specification.

2.4 Physical Unclonable Function

In order to be able to track IoT nodes in a blockchain, they need to be uniquely identifiable, in our case even in a distributed manner, using e.g., Distributed Identifiers, DIDs. Common practices are based on placing a cryptographic key into a nonvolatile electrically erasable programmable read-only memory (EEPROM) or battery-backed static random-access memory (SRAM) and use hardware cryptographic operations such as digital signatures or encryption, which is all expensive in design and power consumption. [20]

PUFs are unpredictable and uncontrollable, therefore making it unclonable and an ideal security vector. They are dependent on random physical factors introduced during manufacturing, e.g., inequalities of SRAM cells, although factors such as the altering of the physical components, voltage and temperature need to be taken into account. [21] For reasons of simplicity and because it is not the main focus of this thesis, we will neglect this aspect.

By implementing the Challenge-Response Pair, CRP, is used to evaluate the microstructure, whereas a physical challenge makes the device react, the response, in an unpredictable, but repeatable way. In order to turn this 'silicon key' into a cryptographic root key, processing algorithms need to be applied, that ensure that the distribution of 0s and 1s are uniform. [20]

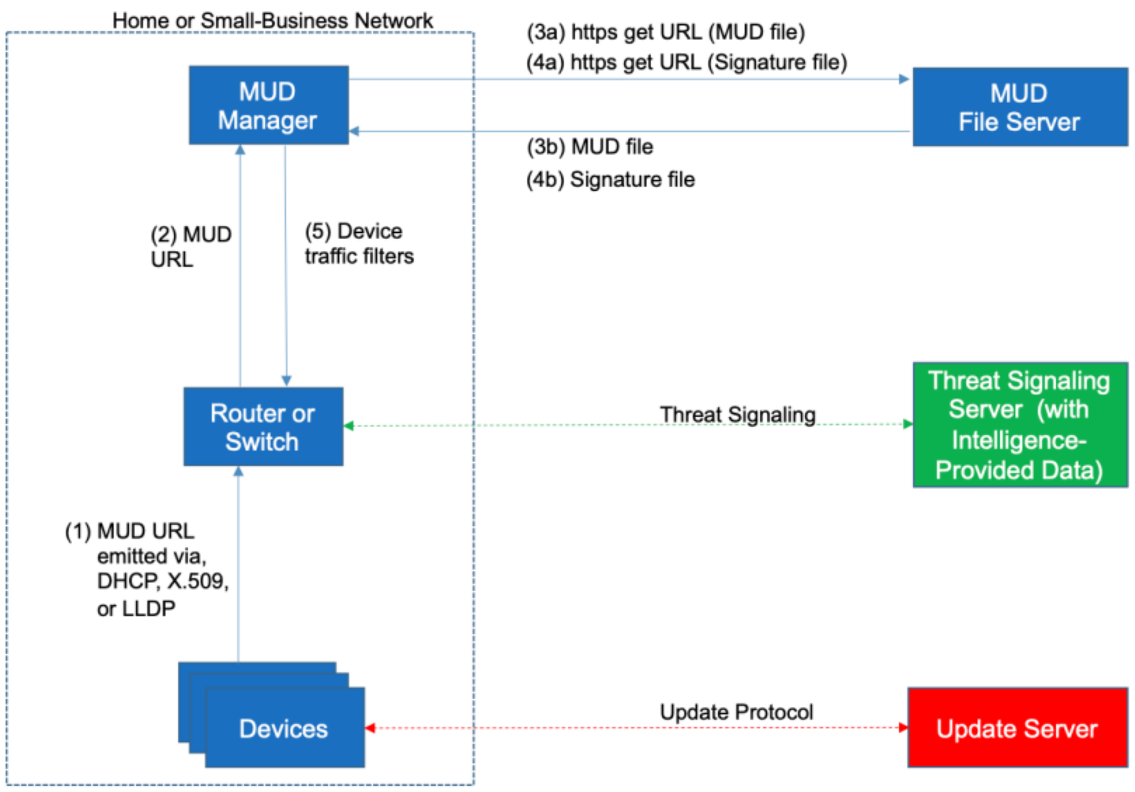
Home or Small-Business Network
MUD Manager
(3a) https get URL (MUD file)
(4a) https get URL (Signature file)
MUD File Server
(3b) MUD file
(4b) Signature file
(2) MUD URL
(5) Device traffic filters
Router or Switch
Threat Signaling
Threat Signaling Server (with Intelligence-Provided Data)
(1) MUD URL emitted via, DHCP, X.509, or LLDP
Devices
Update Protocol
Update Server

Figure 2.1: NIST MUD Reference Architecture

2.4.1 SRAM-Based PUF Readouts

Methods of creating identifiers that are unique to devices exist, such as SRAM-Based Physical Unclonable Function, PUF, readouts. Therein PUFs are among the most cost-effective security primitives to establish hardware trust. [22]

Even though the evaluation process of the characterization of guarantee over lifetime and differing operating conditions are still subject to research following metrics have become widespread: *reliability*, the variation of bit-wise startup patterns; *uniformity*, i.e., the repeatability and reproducibility on a given device after any amount of restarts; *uniqueness*, the probability of other devices with same signatures; *bit-aliasing*, the probability of specific bit position of the signature to be biased towards 0 or 1. [21]

2.5 Over the Air IoT updating

In order to stray away from the classic client-server architecture for updating devices, which demonstrate a single point of failure, we will discuss other decentralized methods to achieve Over the Air, OTA, updates for IoT devices.

2.5.1 Distributed OTA

TODO

2.6 Networking

TODO

2.6.1 Software-Defined Networking

TODO

2.6.2 Network Functions Virtualization

TODO

Chapter 3

Related Work

3.1 CERTIFY

This thesis is carried out in conjunction with the CERTIFY project [23].

The National Institute for Standard and Technology has a few ongoing projects and white papers on security related mitigation methods for IoT devices.

3.2 DLT-based Asset-Tracking

[3] analyzed how blockchain-based approaches might be used for data accountability and provenance tracking under the then recently released GDPR legislation, highlighting challenges of scalability and considering sharding as a method to address it. [3] Further they also mentioned issues of clonability of the tracked assets, which we can also correlate to the physical assets that are tracked inside blockchain.

3.3 Cybersecurity of IoT Devices

In order to maintain participation rights for only valid users/clients, Manufacturer Usage Descriptions, MUDs, are getting more and more relevant, as also the National Institute for Standards and Technology, NIST, have been considering their use cases. [7]

3.3.1 SRAM-based PUF Readouts

For this thesis we will not be implementing a sophisticated Designated Accrediting Authority, DAA, leading to the assumption, that it will be implemented as part of another thesis. For our use-case we will refer to simple hardcoded authentication, with a key being provided for each device by us and trusted by us for simplicity.

Nonetheless the topic is still relevant from a cybersecurity perspective and there have been a few attempts at creating secure keys out of SRAM readouts, such as [21] or [24]. We will also be considering using SRAM-based PUF readouts in our device configurations in order to get DIDs that are absolutely unique for each device.

3.3.2 Device Fingerprinting

For classification of device capabilities NIST has been considering the usage of MUDs, so that devices do not step out the bounds of their official and appointed capabilities. [7]

3.4 Decentralized Identity Management Technologies

In the quest for the right technologies, we are only considering open source projects, so that we may extend frameworks and applications with modules to fit out needs, which why there may be many options not listed in here.

3.4.1 Decentralized Identity and Access Management for IoT (DIAM-IoT) Framework, 2020)

There has been a similar work in the field of our thesis, which has been made by [8], who investigated decentralized Identity and Access Management, IAM. They proposed a framework, DIAM-IoT, which incorporates DIDs and VCs into the lifecycle of IoT devices, while building it up on blockchain as a bridge to connect IoT data silos.

They identified DID and global key-value databases as integral parts for building a self-sovereign identity system.

Their onboarding process includes the creation of a cryptocurrency wallet for manufacturers, through which each has to stake some of the cryptocurrency tokens to deploy a smart contract for a new device. Those with poor products are punished this way.

In contrast to our thesis, they refrained from using MUDs to specify device capabilities and designated each user with their DID which is bound to a device that has a DID as well. Their use case is consumer oriented. Each user gets a mobile application that can be used to do the device registration. A VC is generated by an IoT-Cloud after a successful device binding (binding the user and a device). See Figure 3.1.

The DIAM-IoT framework is blockchain agnostic, although they implemented it in the paper through the IoTeX Framework, mentioned in Section 3.5. They mention scalability in their paper, there is no hard data presented. The IoTeX documentation mentions scalability.

Table 3.1: Comparison of Identity Management and Verified Credentials related Software

Technology Name	Short Description	Main Features
Hyperledger Aries [15]	creating, transmitting and storing verifiable digital credentials, (uses Indy per default, should be pluggable though)	• VC issuing • Blockchain-agnostic
Hyperledger Indy [15]	Tools/Libraries for providing digital IDs rooted on Blockchains or DLTs	• public read access • Distributed Ledger for DID Management
Hyperledger Fabric [25]	Framework to glue together blockchain, IdM and other DLT services to manage IoT devices, also supporting smart contracts; permissioned access	• Distributed (Blockchain) OS
IoTeX [26]	Framework including blockchain to manage IoT devices	Does not fit our needs as it is permissionless
Sovrin	***TODO***	***TODO***
IBM Platform	***TODO***, works on top of Hyperledger Fabric	***TODO***

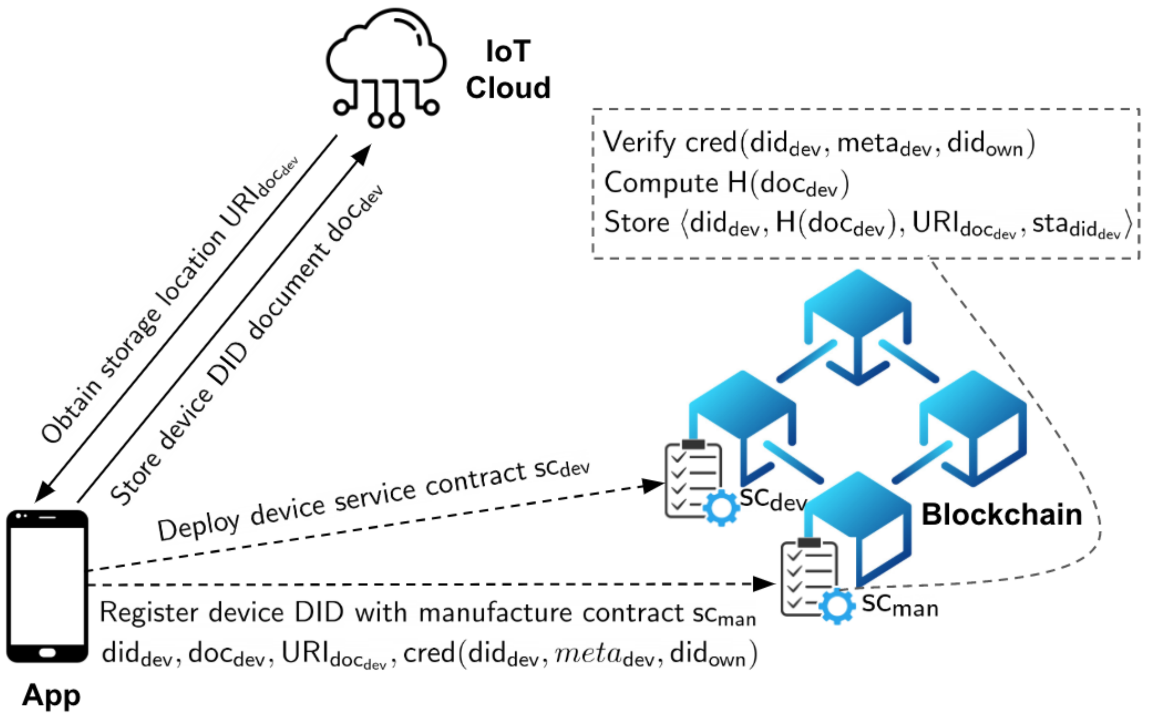
IoT Cloud
Verify cred(did_dev, meta_dev, did_own)
Compute H(doc_dev)
Store ⟨did_dev, H(doc_dev), URI_doc_dev, sta_did_dev⟩
Obtain storage location URI_doc_dev
Store device DID document doc_dev
Deploy device service contract sc_dev
sc_dev
Blockchain
Register device DID with manufacture contract sc_man
did_dev, doc_dev, URI_doc_dev, cred(did_dev, meta_dev, did_own)
sc_man
App

Figure 3.1: DIAM-IoT DID Registration Process [8]

3.4.2 Hyperledger

The Hyperledger Foundation, run by the Linux Foundation, provides a great many technologies related to blockchains and DLTs in general.

3.4.2.1 Hyperledger Fabric

While there exist many blockchains, and with Ethereum having introduced smart contracts, enterprise use is growing. The issue at hand, which Hyperledger Fabric tries to solve, is that these blockchains try to adapt to this environment, which consists of identifiable participants, permissioned networks, high transaction throughput performance, therefore also low latency of transaction confirmation and finally privacy and confidentiality of transactions and data relating to business attractions. Hyperledger Fabric was designed solving these requirements in mind. [25]

It is also able to leverage consensus protocols, that do not necessitate cryptocurrency to incentivize mining or to fuel smart contract execution. Due to its modularity it may be configured in any way users may require.

To address the issue of non-determinism mentioned in Section 2.1.2, Hyperledger Fabric deploys a new architecture *execute-order-validate*, which enables the use of other languages. Execute a transaction and check its correctness, thereby endorsing it, order transactions via a (pluggable) consensus protocol, and validate transactions against an application-specific endorsement policy before committing them to the ledger. [25]

3.4.2.2 Hyperledger Aries

Hyperledger Aries is a graduated project from the Hyperledger foundation that focuses on on creating, transmitting and storing verifiable digital credentials, with its infrastructure aimed at blockchain-rooted peer-2-peer interactions.

As mentioned in Section 2.2, Aries is already used to manage VCs through compatible wallets in larger organizations such as the EU. Often used in conjunction with Indy and Ursa at client layer...

3.4.2.3 Hyperledger Indy

Hyperledger Indy is a blockchain platform designed for decentralized identity management. It provides tools, libraries, and reusable components for creating and using independent digital identities rooted on blockchains or other distributed ledgers. Indy focuses on providing a decentralized identity infrastructure that is secure, privacy-preserving, and interoperable. It offers an implementation of the resolver and works closely with Hyperledger Aries to enable interoperability with other platforms. Closely associated with the Sovrin Project

3.5 IoTeX

IoTeX is a blockchain based framework focusing on decentralized application and IoT devices. It provides functionalities such as blockchain, interconnections of decentralized applications, Dapps, using DIDs. Architecturally is is quite similar to Ethereum, with a focus on IoT devices and also collaborating with Ethereum, making it Ethereum Virtual Machine, EVM, compatible. Further DIDs are used to as on-chain identity framework that enables users and devices to own their data, identity, and credentials. Further Real World Data Oracles convert real world phenomena into verifiable and blockchain-ready data for use in the IoTeX Dapps. IoTeX also supports Trusted Execution Environment, TEE, which is important for this thesis, i.e., for secure storage and execution of certain applications and transactions.

Transactions are managed through the *IOTX* token, e.g., to stake smart contracts as mentioned in Section 3.4.1.

An advantage in using IoTeX would be, that it has already been used in IoT related projects, proving performance in the area.

A shortcoming is, that the blockchain is not usable in a private environment, as it is permissionless and public. Therefore in respect to this thesis, one needs to contrast, whether storing information is viable, which it does not for us.

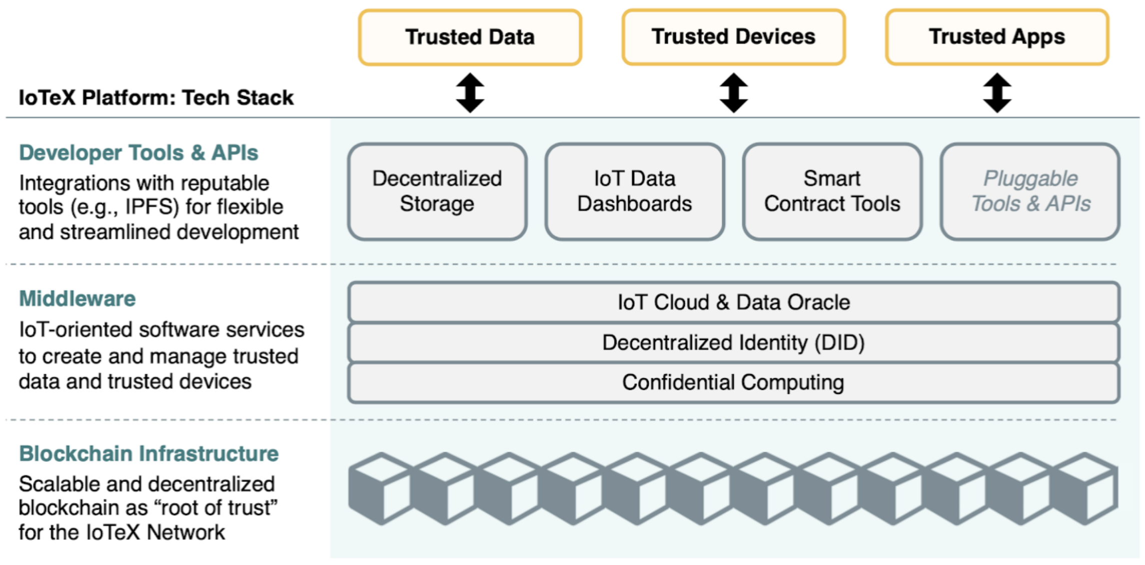
Trusted Data
Trusted Devices
Trusted Apps
IoTeX Platform: Tech Stack
Developer Tools & APIs
Integrations with reputable tools (e.g., IPFS) for flexible and streamlined development
Decentralized Storage
IoT Data Dashboards
Smart Contract Tools
Pluggable Tools & APIs
Middleware
IoT-oriented software services to create and manage trusted data and trusted devices
IoT Cloud & Data Oracle
Decentralized Identity (DID)
Confidential Computing
Blockchain Infrastructure
Scalable and decentralized blockchain as "root of trust" for the IoTeX Network

Figure 3.2: IoTeX Platform Overview

Table 3.2: Considered Blockchains

Blockchain Name	**Key Characteristics**
Hyperledger Iroha [15]	Permissioned Network
Hyperledger Sawtooth [15]	Permissioned Network, private, (aimed at enterprise use, scalable)
Ethereum	Permissionless
IoTeX [26]	Permissionless
IBM Blockchain	***TODO***

3.6 Sovrin

TODO

3.7 Further non-DLT based IdM Frameworks

There are more frameworks to manage IoT devices, although they do not employ DLTs and therefore are not a focus of this thesis.

3.8 Blockchains

3.8.1 Ethereum

Ethereum is a widely known and used blockchain.

Also there is the Hyperledger Besu [15] client to interact with Ethereum.

There is the Ethereum Virtual Machine...

3.8.2 Hyperledger Iroha

Hyperledger Iroha is a relatively new and unknown blockchain from the Hyperledger Foundation [15].

3.8.3 IoTeX (Blockchain)

See Section 3.5.

3.8.4 IBM Blockchain

We will not consider IBM Blockchain, as it is proprietary and builds up on Hyperledger products anyway.

3.9 Tools

3.9.1 Docker

Docker is a great alternative to Virtual Machines, VM, and makes it easier to distribute environment or images to end users or developers.

Chapter 4

Architecture and Design

4.1 Actors

In our architecture there will be a few actors, that play various roles, outlined in the below sections.

There will be a summarized device entity, that will be split up into two devices, as seen below in Section 4.2, and further explained in Chapter 5. There will be a Manufacturer, that is responsible for the fabrication of said devices, as well as the creation of Certificates and credentials, that will later be used by the devices and other entities for verification. Further there shall be an auditor, that inspects said credentials and verifies them, or the necessary cases also invalidation. Finally an entity Infrastructure shall exist, that will enable the communication facilities for the entities.

We will assume a simplified and already predefined *Authentication, Authorization and Accounting*, AAA, server in order to simplify our environment.

Further external stakeholders include a certification authority and a vulnerability database.

4.2 Physical Components

The composition of our physical architecture will entail two main devices of importance for us, as later defined in more detail in Chapter 5, Section 5.3.

4.2.1 Edge Nodes

The edge nodes, also Internet of Things, IoT, nodes, are the frontier where direct interaction happens, be that sensing, actuating or interaction through a Human Machine Interface, HMI.

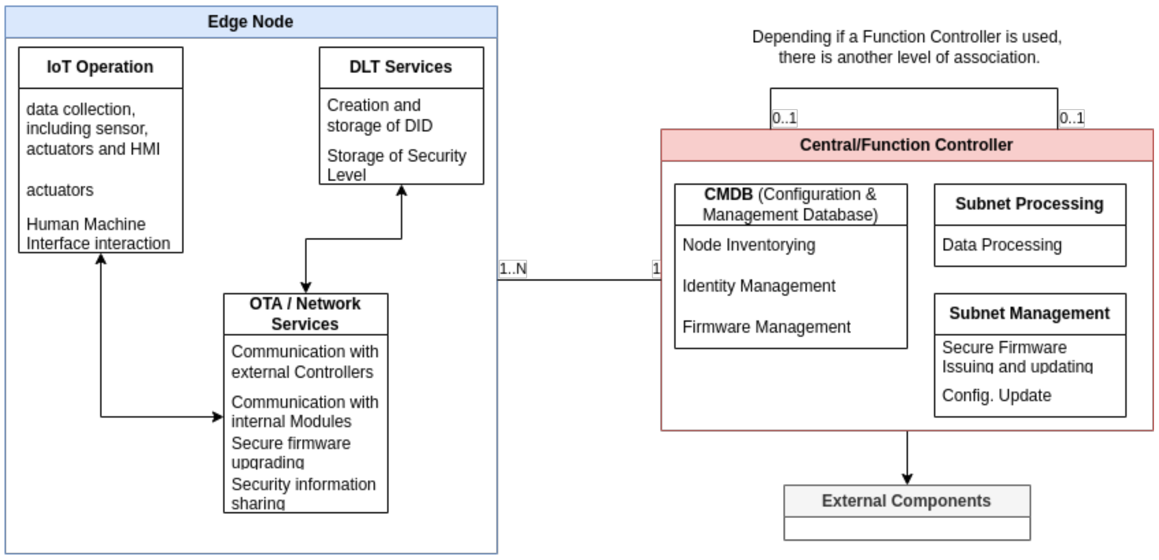
Edge Node
IoT Operation
data collection, including sensor, actuators and HMI
actuators
Human Machine Interface interaction
DLT Services
Creation and storage of DID
Storage of Security Level
OTA / Network Services
Communication with external Controllers
Communication with internal Modules
Secure firmware upgrading
Security information sharing
1..N
1
Depending if a Function Controller is used, there is another level of association.
0..1
0..1
Central/Function Controller
CMDB (Configuration & Management Database)
Node Inventorying
Identity Management
Firmware Management
Subnet Processing
Data Processing
Subnet Management
Secure Firmware Issuing and updating
Config. Update
External Components

Figure 4.1: Device Architecture Overview

Each edge node is uniquely identifiable through a Decentralized Identifier, DID, which will later be used for device verification for outside entities through usage of a DLT. Furthermore, security levels, access levels will be stored on each device. Through the use of MUDs, each device will be defined to have certain capabilities and access privileges, which all will be enforced by the controller node.

Warning: Whether we will store our private key in a Secure Element, SE, or on the SoC FLASH, is not yet decided.

4.2.2 Controller Nodes

The controller nodes are managers of the edge nodes. They fulfill various responsibilities and shield edge nodes in a subnet for increased security control

4.3 Onboarding

The onboarding of a device can be observed in a sequence diagram in Figure 4.2. All of the actors in Section 4.1 are involved.

We assume a device has been manufactured and that it is now at the desired location. In the first time setup, the device will be either programmed to create a DID on setup, or will be assigned/created one in the manufacturing process, depending on the computational and hardware facilities. Building up on that, as mentioned in Section 2.4 the device will have unique key, but we will not be generating such a function and neither derive a key for further use and we assume a predefined key to be placed into the device.

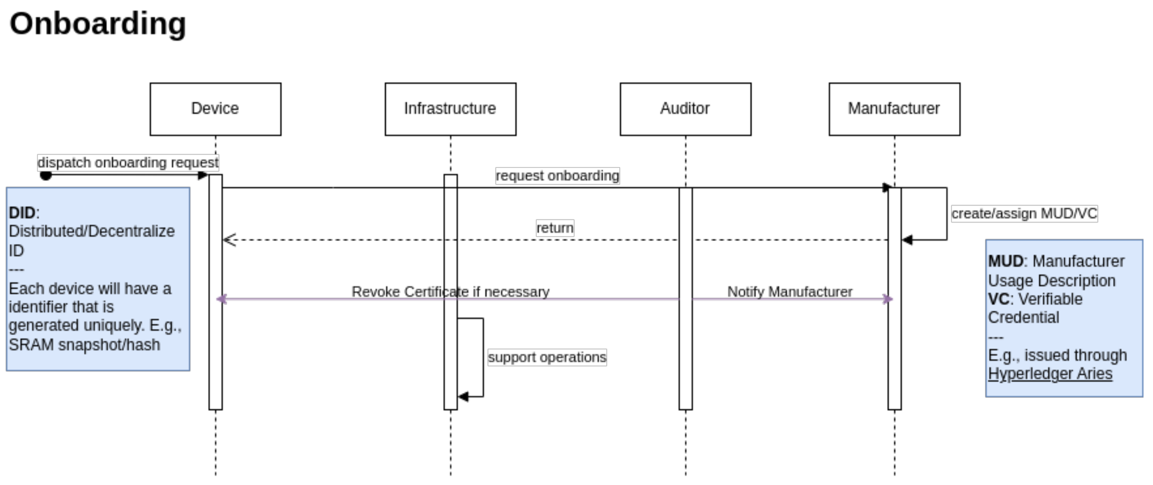
Onboarding
Device
Infrastructure
Auditor
Manufacturer
dispatch onboarding request
request onboarding
create/assign MUD/VC
return
DID: Distributed/Decentralize ID --- Each device will have a identifier that is generated uniquely. E.g., SRAM snapshot/hash
Revoke Certificate if necessary
Notify Manufacturer
support operations
MUD: Manufacturer Usage Description VC: Verifiable Credential --- E.g., issued through Hyperledger Aries

Figure 4.2: Onboarding Sequence Diagram

Both an edge and controller node shall have their DID, since both will be managed and or inventoried by another instance or entity. Necessary for this process is a valid MUD file, also issued by the manufacturer, on device fabrication or on first time setup. This MUD does not necessarily need to be on the device, but an URL is needed, which points to the location of such file, see Section 2.3. Each device shall request and onboarding by the manufacturer, which then verifies the DID, and in case of validity issues a Verifiable Credential, VC, to the device.

In more detail, an edge node shall request onboarding from the controller node, which then forwards the request to the manufacturer.

The Auditor then proceeds to verify the validity of the VC and checks for any known vulnerabilities, that might be affect the device that requested the VC. The Auditor shall also monitor the VC periodically, in order to check for any vulnerabilities the devices themselves might have missed. In case of an insecurity or vulnerability the auditor shall notify both the device and the manufacturer and revoke the VC, until the device has been known to update its Firmware to a secure version, see Section 4.6.

4.3.1 VC Verification

The verification of a VC may happen through the usage of a Smart Contract. Therein the invalidation of said smart contract happens through the fulfillment of the contract condition, which could be a discovered vulnerability.

4.4 Inventorying

Getting to the first emphasis of our thesis, we will explore, how we can safely and efficiently store and keep an inventory of our devices.

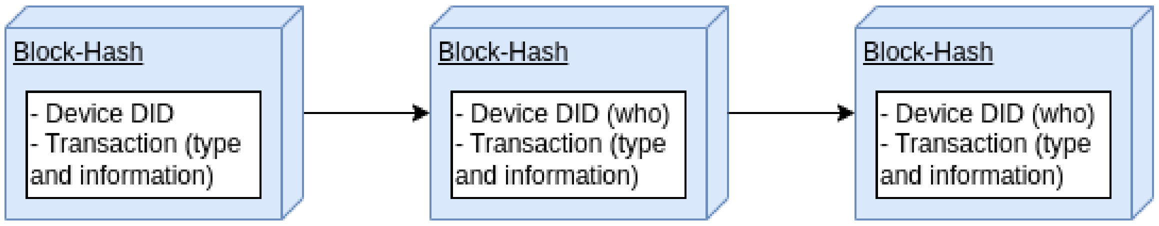
Block-Hash
- Device DID
- Transaction (type and information)
Block-Hash
- Device DID (who)
- Transaction (type and information)
Block-Hash
- Device DID (who)
- Transaction (type and information)

Figure 4.3: DLT Architecture

For this we will rely on a DLT, in which any action will be saved as a block. The DLT used shall be permissioned and possibly private, so that only those permissioned may be a part of this blockchain. We don't see the need for the chain to be publicly accessible, **although whether public read-access is out of the question is yet to be determined**. The DLT will work as a decentralized Configuration and Management Database, CMDB, as seen in Figure 4.3. Transaction will be fulfilled by the controller node as indicated in Figure 4.1.

4.5 Operations

In order to be able to simulate the whole device lifecycle, we will also implement simple day-2-day operations, such as reading out sensors, actuating or minor interactions through HMIs.

Depending on the device type, edge or controller, further responsibilities are expected. A controller node has to constantly check against the DLT, whether there were any new vulnerabilities discovered and if the VC was consequently revoked, as mentioned before, the controller node has to check its own status, as well as the ones of the subnet of edge nodes. If such were the cases, concerned devices need be recertified.

4.6 Secure Firmware Updating

As the second emphasis of this thesis we aim to provide a secure update mechanism for outdated and or vulnerable firmware versions.

4.7 Secure Information Sharing

In order to enable the sharing of information related to the security of the device, we will rely on directional information sharing, meaning, the devies receive vulnerability information, updates and MUD files, while external sources receive status updates and co. of the devices concerned.

These information can be embedded into transaction inside the DLT, depending on the importance of the information, it can also be shared through less sophisticated measures.

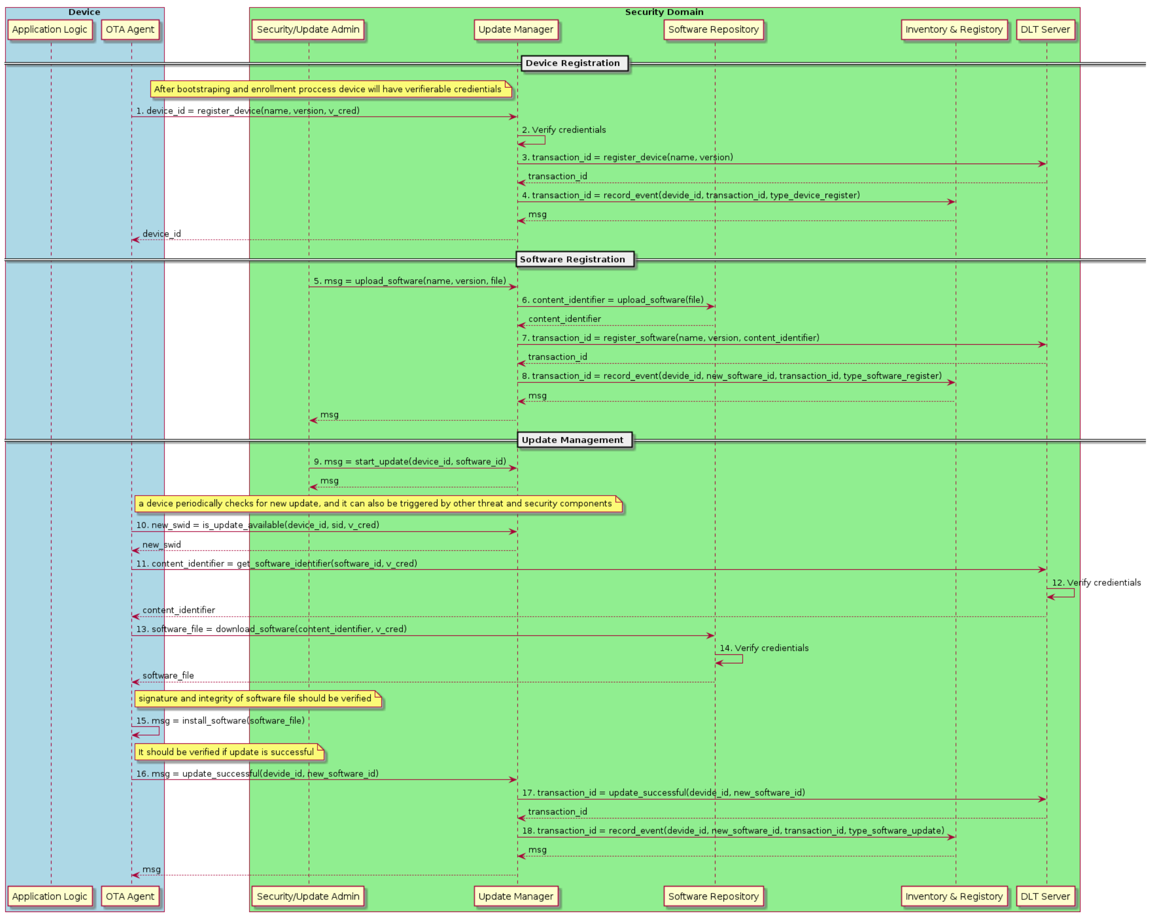
Device
Security Domain
Application Logic
OTA Agent
Security/Update Admin
Update Manager
Software Repository
Inventory & Registory
DLT Server
Device Registration
After bootstraping and enrollment proccess device will have verifierable credientials
1. device_id = register_device(name, version, v_cred)
2. Verify credientials
3. transaction_id = register_device(name, version)
transaction_id
4. transaction_id = record_event(devide_id, transaction_id, type_device_register)
msg
device_id
Software Registration
5. msg = upload_software(name, version, file)
6. content_identifier = upload_software(file)
content_identifier
7. transaction_id = register_software(name, version, content_identifier)
transaction_id
8. transaction_id = record_event(devide_id, new_software_id, transaction_id, type_software_register)
msg
msg
Update Management
9. msg = start_update(device_id, software_id)
msg
a device periodically checks for new update, and it can also be triggered by other threat and security components
10. new_swid = is_update_available(device_id, sid, v_cred)
new_swid
11. content_identifier = get_software_identifier(software_id, v_cred)
12. Verify credientials
content_identifier
13. software_file = download_software(content_identifier, v_cred)
14. Verify credientials
software_file
signature and integrity of software file should be verified
15. msg = install_software(software_file)
It should be verified if update is successful
16. msg = update_successful(devide_id, new_software_id)
17. transaction_id = update_successful(devide_id, new_software_id)
transaction_id
18. transaction_id = record_event(devide_id, new_software_id, transaction_id, type_software_update)
msg
msg
Application Logic
OTA Agent
Security/Update Admin
Update Manager
Software Repository
Inventory & Registory
DLT Server

Figure 4.4: CERTIFY OTA Update [23]

Chapter 5

Use Case Definition - Connected Cabin System

5.1 Background

Our use case will take the CCS scenario from Figure 5.1 into consideration and build up on their use cases, which were borrowed from the CERTIFY project [23].

Nowadays more and more IoT devices are being deployed to aircraft cabins to improve passenger experience and airline operations. Benefits span from remote PHM to reduced maintenance time, while also supporting a continuous (re)certification process.

5.2 Actors

We will consider following actors for our use case.

- Airline: Owns the aircraft and oversees daily interactions and systems operations.
- Airplane maintainer: They could be e.g., the airplane manufacturer. Oversees maintenance of the aircraft, including the integration of systems designed by different manufacturers and their configuration.
- Product Owner: Oversees design and maintenance of systems deployed in the aircraft on assignment of the airplane maintainer.
- Maintenance operator: They work for the airplane maintainer. Their responsibilities include e.g., the replacement of devices or on-site software upgrades of e.g., portable data loaders.
- Passenger, Attendant, Pilot: They interact with the aircraft through sensors, actuators or HMI.

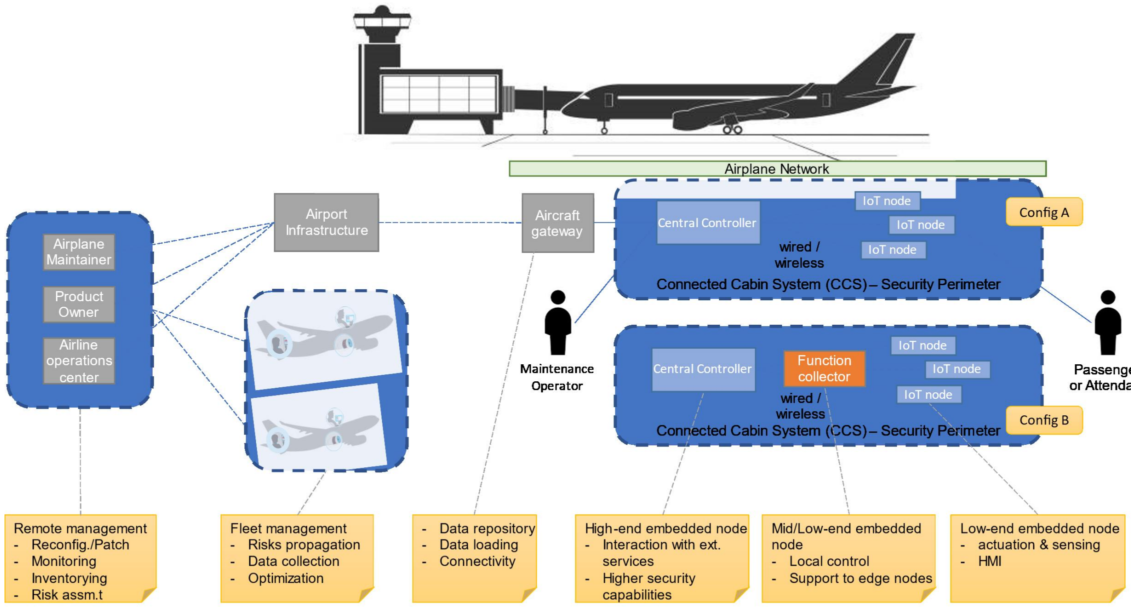

Figure 5.1: Collins CCS

5.3 System Components

We will consider an aircraft to have multiple networks, covering various aspects.

- In-flight entertainment system
- Aircraft System
- Flight Maintenance

For our use case we will assume config 'A' as the main configuration of the networks, where edge nodes are connected to a central controller that manages the edge nodes as a subnet.

- IoT / Edge Nodes: low-end devices, including actuation, sensing or HMI capabilities, with limited room for hardware and software based cybersecurity, that requires offloading to a more capable instance.
- Central Controller: High-end devices with ability to host full-fledged security functionalities.

External communication will take place through aircraft gateway offering services for data repository, data loading and connectivity with external environment. The airline operations center, product owner and airplane maintainer can interact through the airport infrastructure. A technician may directly access the aircraft if necessary.

Table 5.1: Actors involved

Airline	Airplane Maintainer	Product Owner	Maintenance Operator	Passenger, Attendant, Pilot
X	X	X	X	X

Table 5.2: Lifecycle stages involved

Bootstrapping	Operation	Update	Repurposing	Decommissioning
X	-	X	-	X

5.4 Scenarios

5.4.1 Installation of Connected Cabin Systems

5.4.1.1 Goals

The goals of this scenario include bootstrapping and customization of devices for specific deployment, updating and decommissioning of previous systems, guaranteeing a reset to a known and fresh, wiped data, state. Table 5.1 highlights the involved actors and Table 5.2 shows the stages involved in this scenario.

5.4.1.2 Pre-condition

In order for this scenario to be valid, following pre-conditions need to be met:

- Actors involved can establish a secure connection with the aircraft, wireless or wired, through airport infrastructure
- Airport and Aircraft network infrastructure can receive authorization requests for needed connections from the external environment.
- The Maintenance Operator is provided access to the airplane and to maintenance ports of the target CCS.

5.4.1.3 Sub-Scenario 1: Component Installation

Flow of Events The flow of events can be tracked in Figure 5.2 and is verbalized as follows:

- Airline requests installation of new component to the Airplane Maintainer also issuing an authorization request to Airport and Airplane gateways.

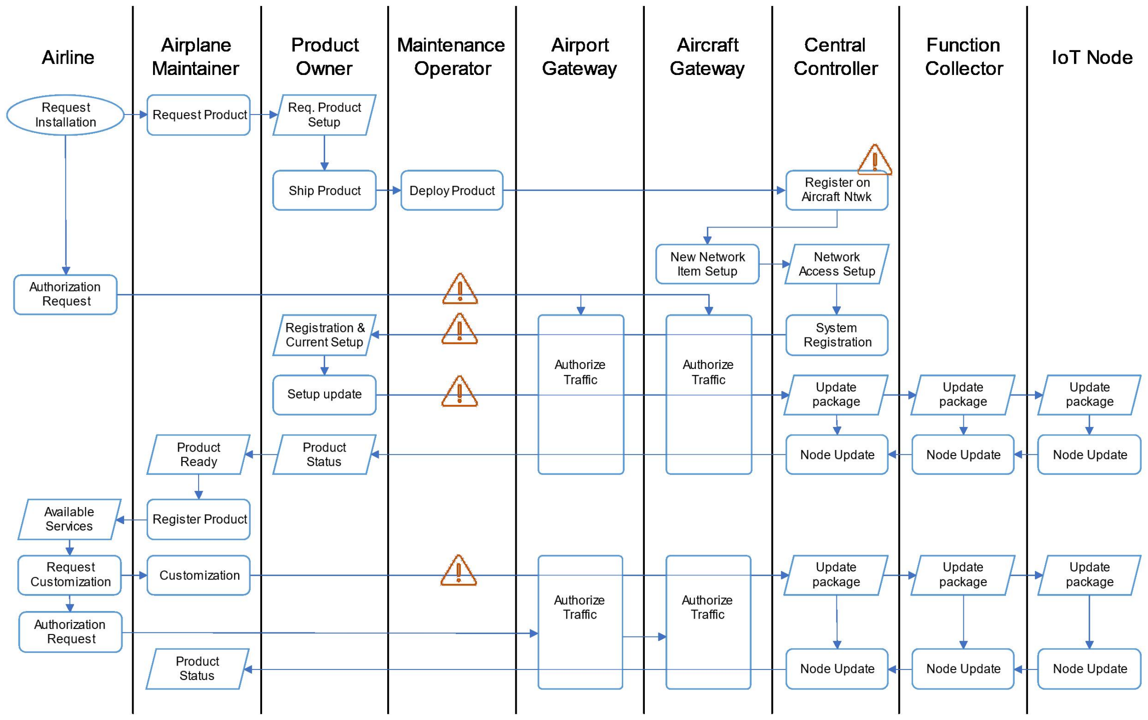

Figure 5.2: Collins Scenario 1: Device Installation

- The request is forwarded to the Product Owner and then to the Maintenance Operator, who oversees the physical deployment of the product.
- Once connected, Central Controller registers on the Aircraft Network and receives required setup to complete network access and system registration.
- Product Owner is now able to reach the CCS, push configuration and security updates to the Central Controller, as well as the Function Collector and IoT nodes.
- After the update, the product is registered and the Airplane Maintainer can offer remote services to the Airline.
- The Airline requests a customization of the CCS. It is performed by the Airplane Maintainer by pushing an update package and/or modifying specific configurations as allowed by the Product Owner API for Maintenance.
- The new product status is confirmed with a feedback message.

5.4.1.4 Sub-Scenario 2: Component Replacement

Flow of Events

- The Airline requests replacement of a component to the Airplane maintainer, also issuing an authorization request to the Airport and Airplane Gateways.

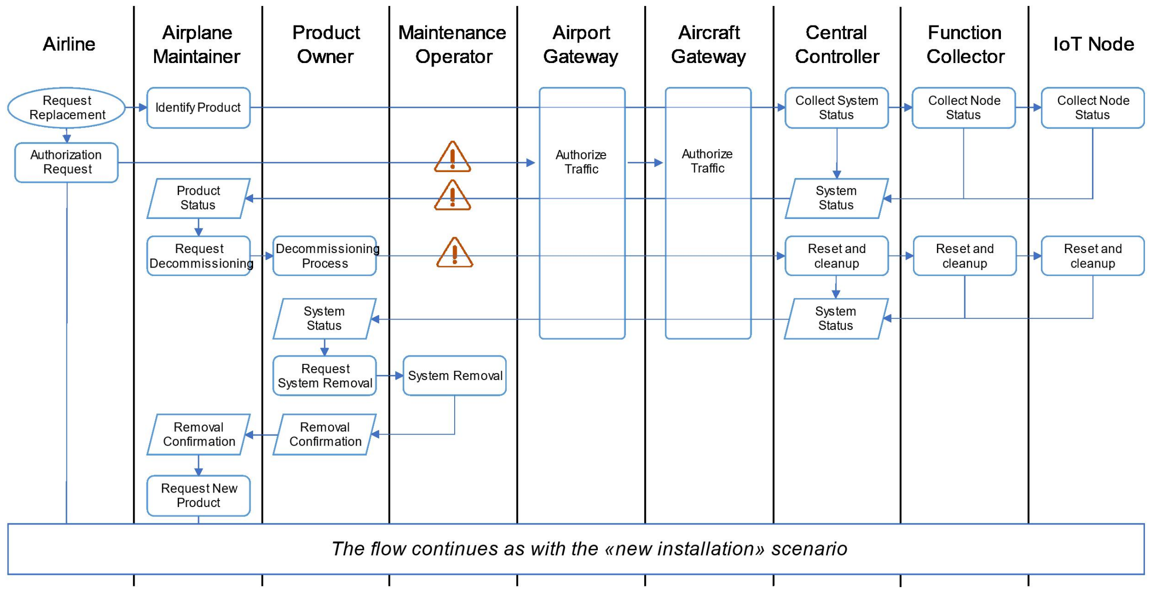

Figure 5.3: Collins Scenario 1: Device Replacement

- The Airplane Maintainer identifies the target product (location) and collects latest system status.
- The Airplane Maintainer issues a decommissioning request to the Product Owner and starts the decommissioning process, which causes a reset and cleanup of all the nodes that will be replaced.
- After remote reset and clean-up, product owner requests the Maintenance Operator to physically remove the system from the cabin.
- The product is then unregistered and can be dismissed.
- The remaining part continues with the 'New installation process flow'

5.4.1.5 Post-Condition

After completion of the Installation Scenario, following post-conditions need to be met:

- New component is deployed in the CCS, integrated into the network, updated with latest security patches and configured by the Airline for their specific needs.
- Component is securely onboarded in the CCS, unique identity and certificates are dispatched for authentication.
- Regarding the configurations, their integrity is verified and confidentiality has been preserved.

Table 5.3: Actors involved operations

Airline	Airplane Maintainer	Product Owner	Maintenance Operator	Passenger, Attendant, Pilot
X	X	X	-	X

Table 5.4: Lifecycle stages involved

Bootstrapping	Operation	Update	Repurposing	Decommissioning
-	X	X	-	-

5.4.1.6 Attack Scenario

As an alternative flow of events, i.e., in an attack scenario, highlighted by the yellow triangles in Figure 5.2 and Figure 5.3 following points were identified:

- Attacker can inject malicious payloads in place of the intended one, (confidential) credentials provided to the Central Controller for network access and authentication ca be stolen.
- IP sensitive data can be leaked by the Airplane Maintainer when retrieving the system status.
- The integrity of maintenance/reset/cleanup procedures can be compromised.

5.4.2 System Operation and Monitoring

5.4.2.1 Goals

The goals of System Operations and Monitoring incorporate following points:

- Periodic collection of data from airplane.
- Data offload/upload to ground stations for performance monitoring, optimization and PHM operations.
- Attendants interact with CCS through a HMI
- CCS Information is collected and stored in Gateway.
- A limited set of predefined reconfigurations may be performed on plane, when requested by Airline, Maintainer or product owner.

5.4.2.2 Pre-conditions

- Maintainers have established remote secure connection with the aircraft (wireless or wired) through the airport infrastructure.
- Passengers/Attendants/Pilots can interact through HMI or are connected through other devices e.g., through WiFi (possibly in the sense of bring you own device, BYOD).
- Device bootstrapping, enrollment, configuration, provisioning are completed for all devices, that are (statically) part of the network.
- IoT devices are equipped with sensors to collect and store data that are then forwarded to their root controller.
- Devices can securely store and transmit collected data.

5.4.2.3 Flow of Events

5.4 5.5 5.6

- IoT devices collect data during airplane operations, see Figure 5.4
- Data is securely stored on-board, see Figure 5.4
- Local computations over critical and non-critical data are executed in separate environments to reconfigure the airplane/flight, see Figure 5.4 **What is meant by this?**
- Remote entities are authenticated and a connection is established with the aircraft network through the gateway, see Figure 5.6 **What Gateway? We are basically in flight so shouldn't we be disconnected? Otherwise this is a scenario for Roaming**
- Data is downloaded from plane to ground
- In case of an Airplane fleet, Collective Analysis is performed, see Figure 5.5
- Upload new data and configurations, from ground to plane
- Data authenticity and integrity are verified before updating the configurations on the plane.

5.4.2.4 Post-Conditions

After any of the sub-scenarios from Figures 5.4, 5.5, 5.6, the following post-conditioned need to be met:

- A new configuration, computed either locally or through the remote connection with the operations center is available.

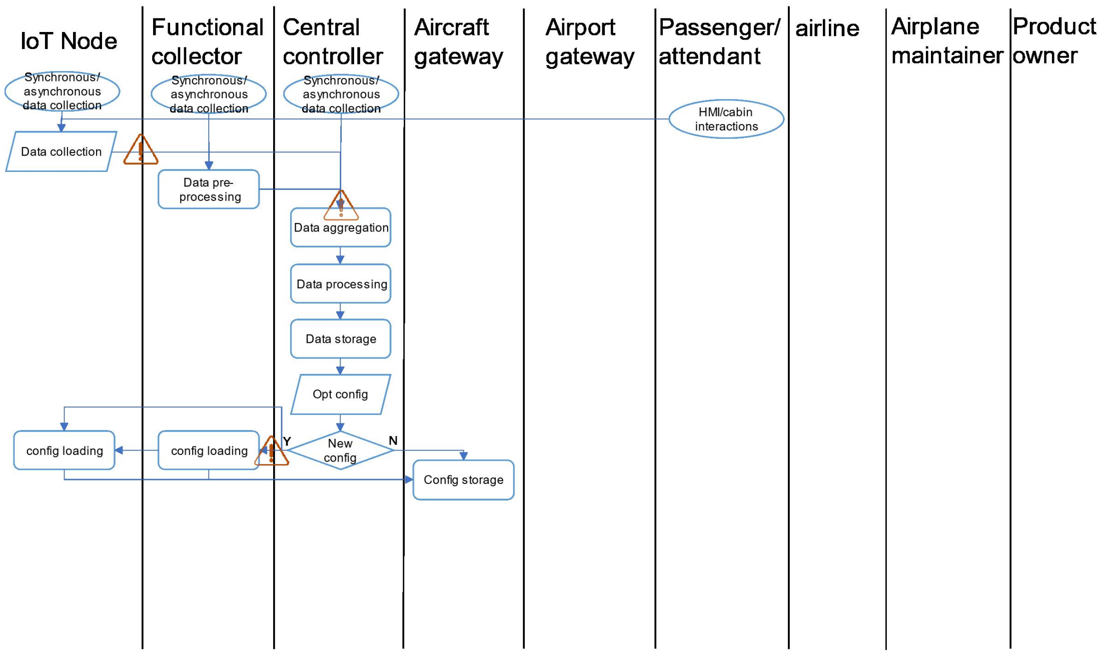

Figure 5.4: Collins Scenario 2.1: Data Collection and Local Reconfiguration

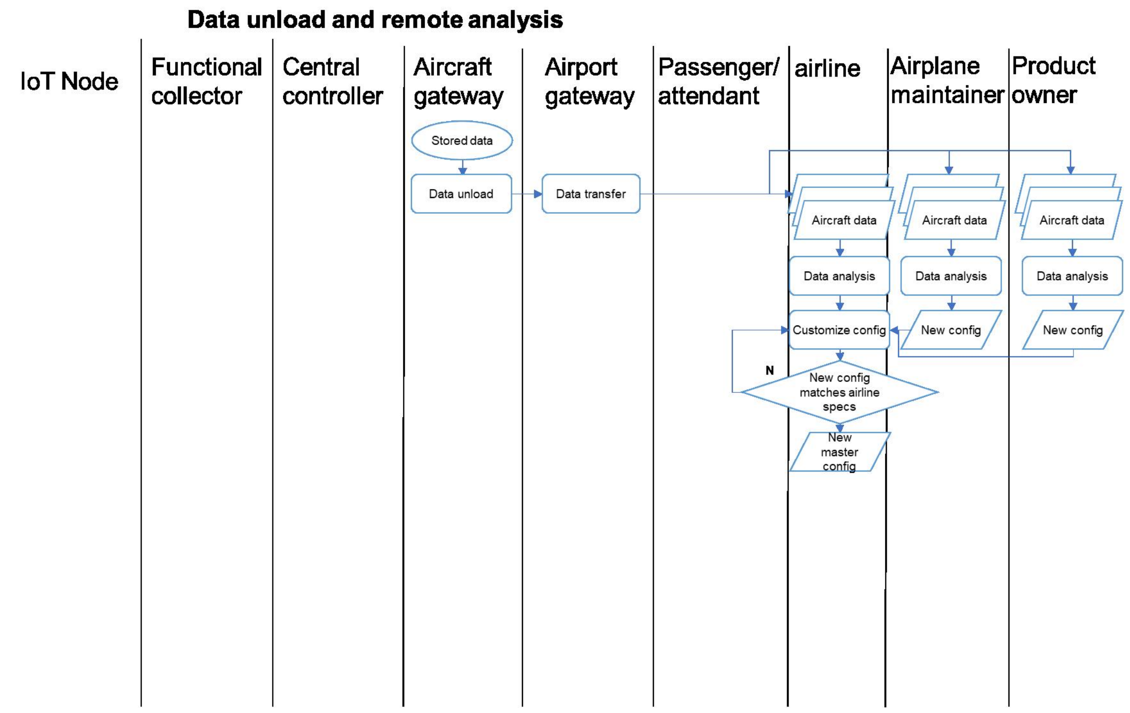

Figure 5.5: Collins Scenario 2.2: Data Unload and Remote Analysis

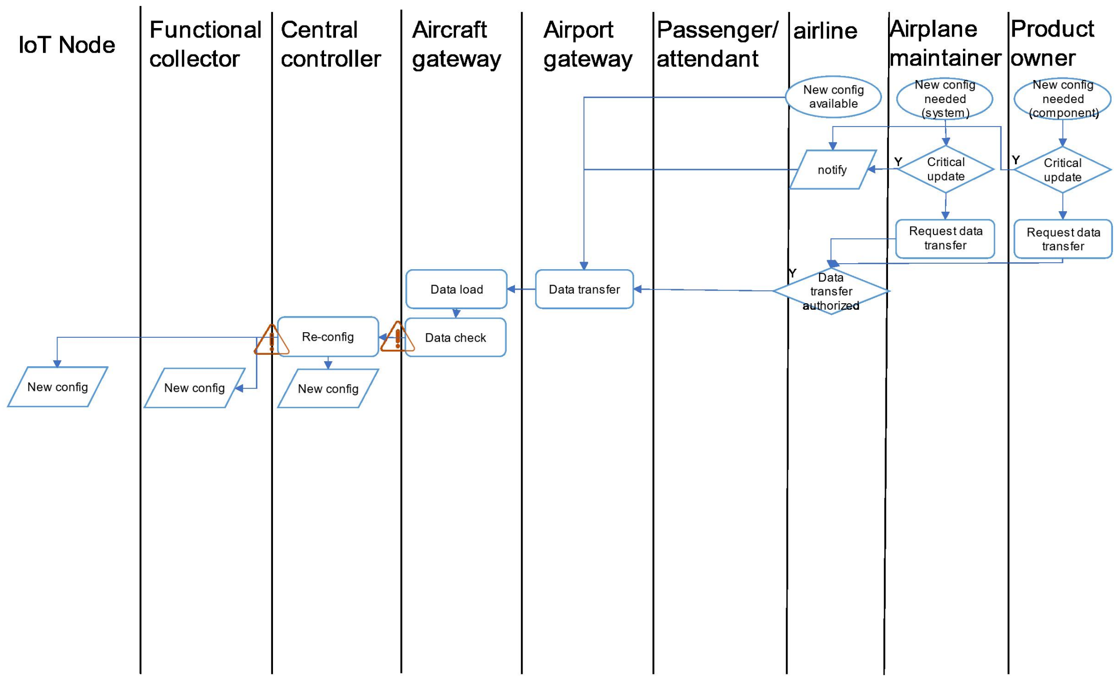

Figure 5.6: Collins Scenario 2.3: Data Load and Remote Reconfiguration

- Data from the aircraft are available for further analysis of Airline, Maintainer or Product Owner.
- For the configurations, integrity is verified, and confidentiality has been preserved (as it could involve IP issues) in the process.
- For the data, in addition to integrity and confidentiality, it is important to also ensure availability (to detect early sings of potential problems).

5.4.2.5 Attack Scenario

As an alternative flow of events, where an attack might happen, the main entry point is through the WiFi access point. A rogue device could inject false data, configurations or software to facilitate subsequent attacks or even cause system unavailability or device malfunctioning. Alternatively in stealth mode, sensitive information could be captures and could be used to perpetrate other attacks.

5.4.3 LRU Replacement and Repurposing

Akin to Scenario 1 in Subsection 5.4.1 in this Scenario we also consider similar steps to replace existing devices, with the difference, that the devices that replace the broken devices were not exactly meant for this situation.

Table 5.5: Actors involved

Airline	Airplane Maintainer	Product Owner	Maintenance Operator	Passenger, Attendant, Pilot
X	X	X	X	-

Table 5.6: Lifecycle stages involved

Bootstrapping	Operation	Update	Repurposing	Decommissioning
-	-	X	X	X

5.4.3.1 Goals

The goal of this scenario is to quickly replace a malfunctioning Central or Functional Controller, but a LRU is not directly available. To minimize downtime, a spare LRU is retrieved from the same manufacturer and repurposed to the specific target system. Airline, Airplane Maintainer, Product Owner, and Maintenance Operator are all involved to take care of different steps in the process.

5.4.3.2 Pre-Conditions

Following pre-conditions must be met:

- Actors involved can establish remote secure connection with aircraft.
- Airport has a spare LRU, that is compatible with CCS.
- The Maintenance Operator is provided access to the airplane and to the maintenance ports.

5.4.3.3 Flow of Events

- The Airline requests to the Airplane Maintainer the repair of a cabin system, issuing and authorization request to the Airport/Airplane gateways for following remote software update operations.
- The Airplane Maintainer identifies the target product, including its location, and the failure condition then requests repair to Product Owner.
- The Product Owner starts the removal process of the LRU, including reset and cleanup
- Failed LRU removal executed locally by Maintenance Operator (also in charge of reset and cleanup)

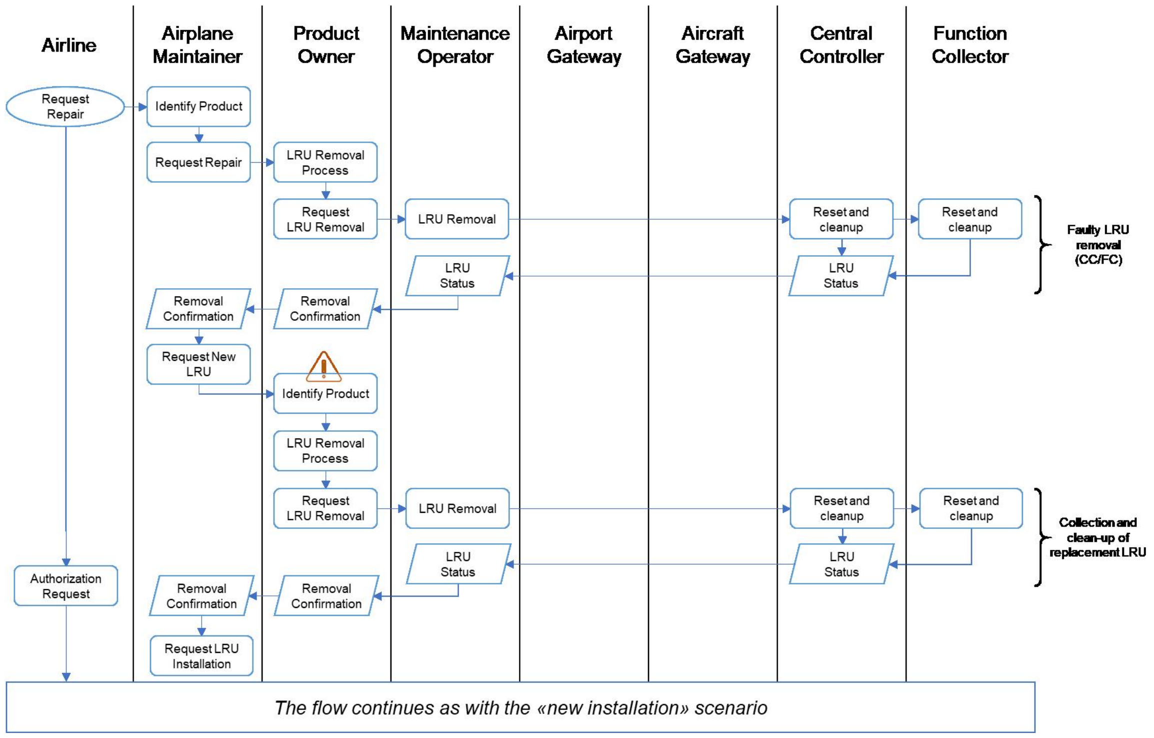

Figure 5.7: Collins Scenario 3: LRU Replacement

- Product owner informs Airplane Maintainer of the removal and receives information of available replacement LRUs.
- The replacement LRU shall be available from a remote location.
- The remaining flow proceeds as in Scenario 1 in Subsection 5.4.1

5.4.3.4 Post-Conditions

- A new LRU is deployed, integrated into network, updated with latest security patches and configured by the Airline for specific needs
- CCS is registered with unique identifier and certificates are dispatched for authentication.
- For the configuration, integrity is verified and confidentiality has been preserved (for IP protection) in the process

5.4.3.5 Attack Scenario

Attacks follow same pattern as in Subsection 5.4.1 from Scenario 1. An attacker can inject malicious software or a counterfeit LRU through the supply chain.

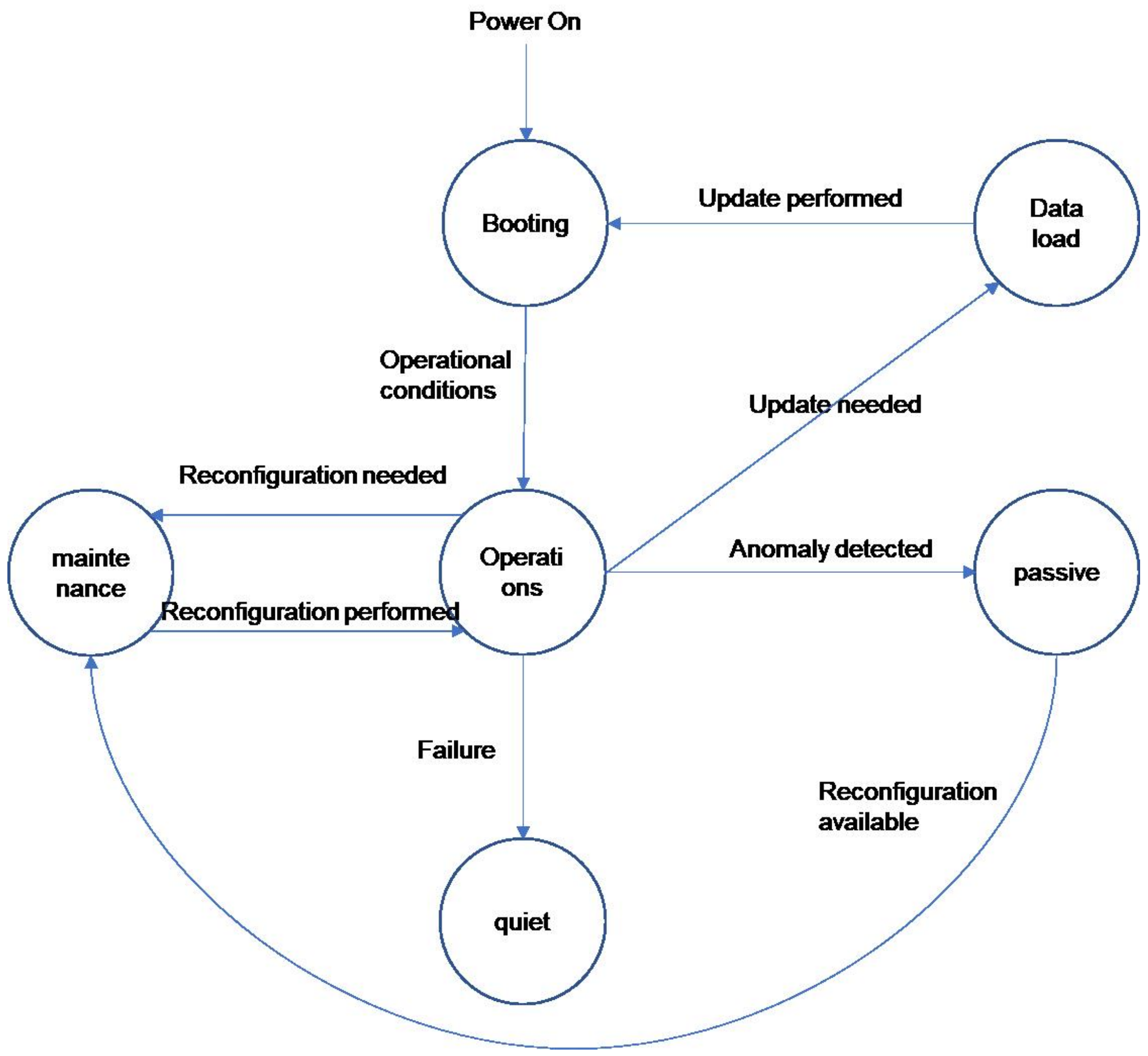

Figure 5.8: Collins Operation Modes for an On-Board Device

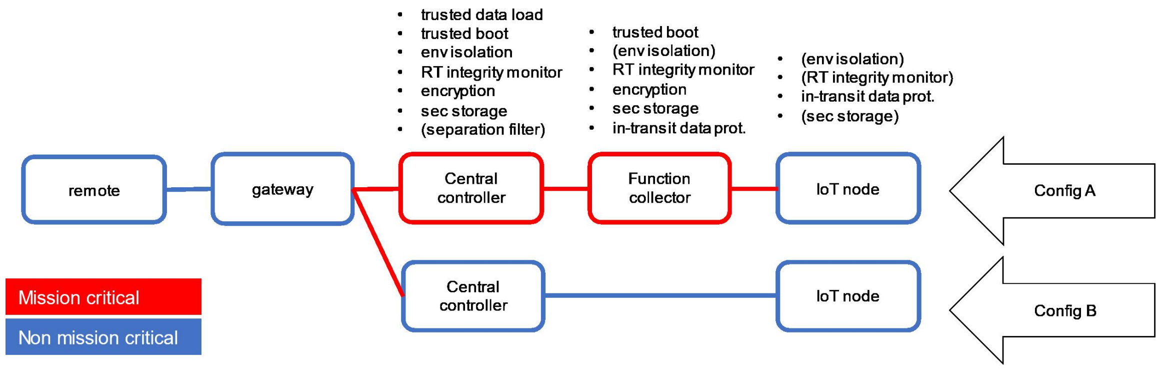

Figure 5.9: Collins Desired Security Features for the Different CCS Components

5.5 Cybersecurity Assessment

The CERTIFY [23] project describes following security levels:

- Basic: Focus on common and simple cyber threats. User authentication, access controls and basic security configuration.
- Substantial: More robust security measures including intrusion detection systems, IDS, incident response plans and periodic security assessment.
- High: Requires organizations to establish comprehensive and proactive cybersecurity programs. Further advanced security measures, network segmentation, encryption, continuous monitoring and regular vulnerability assessments are needed. Threat intelligence sharing and regular security audits are also in order.

Our use case, CCS, falls into the category of level *HIGH*, as from a cybersecurity viewpoint the aircraft cabin is considered a highly volatile and highly targeted environment.

Chapter 6

Implementation

For our implementation it was important that we do not reinvent the wheel at each corner.

Choosing a DLT platform was not easy, as there were many options for Blockchains/DLTs but very few that were more or less directly applicable to our thesis, which is IoTeX but its permissionless nature is not of good to us, as we would like our network to be permissioned, which is another increase in security.

Another aspect of was the quality of documentation. In order not to waste time trying to figure out the source code, documentation of it is important.

The choice of physical devices, whose purpose lies with demonstrating our platforms capabilities, fell to the following. For the controller nodes, we will be using Raspberry Pi 4B, demonstrating portability and the use of embedded Linux, reflecting future capabilities. It is able to coordinate with the chosen blockchains.

For the edge nodes we have decided to use the Raspberry Pi Pico, which is a capable and demonstrative microcontroller unit, MCU, able to run CircuitPython, as well as simple bare-metal software.

6.1 Configuration and Management Database

6.2 Device onboarding

For sake of simplicity, an AAA server and process will be assumed and therefore no secure onboarding will be included.

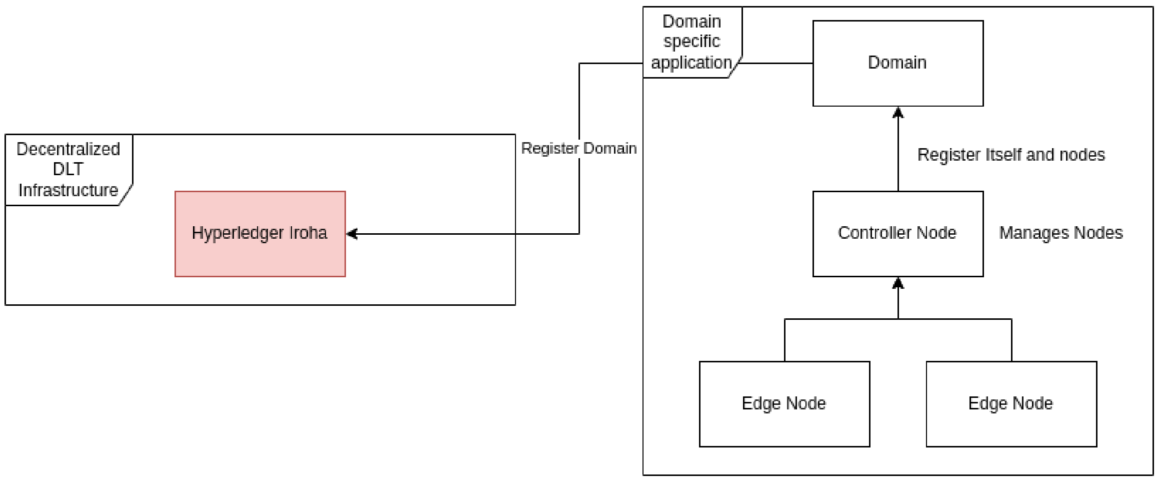
Decentralized DLT Infrastructure
Hyperledger Iroha
Register Domain
Domain specific application
Domain
Register Itself and nodes
Controller Node
Manages Nodes
Edge Node
Edge Node

Figure 6.1: CMDB using Hyperledger Iroha

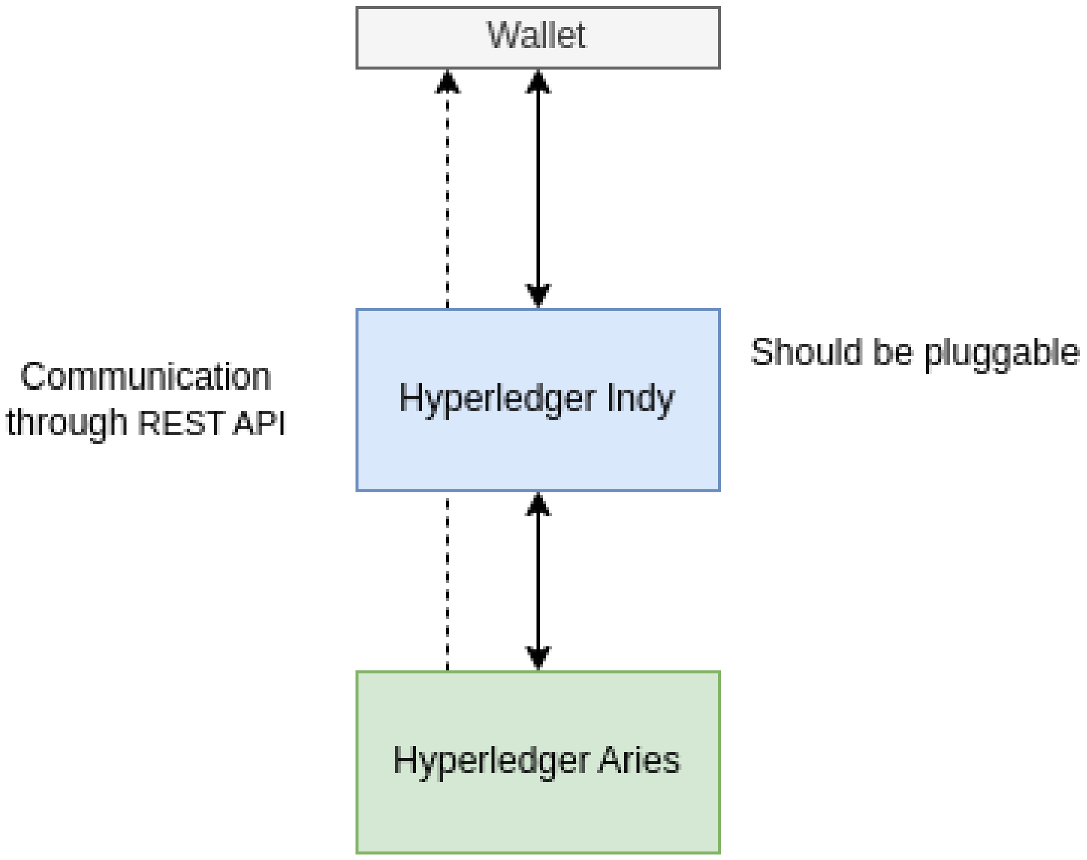

Figure 6.2: Credential Management Architecture

Chapter 7

Evaluation

Chapter 8

Summary and Conclusions

Bibliography

[1] "The eu cybersecurity act." [Online]. Available: https://digital-strategy.ec.europa.eu/en/policies/cybersecurity-act 1

[2] "Directive on measures for a high common level of cybersecurity across the union (nis2 directive)." [Online]. Available: https://digital-strategy.ec.europa.eu/en/policies/nis2-directive 1

[3] R. Neisse, G. Steri, and I. Nai-Fovino, "A blockchain-based approach for data accountability and provenance tracking," in *Proceedings of the 12th international conference on availability, reliability and security*, 2017, pp. 1–10. 2, 9

[4] (2023) Trusted iot device network-layer onboarding and lifecycle management. [Online]. Available: https://www.nccoe.nist.gov/projects/trusted-iot-device-network-layer-onboarding-and-lifecycle-management 2

[5] E. J. Scheid, T. Hegnauer, B. Rodrigues, and B. Stiller, "Bifröst: a modular blockchain interoperability api," in *2019 IEEE 44th Conference on Local Computer Networks (LCN)*. IEEE, 2019, pp. 332–339. 2

[6] V. A. Siris, P. Nikander, S. Voulgaris, N. Fotiou, D. Lagutin, and G. C. Polyzos, "Interledger approaches," *Ieee Access*, vol. 7, pp. 89 948–89 966, 2019. 2

[7] D. Dodson, D. Montgomery, W. Polk, M. Ranganathan, M. Souppaya, S. Johnson, A. Kadam, C. Pratt, D. Thakore, M. Walker *et al.*, "Securing small-business and home internet of things (iot) devices: Mitigating network-based attacks using manufacturer usage description (mud)," National Institute of Standards and Technology, Tech. Rep., 2021. 2, 6, 9, 10

[8] X. Fan, Q. Chai, L. Xu, and D. Guo, "Diam-iot: A decentralized identity and access management framework for internet of things," in *Proceedings of the 2nd ACM International Symposium on Blockchain and Secure Critical Infrastructure*. Taipei Taiwan: ACM, Oct 2020, p. 186–191. [Online]. Available: https://dl.acm.org/doi/10.1145/3384943.3409436 3, 4, 10, 12, 51

[9] H. Foundation. Hyperledger aries rfc. [Online]. Available: https://github.com/hyperledger/aries-rfcs 3

[10] W. W. W. Consortium *et al.* (2022) Verifiable credentials data model 1.1: expressing verifiable information on the web. [Online]. Available: https://www.w3.org/TR/vc-data-model/ 4, 5, 55, 61

[11] D. I. A. Domingo, “How eidas can legally support digital identity and trustworthy dlt-based transactions in the digital single market,” 2020. 4

[12] J. Sedlmeir, R. Smethurst, A. Rieger, and G. Fridgen, “Digital identities and verifiable credentials,” *Business & Information Systems Engineering*, vol. 63, no. 5, p. 603–613, Oct 2021. 4

[13] Decentralized identity. [Online]. Available: https://ethereum.org 4, 5, 49

[14] A. Preukschat and D. Reed, *Self-sovereign identity.* Manning Publications, 2021. 4

[15] H. Foundation, “Hyperledger wiki.” [Online]. Available: https://wiki.hyperledger.org/ 4, 11, 14, 15

[16] I. Williams, “Cross-chain blockchain networks, compatibility standards, and interoperability standards: The case of european blockchain services infrastructure,” in *Cross-Industry Use of Blockchain Technology and Opportunities for the Future.* IGI global, 2020, pp. 150–165. 4

[17] W. W. W. Consortium *et al.* (2022) Decentralized identifiers (dids) v1.0. [Online]. Available: https://w3c.github.io/did-core/ 5, 50, 55, 62

[18] ——. (2021) A primer for decentralized identifiers: An introduction to self-administered identifiers for curious people. [Online]. Available: https://w3c-ccg.github.io/did-primer/ 5

[19] E. Lear, R. Droms, and D. Romascanu, “Manufacturer Usage Description Specification,” RFC 8520, Mar. 2019. [Online]. Available: https://www.rfc-editor.org/info/rfc8520 6

[20] C. Herder, M.-D. Yu, F. Koushanfar, and S. Devadas, “Physical unclonable functions and applications: A tutorial,” *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1126–1141, 2014. 6

[21] S. Vinagrero, H. Martin, A. de Bignicourt, E.-I. Vatajelu, and G. Di Natale, “Sram-based puf readouts,” *Scientific Data*, vol. 10, no. 1, p. 333, 2023. 6, 7, 10

[22] D. E. Holcomb, W. P. Burleson, K. Fu *et al.*, “Initial sram state as a fingerprint and source of true random numbers for rfid tags,” in *Proceedings of the Conference on RFID Security*, vol. 7, no. 2, 2007, p. 01. 7

[23] C. Consortium. (2023) active security for connected devices lifecycles. [Online]. Available: https://certify-project.eu/ 9, 21, 23, 35, 51

[24] S. R. Niya, B. Jeffrey, and B. Stiller, “Kyot: Self-sovereign iot identification with a physically unclonable function,” in *2020 IEEE 45th Conference on Local Computer Networks (LCN).* Sydney, NSW, Australia: IEEE, Nov 2020, p. 485–490. [Online]. Available: https://ieeexplore.ieee.org/document/9314816/ 10

[25] H. Foundation. Hyperledger fabric documentation. [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/latest/ 11, 12

[26] Iotex: Connecting the real world to web3. [Online]. Available: https://iotex.io/ 11, 14

[27] NIST, "Nist glossary." [Online]. Available: https://csrc.nist.gov/glossary 49, 50

[28] [Online]. Available: https://www.techtarget.com/iotagenda/definition/Internet-of-Things-IoT 49

[29] E. Lear and V. Andalibi. Example mud json file. [Online]. Available: https://www.mudmaker.org/examples.html 55, 59

Abbreviations

AAA	Authentication, Authorization, and Accounting
ACL	Access Control List
CCS	Connected Cabin System
CRP	Challenge-Response Pair
CTIS	Cyber Threat Information Sharing
DAA	Designated Accrediting Authority
Dapp	Decentralized Application
DIAM	Decentralized Identity and Access Management
EU	European Union
EVM	Ethereum Virtual Machine
GDPR	General Data Protection Regulation
HMI	Human Machine Interface
IAM	Identity and Access Management
IDS	Intrusion Detection System
IETF	International Engineering Task Force
IFE	In-flight Entertainment System
IPS	Intrusion Prevention System
IoT	Internet of Things
LRU	Line Replacable Unit
MUD	Manufacturer Usage Description
NIST	National Institute of Standards and Technology
OTA	Over the Air
PHM	Prognostics and Health Management
PKI	Public Key Infrastructure
PUF	Physically Unclonable Function
SCADA	Supervisory control and data acquisition
SRAM	Static Random-Access Memory
SSI	Self-Sovereign Identity
TEE	Trusted Execution Environment
TOE	Target of Evaluation
VC	Verifiable Credential

Glossary

Attestation It is the claim by one entity over another, e.g, the driver license is an attestation by the Motor Vehicle Department (US), that one may drive a car. [13]

Authentication Verifying the identity of a user, process, or device, often as a prerequisite to allowing access to resources in an information system. [27]

Accounting Access privileges granted to a user, program, or process or the act of granting those privileges. [27]

Authorization Authorization is the decision whether an entity is allowed to perform a particular action or not, e.g., whether a user is allowed to attach to a network or not.

Cloud Computing Cloud computing is the on-demand availability of computer system resources, especially data storage and computing power, without direct active management by the user. [27]

Edge Computing Edge computing is the placement of storage and computing resources closer to source, where the data is generated. [27]

Firmware Computer programs and data stored in hardware - typically in read-only memory (ROM) or programmable read-only memory (PROM) - such that the programs and data cannot be dynamically written or modified during execution of the programs. [27]

IoT Internet of Things refers to an interconnected infrastructure, featuring objects, people, systems and information resources, where with intelligent services the processing of physical information and the virtual world can react. [28]

Line-Replaceable Unit modular component of airplane, designed to be replaced quickly

Fog Computing As an extension of Cloud computing, Fog Computing brings the computation closer to IoT Edge devices. [27]

Manufacturer Usage Description A component-based architecture specified in Request for Comments (RFC) 8520 that is designed to provide a means for end devices to signal to the network what sort of access and network functionality they require to properly function. [27]

Self-Sovereign Identity Defined as a portable digital ID that exists throughout the lifetime of a device and does not depending on a central authority. [17]

Trust Model In the trust model the issuer issues credential to a holder while the holder can prove identity by showing the credential to a verifier.

Trusted Execution Zone An area or enclave protected by a system processor. [27]

List of Figures

List of Tables

List of Code Snippets and Examples

Appendix A

Installation Guidelines

Appendix B

Code Snippets and Examples

Code 1: Example MUD file [29]

```
{
  "ietf-mud:mud": {
    "mud-version": 1,
    "mud-url": "https://iot-device.example.com/dnsname",
    "last-update": "2019-01-15T10:22:47+00:00",
    "cache-validity": 48,
    "is-supported": true,
    "systeminfo": "This is an example of a device that just wants to talk
                to its cloud service",
    "mfg-name": "Example, Inc.",
    "documentation": "https://iot-device.example.com/doc/dnsname",
    "model-name": "dnsname",
    "from-device-policy": {
      "access-lists": {
        "access-list": [
          {
            "name": "mud-96898-v4fr"
          },
          {
            "name": "mud-96898-v6fr"
          }
        ]
      }
    },
    "to-device-policy": {
      "access-lists": {
        "access-list": [
          {
            "name": "mud-96898-v4to"
          },
          {
            "name": "mud-96898-v6to"
          }
        ]
      }
    }
  },
  "ietf-access-control-list:acls": {
    "acl": [
      {
        "name": "mud-96898-v4to",
        "type": "ipv4-acl-type",
        "aces": {
          "ace": [
            {
              "name": "cl0-todev",
              "matches": {
                "ipv4": {
                  "ietf-acldns:src-dnsname": "cloud-service.example.com"
                }
              },
              "actions": {
                "forwarding": "accept"
              }
            }
          ]
        }
      },
      {
        "name": "mud-96898-v4fr",
        "type": "ipv4-acl-type",
        "aces": {
          "ace": [
            {
              "name": "cl0-frdev",
              "matches": {
                "ipv4": {
                  "ietf-acldns:dst-dnsname": "cloud-service.example.com"
                }
              },
              "actions": {
                "forwarding": "accept"
              }
            }
          ]
        }
      },
      {
        "name": "mud-96898-v6to",
        "type": "ipv6-acl-type",
        "aces": {
          "ace": [
            {
              "name": "cl0-todev",
              "matches": {
                "ipv6": {
                  "ietf-acldns:src-dnsname": "cloud-service.example.com"
                }
              },
              "actions": {
                "forwarding": "accept"
              }
            }
          ]
        }
      },
      {
        "name": "mud-96898-v6fr",
        "type": "ipv6-acl-type",
        "aces": {
          "ace": [
            {
              "name": "cl0-frdev",
              "matches": {
                "ipv6": {
                  "ietf-acldns:dst-dnsname": "cloud-service.example.com"
                }
              },
              "actions": {
                "forwarding": "accept"
              }
            }
          ]
        }
      }
    ]
  }
}
```

Code 2: Example VC Document[10]

```
{
  "@context": [
    "https://www.w3.org/ns/credentials/v2",
    "https://www.w3.org/ns/credentials/examples/v2"
  ],
  "id": "http://university.example/credentials/1872",
  "type": ["VerifiableCredential", "ExampleAlumniCredential"],
  "issuer": "https://university.example/issuers/565049",
  "validFrom": "2010-01-01T19:23:24Z",
  "credentialSubject": {
    "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
    "alumniOf": {
      "id": "did:example:c276e12ec21ebfeb1f712ebc6f1",
      "name": "Example University"
  },
  "proof": {
    "type": "DataIntegrityProof",
    "cryptosuite": "eddsa-2022",
    "created": "2023-06-18T21:19:10Z",
    "proofPurpose": "assertionMethod",
    "verificationMethod": "https://university.example/issuers/565049#key-123",
    "proofValue": "zQeVbY4oey5q2M3XKaxup3tmzN4DRFTLVqpLMweBrSxMY2xHX5XTYV8nQApmEcqaqA3Q1
  }
}
```

Code 3: Example DID Document[17]

```
{
  "@context": [
    "https://www.w3.org/ns/did/v1",
    "https://w3id.org/security/suites/ed25519-2020/v1"
  ],
  "id": "did:example:123456789abcdefghi",
  "authentication": [{
    "id": "did:example:123456789abcdefghi#keys-1",
    "type": "Ed25519VerificationKey2020",
    "controller": "did:example:123456789abcdefghi",
    "publicKeyMultibase": "zH3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV"
  }],
  "proof": {
    "type": "Ed25519Signature2018",
    "created": "2020-04-22T10:37:22Z",
    "proofPurpose": "assertionMethod",
    "verificationMethod": "did:example:456#key-1",
    "jws": "eyJjcml0IjpbImI2NCJdLCJiNjQiOmZhbHNlLCJhbGciOiJFZERTQSJ9..BhWew0x-txcroGjgdt
  },
  "created": "2020-04-22T10:37:22Z",
  "updated": "2020-04-23T10:37:22Z",
  "service": [{
    "id":"did:example:123#linked-domain",
    "type": "LinkedDomains",
    "serviceEndpoint": "https://bar.example.com"
  }]
}
```