Improving the Identity and Access Management Capabilities of Industrial Internet of Things

Akanksha Dixit

Supervisor: Prof. Muttukrishnan Rajarajan

A Thesis Submitted to
School of Science and Technology, City, University of London, UK

In Partial Fulfilment of the Requirement for the Degree Doctor of Philosophy (PhD)
In Information Engineering

September 2024

Declaration

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

Akanksha Dixit
September 2024

Acknowledgement

The last four years have been a journey to remember. It has been a journey that demanded the utmost discipline, hard work and resilience from me. This thesis would not have come to fruition without the selfless and unwavering support of a considerable number of people. This section is intended as a small gesture from my side to thank the following people.

I would first like to thank my PhD supervisor, Professor Muttukrishanan Rajarajan. When I first applied to City in late 2019, Professor Raj kindly accepted to mentor me as his student. He advised and answered my questions at each step from publishing my first conference paper as a PhD student to motivating me to present my research at various forums to gather a holistic insight in my domain. He has always acted as a mentor and shared his wisdom whenever I was confronted with a roadblock in my research. His unparalleled work ethic and sincere dedication to work have always inspired me. Thank you, Professor.

I have been fortunate to have the support of amazing fellow researchers through this PhD. The following deserve thanks for their help and guidance: Dr Waqar Asif for his guidance during the initial days of my PhD; Dr Max Smith-Creasey for his help and support throughout my internship at BT and beyond; Dr Yogachandran Rahulamathavan for his support in the data marketplace work; Dr Bruno Bogaz Zarpelo for the invaluable discussion during our paper collaboration, giving comments on my thesis and beyond. My friends Shahrbano, Safwana, Subhajit and Veniamin at the City and my friends Vijaya, Subhra, Manish and Sangita back home.

At every stage of my research, through good days and bad, my family stood by my side as my rock. My mother and father always believed in the importance of education and they always went out of their way to support me in my academic endeavours. My siblings Anketa and Harsh, have always been my greatest champions. Thank you all for believing in me more than I did myself. I would lastly like to thank my partner (now husband) Arjun. He has been my strongest support, collaborator and guide in many ways than I could have imagined. When I decided to move abroad for my PhD studies, he supported me with the biggest heart and smile. Thank you for always being there.

Abstract

Internet-of-Things is one of the most widely used technologies in the world. It is used in personal spaces in the forms of virtual assistant technologies, home sensors, baby monitors etc. and in enterprise/public settings to monitor the health and safety of citizens. But the ecosystem that has seen the most significant disruption due to this technology is industry, supply chain and retail management. In these settings, IoT provides monitoring of very specific parameters. It is worth noting that the mode and purpose of using IoT in an industrial setting also known as *Industry 4.0* are quite different from the above-mentioned use cases. IoT devices in an industrial setting are quite mobile, distributed across different networks and often handled by various operators. These devices collect critical data that could cause large-scale damage if obtained by attackers or insider malicious entities. Recently, traditional authentication and authorization techniques have been shown to have flaws that enable attackers to bypass them. Furthermore, traditional authentication schemes are designed with a client-server setup which does not scale well with the growing number of IoT devices. These systems are highly centralized which makes it difficult to scale and protect against large-scale attacks as the single root of trust is compromised. Another operational challenge includes the use of static *one-time-off* authorization mechanism which enables attackers to cause damage if they maliciously gain access to the system. These challenges not only deter the secure functioning of current forms of IoT networks but also creates hindrance in the evolution of IoT networks to support more collaborative use cases that entail device sharing, IoT data sharing and peer-to-peer transactions to support innovative use cases.

While significant progress has been made in adopting existing connectivity, identity and access management frameworks for IoT, most of these frameworks are designed to work for unconstrained devices in centralized settings. IoT devices are usually constrained with an inherent tendency to operate in a decentralized and peer-to-peer arrangement. Therefore, the thesis utilises the concepts of decentralization introduced in Distributed Ledger (DLT) technologies and the capability of automating business flows through smart contracts. This thesis focuses on advancing the traditional identity and access management techniques to enable scalable, decentralized and secure identification of devices by providing novel

mechanisms. In order to achieve this, the thesis proposes four novel contributions described in the following:

The first contribution is a smart contract-enabled decentralized identity management framework for Industry 4.0. A novel methodology to create and manage identities and access control using the autonomous functioning of smart contracts on distributed ledgers is proposed. This framework boasts inherent decentralization and scalability, addressing a critical gap in the existing IIoT management landscape, which is largely dominated by centralized models.

The second contribution provides an improvement on the previous contribution in the form of a decentralized IIoT identity framework based on self-sovereign identity. While the core principle of decentralized identity management endures, the innovative model introduces a robust decoupling between the roles of issuer, holder, and verifier. This strategic evolution seeks to reconcile the decentralized digital identity concept, introduced in our prior work, with the emerging standard of Self-Sovereign Identity (SSI).

The third contribution is a fair, secure and trusted decentralized IIoT data marketplace enabled by blockchain. The insights derived from their first two contributions were ingeniously harnessed in this contribution, leading to the consolidation of their expertise in exploring a forward-looking application of IoT within the context of recent advances in AI/ML-driven innovations. Through this design, we explore the parameters like trust, fairness and fault tolerance of such a design pattern.

The last contribution extends and improves some aspects of the previous contribution. Existing data exchange schemes depend on a trusted third party for key management during authentication and rely on a ‘one-time-off’ approach to authorization. This contribution proposes a user-empowered, privacy-aware, authentication and usage-controlled access protocol for IIoT data marketplace.

Publications

The following publications have been produced from the research conducted in this thesis.

International Conference Papers

- **Publication I**: **Akanksha Dixit**, W. Asif and M. Rajarajan, "Smart-Contract Enabled Decentralized Identity Management Framework for Industry 4.0," IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society (IECON), Singapore, 2020, pp. 2221-2227, doi: 10.1109/IECON43393.2020.9254545.

- **Publication II**: **Akanksha Dixit**, M. Smith-Creasey and M. Rajarajan, "A Decentralized IIoT Identity Framework based on Self-Sovereign Identity using Blockchain," 2022 IEEE 47th Conference on Local Computer Networks (LCN), Edmonton, Canada, 2022, pp. 335-338, doi: 10.1109/LCN53696.2022.9843700.

Journal Papers

- **Publication III**: **Akanksha Dixit**, A. Singh, Y. Rahulamathavan and M. Rajarajan, "FAST DATA: A Fair, Secure and Trusted Decentralized IIoT Data Marketplace enabled by Blockchain," in IEEE Internet of Things Journal, 2021, doi: 10.1109/JIOT.2021.3120640.

- **Publication IV**: **Akanksha Dixit**, B.B. Zarpelao, M. Smith-Creasey and M. Rajarajan, "A privacy-aware authentication and usage-controlled access protocol for IIoT decentralized data marketplace", in Computers & Security, Volume 146, 2024, doi: 10.1016/j.cose.2024.104050.

Abbreviations

ABAC
Attribute-Based Access Control
ABI
Application Binary Interface
ACE
Authentication and Authorization for Constrained Environments
ACL
Access Control List
AES
Advanced Encryption Standard
API
Application Programming Interface
BFT
Byzantine Fault Tolerant
BOPS
Biometric Open Protocol Standard
CA
Certificate Authorities
CBC
Cipher-Block Chaining
CCPA
California Consumer Privacy Act
CFT
Crash Fault Tolerant
CoAP
Constrained Application Protocol
CPU
Central Processing Unit
CRL
Certificate Revocation List
CTAP
Client to Authenticator Protocol
DApp
Decentralized Application
DB

Database

DBMS

Database Management Systems

DeFi

Decentralized Finance

DIDs

Decentralized Identifiers

DoS

Denial-of-Service

DDoS

Distributed Denial-of-Service

DLT

Distributed Ledger Technology

DPKI

Distributed Public Key Infrastructure

EOA

Externally Owned Account

ECDSA

Elliptic Curve Digital Signature Algorithm

ERC

Ethereum Request for Comment

EVM

Ethereum Virtual Machine

FIDO

Fast ID Online

GCP

Google Cloud Platform

GDPR

General Data Protection Regulation

GPS

Global Positioning System

HDD

Hard Disk Drive

HIPAA

Health Insurance Portability and Accountability Act

HMAC

Hash-based Message Authentication Code

HTTP

Hypertext Transfer Protocol

IAM

Identity and Access Management

ICANN

Internet Corporation for Assigned Names and Number

IDAMS

Identity and Access Management System

IdM

Identity Management

IIoT

Industrial Internet of Things

IoT

Internet of Things

IP

Internet Protocol

IPv6

Internet Protocol Version 6

IPFS

Inter-Planatery File System

JSON

JavaScript Object Notation

JSON-RPC

JavaScript Object Notation- Remote Procedure Call

KYC

Know Your Customer

LwM2M

Lightweight Machine-to-Machine

MAC

Message Authentication Code

MITM

Man-in-the-Middle

M2M

Machine-to-Machine

MSP

Membership Service Provider

MQTT

Machine Queuing Telemetry Transportation

NFT

Non-Fungible Token

OEM

Original Equipment Manufacturer

OPC-UA

Open Platform Communications United Architecture

P2P

Peer-to-Peer

PBFT

Practical Byzantine Fault Tolerant

PGP

Pretty-Good Privacy

PII

Personally Identifiable Information

PKC

Public Key Certificate

PKI

Public Key Infrastructure

PoC

Proof-of-Concept

PoS

Proof-of-Stake

PoW

Proof-of-Work

RAFT

Reliable, Replicated, Redundant, And Fault-Tolerant

RAM

Random Access Memory

REST

Representational State Transfer

RFID

Radio-frequency identification

RPC

Remote Procedure Call

SAML

Security Assertion Markup Language

SDK

Software Development Kit

SHA

Secure Hash Algorithm

SSI

Self-Sovereign Identity

SSL

Secure Sockets Layer

TLS

Transport Layer Security

TTP

Trusted Third Party

UCON

Usage Control

URI

Uniform Resource Identifier

UUID

Universally Unique Identifier

VCs

Verifiable Credentials

VCR

Verifiable Credential Registry

VDR

Verifiable Data Registry

W3C

World Wide Web Consortium

WSN

Wireless Sensor Network

XAML

Extensible Access Control Markup Language

ZT

Zero Trust

Table of Contents

List of Figures

Listings

List of Tables

Chapter 1

Introduction

IoT has emerged as one of the most important and widely deployed computing paradigms over the past years. The term *Internet of Things* was initially used to refer to "connected data-collecting hardware". It had few potential use cases back then and was merely a way of collecting sensor data from the environment. However, given a decade-long course of technological development dedicated to IoT-specific protocols, network infrastructure, cloud-based platforms, endpoints and hardware (processors etc.), IoT has made its way to technologically advanced applications such as healthcare, connected vehicles, and supply-chain. This dissertation adopts a rather advanced definition of the term *Internet of Things* which indicates the capability to connect any physical object to the Internet in a decentralized way to cater for innovative enterprise use cases.

The increased proliferation of IoT devices has given rise to new use cases and business models. As shown in Figure 1.1, the spending on Enterprise IoT alone is expected to grow from 158 billion USD to a whopping 525 billion USD in 2027. Recently a plethora of new opportunities has emerged in a wide spectrum of domains on how data and services offered by IoT devices can be monetized. Several startups [1] are taking the lead in innovative IoT products with big organizations such as Cisco, Bosch, Philips, [2], [3] following the lead to capture the market. Corporations such as Amazon, Google, and Thinger [4] are also providing cloud-based IoT platforms for data logging and analytics. These trends indicate the rising supply-demand cycle in the IoT ecosystem.

Whilst IoT as a technology is being widely adopted, its security has long been inherently overlooked. As seen in Figure 1.1, in 2021, out of the total 158 billion USD spent on Enterprise IoT only 4% was spent on security. IoT security can be achieved at four different levels as shown in Figure 1.2. Although readiness at each level ensures distinct security guarantees, securing the device itself is a key step towards the security of the overall ecosystem. According to an industry report prepared by collecting data from key players

[1] https://iot-analytics.com/iot-market-size/

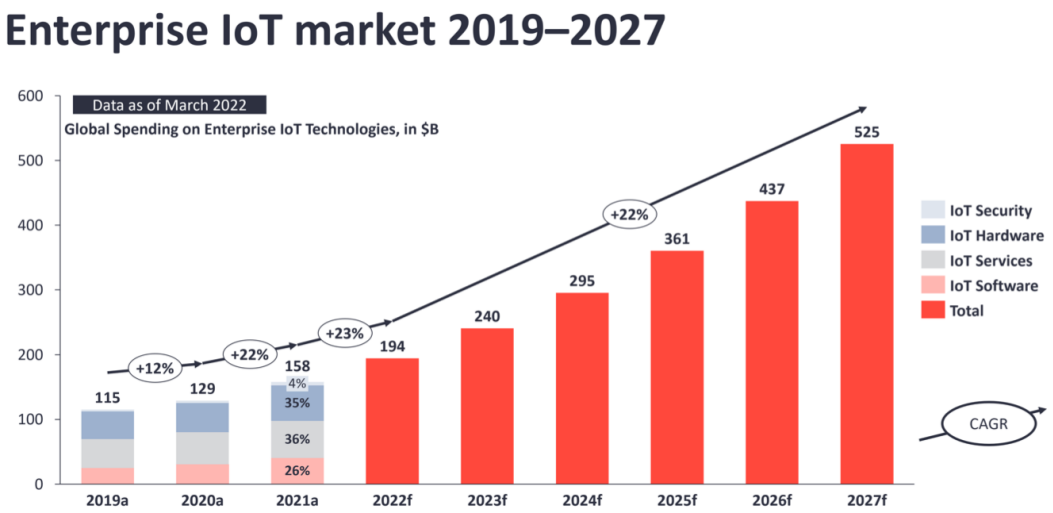


Fig. 1.1 Showing the estimated enterprise IoT market growth between 2019-2027 with CAGR (Compound Annual Growth Rate) [Source: IoTAnalytics][1]

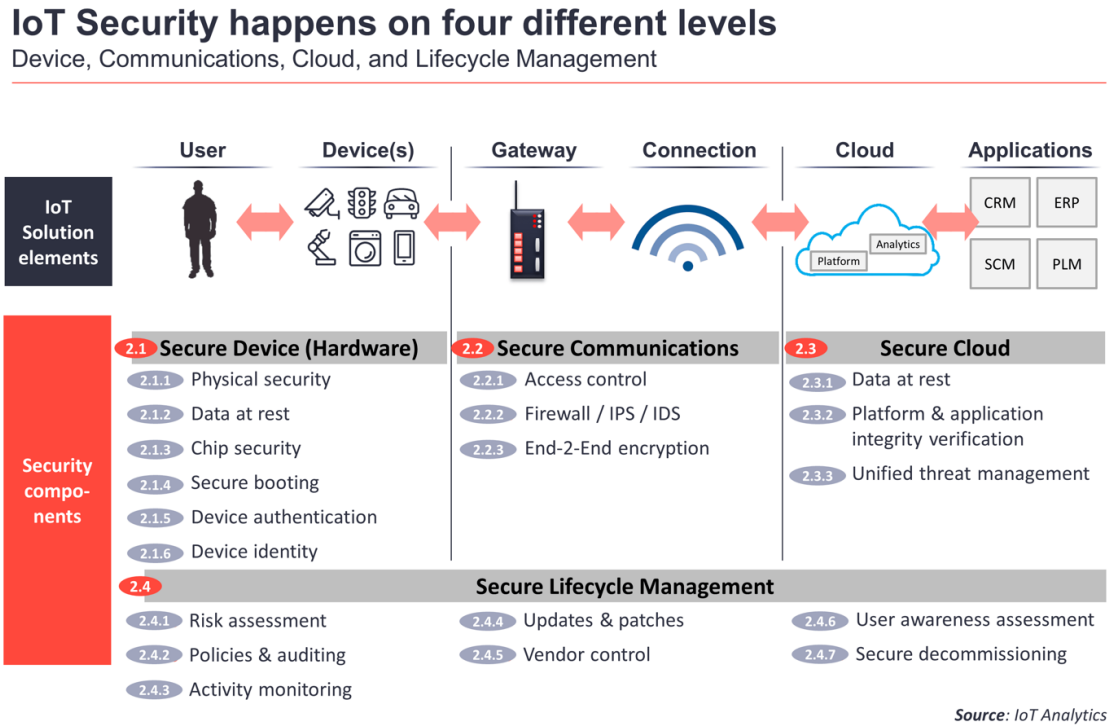


Fig. 1.2 Showing the different levels of IoT security at Device, Communication, Cloud and Lifecycle management and their further subcomponents [Source: IoTAnalytics]

in the IoT market, as shown in Figure 1.3, among all the areas of security concerns, the greatest need for improvement lies in the area of authentication/authorization process, with access control being the second one. A fundamental requirement of secure digital communication is authentication and authorisation of legitimate entities. This is possible with strong identifiers in place at the beginning and their management throughout the device life cycle

Fig. 1.3 Showing the result of a survey illustrating the ranking of a range of IoT security weakness that need greatest improvement [Source: IoT Analytics][1]

1.1 Motivation

IoT devices have rapidly become a ubiquitous part of modern industry. We have witnessed a rise in the use of interconnected devices for various applications like in healthcare, supply chain, smart home, power systems etc. The continuous development of capabilities and applications of IoT devices have made them a platform for both personal and business use.

Industry 4.0 is characterised by the integration of digital technologies into production processes powered by IoT-based advancements. Smart factories equipped with IoT sensors and robotics enable efficient manufacturing operations. IoT facilitates predictive maintenance of machinery, minimising downtime and optimising productivity. Additionally, IoT-powered production system leverages the abundance of high quality data in processing and decision making.

[1] https://www.iiot-world.com/news/reports/an-overview-of-the-iot-security-market-report-2017-2022/

The increased adoption of IoT devices means that they are deeply embedded in and around critical processes. This indicates that there lies an incentive to hack them and gain access to sensitive data such as financial details in the case of mobile devices, infrastructure and sensitive manufacturing data details in the case of industries, and critical health data in the case of healthcare devices. The dynamic and heterogeneous nature of IoT networks can make them prone to attacks if not safeguarded adequately [5].

The thesis primarily explores the challenges with the current state-of-the-art identity management practises in IoT and how it hinders the evolution of novel use-cases to help monetize the IoT data. The prevalent practice of assigning default username-passwords to devices at the bootstrapping stage is a weak authentication mechanism. In industrial setting, this mechanism is generally replaced by use of digital certificates as device identifiers to provides a greater level of security.

There are various inherent weaknesses with the use of complex digital certificates in IoT platforms. Traditional identity servers are employed to maintain the bulk of such certificates, thus they are lucrative assets for malicious entities and therefore potential attack vectors. There is significant financial gain for hackers in compromising such a server [6]. Maintaining such infrastructure is also an arduous task as it requires frequent update of identifiers in case of the root certificate compromise/expiry.

Another challenge is that the vast majority of classical IoT platforms available today are based on a centralized model where the platform provider is trusted with authorization and access control [7]. These solutions are heavily centralized around the cloud framework which creates a bottleneck in real-time data access and enabling access control policies for smart devices [8]. Centralized databases affect scalability and does not allow for peer-to-peer verification. The devices can only operate in their identity silos and the scope of interoperability is limited. This also leads to the issue of vendor lock-in for businesses.

Furthermore, current IoT authentication mechanisms are *one-time off* systems; they do not monitor the communication after the initial device verification. This puts hacked devices at risk of data theft and in case of a malicious device, it gives the device more time to hide under the disguise and cause damage to the network. In order to ensure that imposters do not cause great damage if they happen to get on-time access, there would need to be a mechanism in place to authenticate and monitor the behaviours of devices in the critical process beyond an initial check-in.

Another weakness of the current authentication and access control mechanism is that they reveal a lot of unnecessary information during the process. These current mechanisms do not focus on privacy and data minimization when it comes to designing identity lifecycle. This can lead to device footprint being available to platform owners which can be harvested by them for monetary benefits. Therefore, a transition is required in the way identities are created, managed and maintained in IoT networks [7].

IoT device data generated is a valuable resource with reusable vale. The IoT devices installed in various settings can help build applications that consume various types of data and generate helpful insights. This is only possible if this data can be exchanged/traded in a secure and fair setting. Data marketplaces are one such opportunity to monetize data and reuse its value outside the context in which it was generated for sustained value extraction.

The research of this thesis is motivated by the need for a secure, interoperable and privacy-preserving form of identity management aided by the advancements of new and novel authentication techniques in the context of IoT devices. The evolution and adoption of such an identity and access management system is presented in the IoT data marketplace context.

1.2 Research Challenges

As IoT becomes ubiquitous, there emerges a wide spectrum of novel use cases to monetize IoT data and the associated services [9], [10], [11]. New device-sharing and data-sharing models [12] are emerging that demand a new approach to how identities and their management is carried out in the modern IDAMS. In a dynamic enterprise IoT setting, multiple stakeholders collaborate to share devices, data and network infrastructure in an evolving ecosystem. The legacy access control mechanism is not designed to account for granular permission levels and fails to detect real-time threats. The static authentication techniques can only protect systems at the entry and do not consider the subject's behaviour once admitted [13]. The primary technical challenges associated with the "Secure Device" identity and access management paradigm as discussed above can be listed as follows:

- *Issues with devices identifiers (Repetitive/ Identity Theft):* Unique identifiers are crucial to ascertain the identity of a device on the network. Default username-password combinations, weak passwords and non-standard identifiers for IoT devices are the root cause of a large proportion of attacks on IoT networks [14], [5], [15]. The different ways of identifying IoT devices paired with large-scale deployments of the various IoT platforms have made it increasingly difficult to uniquely identify a device on a global scale. We need new forms of identifiers that are unique, interoperable and cannot be stolen by compromise of a central entity.

- *Fragmented Identity Silos/Honeypot Databases in IDAMS:* [16], [6] When enterprises onboard IoT devices, they follow a nomenclature strategy based on their characteristic requirements. These requirements may not overlap with that of other enterprises and hence create identity silos. These fragmented identity silos consequently lead to creation of centralized ID databases. With large-scale adoption

of cloud-based IoT platforms, centralized repositories of username-passwords that often store this information in unencrypted format are on the rise. Such platforms are only a hack away from revealing information of thousands to millions of devices. There have been several attacks in the past [17], [15] with the intent to steal information about IoT devices/personal devices to hack into a network. Hence, what was originally designed to be an identity-related repository is often a lucrative target of hackers making them the "honeypot" of sensitive data.

- *Interoperability & Poor Scalability:* [18], [8] Due to the heterogeneous nature of IoT networks, it is not feasible to extrapolate the web-based solutions to an IoT system. The proliferation of different IoT systems and lack of their interoperability have withheld the emergence of novel IoT use cases that demand interoperability on the application level. Lack of interoperability leads to poor scalability of solutions. Due to a lack of interoperable identity and access control solutions for IoT it is very difficult to design secure solutions that can scale.

- *Poor Access Control:* [19] In a real-world collaborative IoT system where multiple stakeholders engage in an exchange as well as sharing of information and resources, proper access control management is crucial. However, the legacy access control mechanism [20] is not designed to account for granular permission levels and fails to detect real-time threats. Static and 'one-time-off' authentication techniques do not safeguard against malicious entities that accidentally get access to the network [21].

- *Peer-to-Peer services:* [22], [11] Modern Peer-to-peer IoT scenarios such as self-driving vehicles, electrical-charging kiosks, IoT data marketplaces, and autonomous cargo-fleets inspire novel ways of interacting with machines in an M2M economy. This will include ways in which identities and access control between devices/machines are designed, managed and maintained over their lifecycle.

As discussed above, IoT faces a number of critical challenges that hinder its full-scale adoption in emerging dynamic use cases. This dissertation focuses on finding solutions to the challenges associated with identity and access management practises in IoT and how they can motivate novel use cases that are emerging in the computing paradigm. IoT networks are inherently distributed in nature, therefore improving their capabilities requires a novel approach to onboarding, assigning and maintaining their identity lifecycle. This is a challenge that can be approached from different perspectives. Considering the fairly large number of research challenges in this area, this dissertation identifies a set of most relevant challenges that affect the security of IoT networks right from the beginning i.e. device identity, device authentication and device authorization. The objective of this work is to design, propose and analyse the feasibility of a decentralized framework for the identity

management of IoT networks. For designing such an architecture, we leverage Distributed Ledger Technology (DLT) in specific scenarios. The use of smart contracts (automated computer codes that execute according to the terms of the agreement) is proposed to eliminate the requirement of a trusted third party to act on behalf of IoT entities. The use of smart contracts to enhance privacy, security and authenticity of data exchange along with designing smart peer-to-peer applications is studied. Later, this dissertation also delves into the issues surrounding static authorization techniques and threats associated with such techniques.

1.3 Research Objectives

As detailed in the previous section, IoT faces a number of critical challenges that hinder its full-scale adoption across various industries. IoT networks are inherently distributed in nature, therefore improving their capabilities requires a novel approach to onboarding, assigning and maintaining their identity lifecycle. Considering the fairly large number of research challenges in this area, this dissertation identifies a set of most relevant challenges that affect the security of IoT networks right from the beginning i.e. device identity, device authentication and device authorization. The objective of this work is to design, propose and analyse the feasibility of a decentralized framework for the identity management of IoT networks. For designing such an architecture, we leverage DLT in specific scenarios. DLT network can provide several benefits when connected to IoT network. In specific IoT use cases that demand mutual trust and transparency by removing a centralized entity, DLT can offer transparency and auditable transactions. This can lead to improved network resiliency, reduced single points of failure, and increased trust among participants. The use of smart contracts (automated computer codes that execute according to the terms of the agreement) is proposed to eliminate the requirement of a trusted third party to act on behalf of IoT entities. The use of smart contracts to enhance privacy, security and authenticity of data exchange along with designing smart peer-to-peer applications is studied. This dissertation concentrates on following five specific goals:

Objective 1: Decentralized identity and access management framework for IIoT

- To design and adapt new forms of global identifiers for IoT devices that provides unique addressability, avoid single point of failure and siloed identity verticals. To design a decentralized framework for managing the lifecycle of such identifiers and devise access management policies that can function without a centralized authority.
- To explore and adapt advanced identity related standards like SSI (DIDs and VCs) for industrial IoT use-cases while ensuring decentralization and compatibility with the state-of-the-art IoT protocols.

- To develop authentication and authorization protocols for the proposed framework that not only ensures backward compatibility but also support granular control, continuity of decision and continuous monitoring of resources being accessed.

Objective 2: Representation of Complex Industrial IoT use-cases

- To explore and develop proof-of-concept for above stated objective 1. The proof-of-concept (PoC) implementations are developed for complex and novel IoT use-cases that demand decentralized architecture for managing identity and permissions for devices and users.
- To design and develop efficient device discovery and claim presentation mechanism for remote device access using machine to machine communication. It extends the capabilities of machines to function autonomously in complex real case operations. Thus the distributed devices that constitute the network can manage operations like verification/presentation of claims and data signing autonomously.
- To design and develop a fair, secure and trusted decentralized IoT data trading marketplace. To ensure that such a marketplace supports user verification, trust metric calculation, fair payment settlement, fault-tolerance and secure exchange of IoT data. To implement and test such a data trading framework.

Objective 3: Interoperability

- To design and develop identity management framework that provide interoperability of device identifiers. Such a framework should allow device identifiers to be recognizable beyond a specific platform and allow the rotation of such identifiers. It should allow claims of the device to be presentable and verifiable in an interoperable format.
- To design and develop authentication and authorization protocols that are interoperable across vendors and/or platforms for managing device identity lifecycle.

Objective 4: Scalability

- To design and architect the aforementioned identity and access management framework in such a way that it is able to scale with the increased device and concurrent request load.
- To design and develop the architecture for measuring the load scalability of the proposed frameworks in terms of operations it is able to perform when increasing or decreasing the number of the distributed entities/concurrent request load and other relevant parameters.

Objective 5: Security

- To tightly and seamlessly integrate the security mechanism with authentication protocols in the framework in such a way that it causes minimum overhead for the throughput and latency in the communication.
- To design and develop a security protocol for preventing unauthenticated and unauthorised actors from gaining admission to the proposed frameworks and compare its performance with state-of-the-art protocols. The protocols should provide continuous monitoring of resources as opposed to a 'one-time off' solution to ensure enhanced security.
- To design and develop inbuilt security mechanism in the proposed framework that ensures the integrity, trust and confidentiality of the devices/users of an application deploying our identity and access management framework.

1.4 Research Methodology

This dissertation embraces different research methodologies to evaluate the author's ideas and claims. First, a preliminary literature survey was conducted that helped to identify the set of most significant challenges in the area of secure IoT computing from the IDAMS perspective. For each challenge, a framework was designed, implemented as a software prototype and finally tested. Each successive design/framework was an improvement over the previous one. To ascertain the feasibility and performance of the design, its software prototype was analyzed using repeated simulations under controlled environments. Therefore, the research methods applied in this dissertation draw inspiration from software design, system design, proof-of-concept (PoC) development and performance analysis areas of computing.

The objective of PoC development was to have an independent and modular library that can be easily replicated and improved over time. A major part of this work consists of designing and implementing the PoC. Wherever required the design extends or modifies the open-source libraries and projects to reuse the specific functionality implemented by them. However, most of the components were implemented fully as they had to achieve a very specific functionality being targeted in the dissertation. When open-source code was used, the author contributed to the open-source community on various levels by being involved in community discussion, raising issues and bug reporting. The software prototypes developed in this dissertation are open-source too and available on the online code-management platform Github [23].

In order to evaluate the design, many concurrent programs had to be designed to understand and measure the system's performance. Most of the evaluations, therefore, had to be run on cloud platforms as they provided upgraded hardware support. Linux operating system was selected for PoC development and evaluation on these cloud platforms. Linux OS supports most of the system libraries and makes the extension of a project much more effortless. Concurrency was a major design feature in the project evaluation as IoT devices were modelled as concurrent services requesting a resource.

Each solution was designed by using standard IoT communication and data-sharing protocols. Following the standard development practices helps to ensure the reusability and interoperability of the solutions along with their easy adoption and acceptance by the open-source community. Specific databases more suitable to the decentralized design of IoT networks were selected to seamlessly integrate the solution.

To design a complex working model of any system, experimental computer science methods [24] can prove effective in evaluating the hypothesis. The experimental computer science methods are built on experimentation used industriously in this dissertation. It proposes to evaluate and validate a solution using the construction of a prototype system. The results, frameworks and mechanisms designed in this dissertation together are difficult to represent in a mathematical fashion, as prototyping was the only type of holistic tool that could be used to represent and analyse the research problem in a comprehensive manner. However, wherever possible, system components and their respective designs have been represented in the form of mathematical formulae, derivations and computer algorithms.

In summation, the scientific process used in each publication comprises the following steps [24]:

Problem → Hypothesis → Implementation → Evaluation → Conclusion

First, a problem statement is defined after conducting a relevant literature survey. Subsequently, the solution hypothesis is designed as an architecture or theoretically functioning framework. A PoC prototype is implemented as a modular software library. This software library is evaluated under various parameters to study its performance in an IoT landscape. The work is detailed in the form of a research paper. The conclusion along with observations is published in international scientific publications. Finally, the designed solution and research statement are re-evaluated according to the reviews received from the scientific community. As a final note, all the prototypes in this dissertation were fully designed, implemented and evaluated by the author herself and her research collaborators whose names are acknowledged in the acknowledgement section of this thesis.

1.5 Research Contribution

Designing a scalable, interoperable and decentralized identity and access management scheme for distributed networks such as IoT that is robust and resilient is challenging as it must operate consistently and securely in all scenarios. This thesis presents a collection of novel modules that build on top of each other to form a secure, scalable and resilient IoT network to enable novel real-world use cases. These modules in conjunction with each other fulfil the above-discussed objectives. This dissertation consists of four peer-reviewed publications that are the primary contribution of the author. The research was conducted as a progression and each successive contribution was taken up as an improvement to the previous contribution. Therefore, these contributions complement each other and provide a comprehensive secure IDAMS for the IoT ecosystem. Below is a brief overview of the contributions of each publication.

A novel smart-contract enabled framework for identity and access management of distributed IoT networks

In the domain of industrial IoT (IIoT) infrastructure, the prevalent practice of relying on centralized identity servers for authenticating IoT devices has given rise to a series of notable challenges, as previously discussed. This thesis represents a significant stride forward in addressing these challenges and makes distinctive contributions to the field. The primary contribution of this work introduces an innovative paradigm shift in the conception and design of identity and access management for IIoT setups. Unlike conventional approaches, this research advocates for the development of a decentralized framework, which marks a pioneering step in the domain. Moreover, it delves into the evaluation of the performance implications associated with such a design. This contribution has been published in [25].

The core of the contributions manifests in the form of a smart contract-driven, blockchain-based decentralized life-cycle and access management system. This system leverages the capabilities of the InterPlanetary system (IPFS) for decentralized data storage, distinguishing it from the incumbent IIoT management frameworks. Furthermore, this framework boasts inherent decentralization and scalability, addressing a critical gap in the existing IIoT management landscape, which is largely dominated by centralized models. The author designed this framework using standard IoT protocols and functionalities, and its efficacy was evaluated during implementation of crucial component that validated our initial hypothesis at the inception of this dissertation. This work is presented in Chapter 3 of the dissertation.

A decentralized digital identity framework extended using self-sovereign identity model

The idea of having a decentralized identity and access management framework for IIoT by leveraging a distributed ledger technology was explored in the previous contribution. However, certain challenges were identified in scaling such a model. Firstly, multiple smart-contract deployments and growing hierarchy in a multi-stakeholder environment meant greater latency. Secondly, scaling a multi-smart contract-based identity management system could also mean a higher cost of deployment. Thirdly, in order to scale this further and provide stronger security guarantees during authentication and authorization, substantial management complexities emerged.

In response to these challenges, the present contribution represents a pivotal advancement in the field. While the core principle of decentralized identity management endures, the innovative model introduces a robust decoupling between the roles of issuer, holder, and verifier. This strategic evolution seeks to reconcile the decentralized digital identity concept, introduced in our prior work, with the emerging standard of Self-Sovereign Identity (SSI). Self-sovereign identity, integrated with the novel utilization of Decentralized Identifiers (DIDs) and Verifiable Credentials (VCs), transcends the need for a centralized trust authority to maintain validation evidence. Our proposed model stands as a platform-agnostic solution, harmoniously integrable with both existing infrastructure and futuristic technologies like distributed ledger platforms. The practical implementation of this novel framework was carried out on two prominent blockchain platforms, namely Ethereum and Hyperledger Indy. This empirical study was instrumental in dissecting and comprehending the underlying overheads, further affirming the efficacy of our contributions in the domain of IIoT identity and access management. This has been published in [26]. This work is presented in Chapter 4 of the dissertation.

A fair, secure and trusted decentralized data marketplace enabled by blockchain

In this thesis, the insights derived from their first two contributions were ingeniously harnessed in this contribution, leading to the consolidation of their expertise in exploring a forward-looking application of IoT within the context of recent advances in AI/ML-driven innovations. A distinctive contribution emerged in the form of a novel decentralized digital data marketplace, carefully tailored for IoT data. The platform leverages a decentralized data streaming network to host IoT data in a reliable and fault-tolerant manner. Through this design, we explore the parameters like trust, fairness and fault tolerance of such a design pattern. The platform ensures fair trading, data storage and delivery in a privacy-preserving manner and trust metric calculation for actors in the network. In order to study the feasibility of the proposed platform, an open-source library is developed using Hyperledger Fabric and a data network layer built on VerneMQ, the library is deployed on a real-time Google Cloud platform. The library is tested and results are analysed for throughput, overheads and scalability. This contribution has been published in [22]. This work is presented in Chapter 5 of the dissertation.

A Privacy-Aware Authentication and Usage-Controlled Access Protocol for IIoT Decentralized Data Marketplace

While in the last contribution, we focussed on designing a fair, trusted and fault-tolerant data marketplace, the design lacked an in-depth exploration of authentication and authorization patterns. Decentralized data marketplaces allow the democratization of rates, trading terms and fine control to participants. However, in such a marketplace, ensuring privacy and security is crucial. Existing data exchange schemes depend on a trusted third party for key management during authentication and rely on a 'one-time-off' approach to authorization. This contribution proposes a user-empowered, privacy-aware, authentication and usage-controlled access protocol for IIoT data marketplace. The proposed protocol implements a dynamic user-revocation policy. Usage-controlled based access provides secure ongoing authorization during data exchange. In order to study the feasibility of the proposed platform, an open-source library is developed using Hyperledger Indy and Linux Utility-Cron Job to design a UCON system. The library is tested and results are analysed for time and storage cost, overheads and scalability. This work is presented in Chapter 6 of the dissertation and this contribution is published in [27].

1.6 Structure of the Dissertation

The remainder of this dissertation is structured as follows:

Chapter 2 provides the technological background required to understand the building blocks of this dissertation including IoT architecture, Identity and Access Management techniques in IoT, Distributed Ledger Technology, Decentralized Storage, Self-Sovereign Identity and Usage Control. In addition, this chapter also discusses state-of-the-art solutions and related work from the perspective of the challenges addressed in this work. The subsequent four chapters summarize the work published in peer-reviewed publications for this dissertation.

Chapter 3 discusses a decentralized identity management framework for IoT networks in an industrial context. This chapter is primarily based on the preliminary study of designing a decentralized identity model for IoT networks. It presents design, findings and evaluation results for such a framework. This work is presented in Publication I.

Chapter 4 presents a self-sovereign identity-based decentralized digital identity framework for IoT networks. The design, working and data models used in the proposed framework are discussed and presented in this work. The information in this chapter is primarily based on Publication II.

Chapter 5 presents a fair, secure and trusted decentralized IoT data marketplace enabled by blockchain. This chapter builds on the developments of the last two chapters to

use their findings and design a holistic data marketplace to ensure trusted, automated and The information in this chapter primarily describes the findings presented in Publication III.

Chapter 6 presents a user-empowered, privacy-aware, authentication and usage-controlled access protocol for IIoT data marketplace presented in the last chapter. Such a framework ensures real-time access monitoring and instant revocation in case of malicious or unauthorized behaviour. The information in this chapter is based on Publication IV.

Chapter 7 finally concludes the work presented in this doctoral dissertation. An overview of the contributions is first provided. Then, how the contributions have been addressed is described. Finally, the future work that arises from the limitations of the presented work is discussed.

Chapter 2

Technological Foundation

The term Internet-of-Things (IoT) took time to evolve as we know it today. The first *Smart Machine* was discussed very early in around 1982 with a modified Coca-Cola vending machine at Carnegie Mellon University [28] becoming the first Internet-connected appliance, able to report its inventory and whether newly loaded drinks were cold or not. Later, the term IoT was coined by Kevin Ashton and David L. Brock back in 1999 at the MIT Auto-ID Center. The vision to date remains to establish a connected paradigm of intelligent devices that can collect data from isolated, hazardous and hard-to-reach places for humans. Analysing the collected data is used to gather valuable insights to make lives smarter, safer and autonomous. IoT applications are expected to include billions of everyday objects in various domains, namely: smart homes, smart cities, industrial applications, wearable, healthcare, and many others.

2.1 IoT Architecture Overview

The vision of IoT is to extend Internet connectivity to a broad range of physical devices. These devices are embedded with electronics and software that allow intercommunication among them just like connected computers, the main difference being IoT devices are resource constraint and are primarily used for collecting data from the environment they are installed in. IoT as an ecosystem consists of several technologies to enable sensing, data collection, analysis and insight consumption. It is still a very vendor-specific technology as vendors can select the components of each layer to suit the use case. However, most of the researchers and industry specialists break down the architecture into three, four or five layers [29], [30] as shown in Figure 2.1 depending on the required level of abstraction suited for the use case. These architectures each describe the ecosystem from a different perspective, however, a very simplified representation is composed of three main layers:

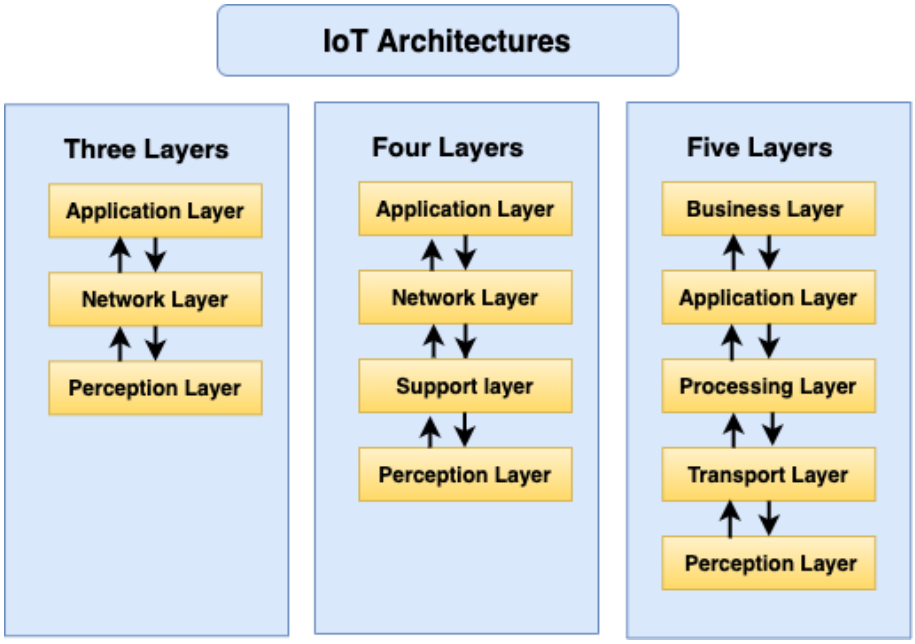


Fig. 2.1 Showing the different layers of 3, 4 and 5 layered architecture of an IoT system

- *Perception Layer:* This layer mainly consists of physical objects such as smart sensors and actuators deployed in hard-to-reach places to gather and respond to raw data such as temperature, oxygen level, soil sensors etc. Therefore, it is the data-gathering layer of the architecture.

- *Network Layer:* This layer is responsible for collecting data from the perception layer and later forwarding to the layer above it for further processing. It consists of components such as network gateways, edge devices, internet routers etc. It is the data acquisition and transport layer of the system.

- *Application Layer:* This layer has the intelligence to process the collected raw data and gather insights from them. It consists of algorithms and applications to carry out resource-intensive tasks which are difficult to process for layers below it.

The above-mentioned three layers comprise of a set of protocols to connect IoT devices with the layers above them. For example, the network layer consists of routers and gateway devices to connect IoT sensors to the cloud and provide an interface for sending data to the application layer. This is achieved using various networking protocols; such as the Constrained Application Protocol (CoAP) [31], Message Queuing Telemetry Transportation (MQTT) [32], and Lightweight Machine to Machine (LwM2M) [33].

The application layer in classic IoT architectures consists of a front-end application hosted by either a proprietary server or a cloud-based server. The front-end application acts as an interface that receives data from sensors, processes the data, performs analytics on it

and later stores it in different forms of storage for effortless retrieval. In addition to these functions, this layer also performs device management functions such as device life-cycle management, authentication, authorization and access control. The device management functions consist of the creation of identities for each IoT device which will be later used for device identification in different layers of the architecture. Device identities are a central component in IoT protocols as they are used in all the layers to enable information routing, resource delivery, authentication, and authorization of access. Building decentralized IoT device management relies on recognizing device identities as a foundation layer, thus the next section discusses the various methods of device identity management and the challenges associated with them.

2.2 Identity Management in IoT

Digital Identity is the unique information used to identify a user, software entity or device online. These unique identities are used to perform authentication and authorization to services and resources present online [34]. Authentication is the process of establishing the truth of the identity provided by the entity i.e. verifying the entity's digital identity [35]. There are three key methods of authenticating an entity:

- something known (password, pincode)
- something assigned (certificate, token)
- something possessed (biometric/digital fingerprint)

While on the other hand, authorization is determining if an entity has permission to access a certain resource after authentication is complete [34], [35].

Identity Management (IdM), is an administrative domain that deals with identifying, assigning and managing identities of individual entities. It define the process of binding digital identity of an entity with its attributes thus allowing authentication followed by access to resources on the network [36]. The definition also includes identity creation and management throughout the device life cycle. IdM services and capabilities also allow users/subscriber entities to control how their identity information is used and disseminated. IdM is a very broad topic that encompasses various applications, functions and capabilities and several different types of entities. As shown in Figure 2.2, it can be described as a layer of entities and their related functions. At the bottom most layer are the various entities in a network. These can be organizations, users, devices and even some virtual objects. The layer above them define the process of identity formation and how identifiers are derived for different entities. There are mainly three types of identity information associated with an entity as shown in Figure 2.2 which can be grouped as follows:

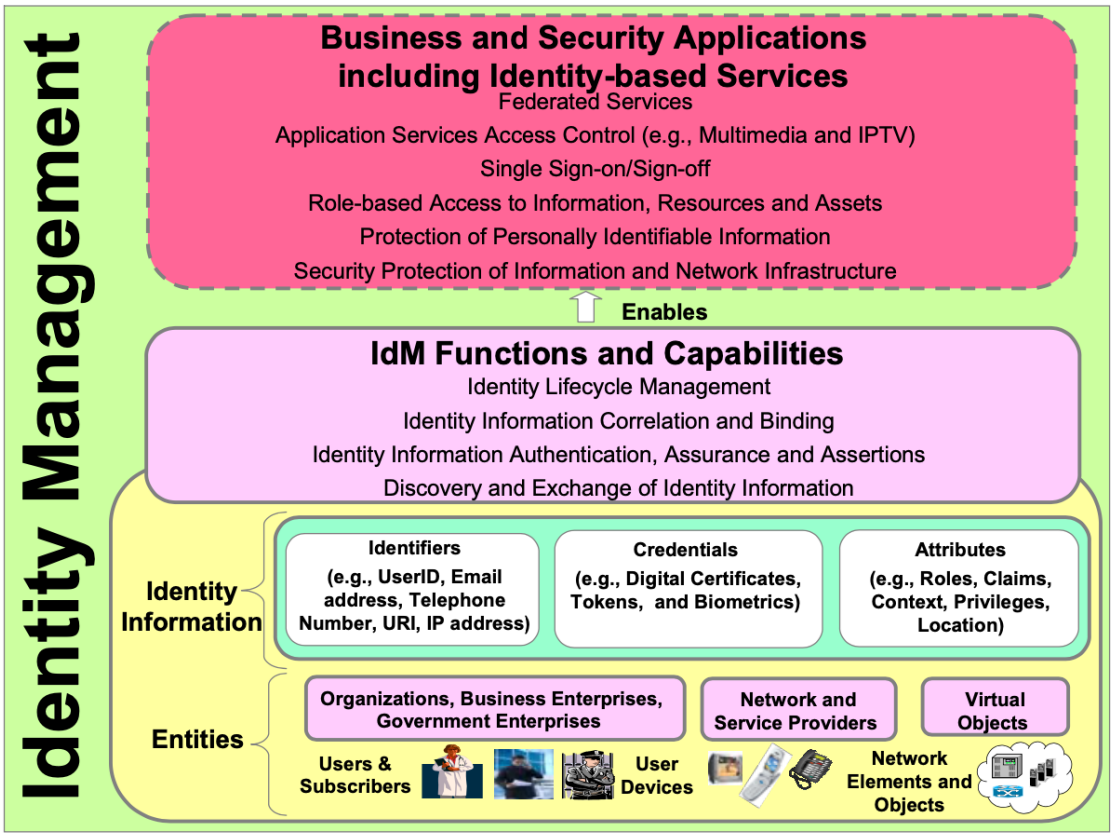


Fig. 2.2 Showing the Identity Management general overview along with various components that form identifiers, attributes and credentials and their functionalities [1]

- identifiers: they are unique combinations of digits, characters and symbols used to identify a subject e.g., UserID, e-mail addresses, telephone numbers. For IoT devices, the commonly used identifiers are Uniform Resource Identifier (URI), IP addresses, RFID Tag ID (TID).
- credentials: they are certain claims associated with the identity of an entity that it receives from an issuing party e.g., for IoT devices, they are digital certificates, and tokens and for users they also include biometrics.
- attributes: they comprise of a set of data that describes the unique characteristics of a subject e.g., roles, claims, privileges, patterns and location and for IoT devices, it can be hardware specification, firmware version, and other physical or logical attributes.

The third layer from bottom is the the IDM functions and capabilities layer. According to ITU-T NGN (Next Generation Networks) a general IdM framework consists of the following functions and capabilities as shown in Figure 2.2:

[1] https://www.itu.int/ITU-T/recommendations/rec.aspx?rec=Y.2720

- **Identity life-cycle management:** It describes the functions that define overall process to create, manage and dispose identities
- **Identity Information Correlation and Binding:** This function defines the process of identity formation by establishing correlation between the identity and attribute of the entity and hence resulting in attribute binding
- **Identity information authentication, assurance and assertions:** These types of functions define the means of establishing the authenticity of identity also known as authentication and performing access control to gauge the assurance level of those entities
- **Discovery and Exchange of Identity Information:** These functions defines the identity registries that enable discovery and protocols that enable exchange of identity information to allow for authentication in a secure manner.

The topmost layer in any IdM system defines the business and security applications including the specific identity-based services like Application level access control and Single Sign On(SSO) etc. These are very application depend functions and will vary widely depending on the application for which IdM is being designed/deployed.

2.2.1 Classification of Identifiers in IoT networks

As seen in Figure 2.2 in the previous section that in the process of identity formation, identifiers play a key role. Identifiers are commonly used to uniquely identify different entities in IoT networks. The most prominent is the *thing identifier* which identifies the "IoT devices" themselves, the entities of interest of an IoT application. Other important entities that are identified include applications and services, users, data, communication endpoints, protocols and locations. Most of the IoT setups today leverage state-of-the-art industrial identification techniques such as digital certificate infrastructure, unique usernames or even IP addresses. Despite that, a common identification scheme is not available for IoT networks, which is attributed to the political and commercial drivers of the standard bodies and their sponsors [37].

The availability of various IoT identification schemes has on the one hand allowed a wide variety of devices to be included on the network irrespective of their hardware specifications while on the other hand they have lead to interoperability issues. Even though there are various IoT identification schemes, these schemes are used in vertical silos among the organizations. There are many different and independent solutions resulting in the need for interoperability. Therefore, new, enhanced, automated and interoperable capabilities are necessary. For this reason, ensuring interoperability remains one of the

primary challenges in IoT. This issue is affecting services and applications that require device identity recognition prior to commencing any exchange of data [38].

The different classes of identifiers used in IoT networks [39] are discussed in more detail below:

Thing Identifier: This type of identifier is used to identify beings (humans, cattle, plants), physical objects (e.g. machines, properties) or digital data (e.g. files, data sets, metadata). These kinds of identifiers are used in predictive maintenance, asset tracking, provenance and quality control. E.g. of such identifiers are the Electronic Product Code (EPC) and the Tag Identifier (TID).

Application and Service Identifier: Application and Service identifiers identify software applications and related services. It also includes identifiers for methods that interact with the application or service i.e. Application Programming Interfaces (APIs) and Remote Procedure Calls (RPCs). An example of such identifier usage is in IoT platforms that provide various services like communication, application store, device management, and device registration. Each service can have a unique identifier.

Communication Identifier: Communication identifiers identify communication endpoints (e.g. source, destination) and sessions. These identifiers include Low Power Wide Area Networks (LPWANs), Ethernet MAC Addresses, IP Addresses, HTTP tokens and Phone numbers. As defined by ETSI GS LTN 002 [40], LPWANs use uniquely assigned communication identifiers to identify end devices in the scope of each network's communication. Similarly, in Ethernet networks (IEEE 802.3 [41]) the Media Access Control (MAC) address is an identifier for communication endpoints at the data link (media access) layer. IPv4 (IETF RFC 791 [42]) and IPv6 addresses (IETF RFC 4291 [43]) are used in IP networks to identify communication endpoints at the network layer. IPv4 uses 32-bit and IPv6 128-bit addresses. Phone numbers are assigned to a specific subscriber station in a phone network. Both global and local unique numbers are used based on the specific application. Another example of such type of identifier is HTTP Session Token which keeps track of all user activities for a particular session in a stateless HTTP protocol. It is usually stored as a cookie.

User Identifier: User identifiers are used to identify users of IoT applications and services. Users can be humans, enterprises (e.g. legal entities) or lightweight software applications that access and interact with server-hosted IoT applications on behalf of the users.

Data Identifier: These identifiers cover both identifications of specific data instances and data types (e.g. metadata, properties, classes). Examples of data identifiers are digital twins, time series data sets and property characteristics (weight, dimensions and temperature). A digital twin is a data set containing the virtual representation of the thing. Note that a thing may have more than one digital twin and that it may contain different sets of information. Sensor data from a thing is provided automatically in (constant) intervals. The data is stored as a time series in the IoT platform for further use.

Location Identifier: This class of identifiers relates to the identification of locations within a geographic area (e.g. geospatial coordinates, postal addresses, room numbers). They are used for applications like supply-chain management and real estate management.

Protocol Identifier: Protocol identifiers inform communication protocols of the upper layer protocol they are transporting data to. These identifiers also include applications about the protocol they have to use in order to establish a specific communication exchange. Examples of usage of these types of identifiers are Ethertype, IPv6 Next Header, URI scheme. These identifiers are used extensively in IoT networks by different independent frameworks to keep these identifiers isolated. For example, an RFID tag that carries an EPC code is not identifiable in a LoraWAN network that is based solely on UUID or vice versa.

2.2.2 Identity and Access Management

Identity and Access Management (IAM) is a broader term for Identity Management (IdM) as it also includes authentication and authorization flow along with the management of identities. IAM addresses the crucial requirement of ensuring appropriate access to resources across increasingly heterogeneous technology environments, and to meet highly rigorous compliance needs. Access to resources is defined through a set of roles according to the entity's credentials and attributes.

Specifically in IIoT, IAM remains a dynamic task due to heterogeneous elements, frequent change of ownership and intermittent connections. For IIoT, the IAM frameworks need to be designed in such a manner that they can account for such factors. IIoT requires a globally accessible IAM framework which is not restricted by geography and the transfer of ownership can be achieved with less impact on its identification and optimally with no loss of track of the historical records.

OAuth 2.0 and OpenID Connect [44], [45] are the two most widely used standard frameworks for authentication and authorization in the industry. However, the main challenge in using OAuth and OpenID Connect for IoT devices is that these standards are

mainly bound to HTTP protocol, which limits their use in the case of IoT devices due to the constraints in power and memory as they do not run on HTTP.

The quest to find a lightweight solution to authentication and authorization solutions for IoT resulted in the Authentication and Authorization for Constrained Environments (ACE) standard. It is one of the few standardized approaches that focus on authorization and authentication for constrained devices to ensure that minimum resource-capable IoT devices can still maintain the required level of security [46].

The unique features of IoT networks such as heterogeneity, sheer number of devices, resource constraint nature, mobility and requirement of interoperability render general standards impractical. These requirements have a greater impact on identity and access management designs from the point of view of privacy and secrecy. Some solutions choose to shift those functionalities to gateway devices [47], [48], which does not satisfy the requirement of ensuring a secure framework. The vulnerabilities associated with gateway devices have been a topic of many studies which indicate that gateways can act as a single point of attack [49]. "UDP hole punching" vulnerability, which is inherited in many IoT gateway and routers using the in-built feature of Universal Plug and Play (UPnP) specification is a well-known vulnerability [50]. The vulnerability allowed an attacker to launch a man-in-the-middle attack on IoT devices by listening to the connection between devices, where the attacker utilized basic plain text-based factory-default password authentication and identification techniques. The attack affected more than 2 million devices connected using p2p gateways [50]. Mirai attack [14] proved this conclusively, where it rapidly spread to millions of devices using nothing more than default credentials for IoT device authentication.

Additionally, the concentration of authorization rules at the gateways results in the usage of coarse-grained access control rules, which can result in relaxed security measures and a widened attack surface. For example, the IBM Watson platform defines two coarse-grained gateway access roles that govern the gateway's ability to register devices on the Watson IoT platform service, standard and privileged [20].

2.3 Evolution of Identity Management Models

Over the years identity management models have also evolved as shown in Figure 2.3. The first model comprised of centralized authorities acting as issuers and authenticators of digital identity. Organizations like ICANN determined validity of domain names and later certificate authorities (CAs) created and stored identities for entities. This led to excessive authority vested in a few corporations with little control left to the users. The second model known as federated identity allowed multiple service providers forming a

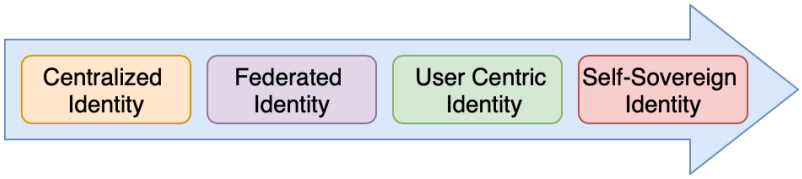


Fig. 2.3 Showing the evolution of Identity Management models in terms of how identities are created, managed and controlled over their lifecycle

federation with one of the identity provider, allowing the user to use same credential across these platforms. This led to great control of technology giants like Google, Facebook etc. There have been multiple cases of identity breaches where identity of a large number of users' was compromised [50], [14], [5].

The third model known as user-centric identity focused on two major elements: user consent and interoperability. User was kept at the center of IDMS, and was free to decide with whom and when to share their identity. Standards like OpenID (2005), OAuth (2010), FIDO (2013), OpenID Connect (2015) and CTAP 2.0 (2018) were introduced for creating user/device authentication. However, these standards use OpenID servers to store passwords/credentials. The most recent model called Self-Sovereign Identity (SSI) gives complete autonomy of its identity to the subject itself. SSI build on ten core principles, namely: existence, control, access, transparency, persistence, portability, interoperability, consent, minimization and protection. These principles are drafted by Christopher Allen, and are influenced by Kim Cameron's laws of identity [51].

2.4 Self Sovereign Identity Model

Self-Sovereign Identity is a new form of digital identity model that gives individuals control over the information they wish to use to prove who they are to websites, services, and applications across the web. It is a more decentralized approach to identity where no single entity controls a majority of user information. The decision about when and what information to share with the verifier lies with the user/identity owner themselves [52].

In this model of digital identity, there are three main actors: identity *Issuer*, *Holder* and *Verifier*. **Issuer** attests and issues credentials (a set of issued attributes) about the **holder** and signs it digitally. When a credential is presented to a **verifier**, it can verify the attributes/claims about the **holder**. For example, lets consider a scenario, where Alice recently graduated from her University and now she wants to apply for a job at a Corporation. In order to verify her educational credentials if she chooses to use SSI model, she can collect her digital educational credentials from university and present them to various potential organizations she want to apply for job as shown in Figure 2.4.

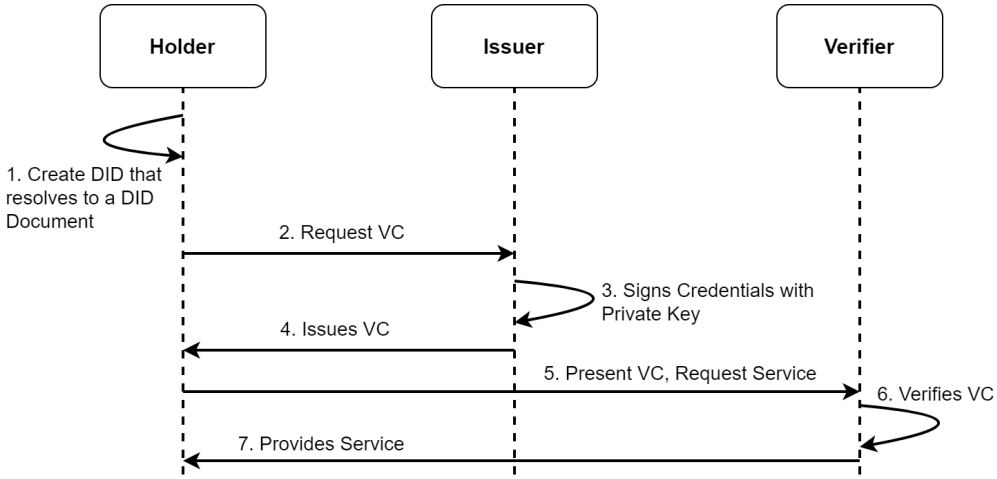


Fig. 2.4 Showing the workflow of interaction between actors to create DID and VC

2.4.1 Decentralized Identifier (DID)

DIDs are a new type of digital identifiers proposed by W3C that identifies a subject (person, thing, abstract entity etc.). No central authority is involved in issuing or maintaining a DID which gives the holder complete control and ownership over their IDs. A holder can create different DIDs for interaction with different parties to ensure unlinkability by correlation of their online activities. Any decentralized network, such as blockchain can be used to resolve a unique key to a unique value. DIDs are underpinned by asymmetric cryptography that uses a public-private keypair. The private keys are used to sign messages by DID owner while public key is available to verify signed messages. The

DID has the following syntax: *"did:example:abcdefgh01234567"*. The *did* prefix, *method name* and *method-specific-id* are separated by colon [53]. DID Methods are a set of specifications by which a particular DID and its associated DID Documents are created, resolved, updated and deactivated using verifiable data registry. A unique identity for that network is defined by *method-specific-id*. Various components of DID ecosystem are shown in Figure 2.5. A full list of did-methods are available for public access and submission [53].

```
{
  "@context": "https://www.w3.org/ns/did/v1",
  "id": "did:example:123456789abcd",
  "authentication": [{     //to authenticate
    "id": "did:example:123456789abcd#keys-1",
    "type": "Ed25519VerificationKey2018",
    "controller": "did:example:123456789abcd",
    "publicKeyBase58": "H3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV"
  }],
```

[2]https://www.w3.org/TR/did-core/

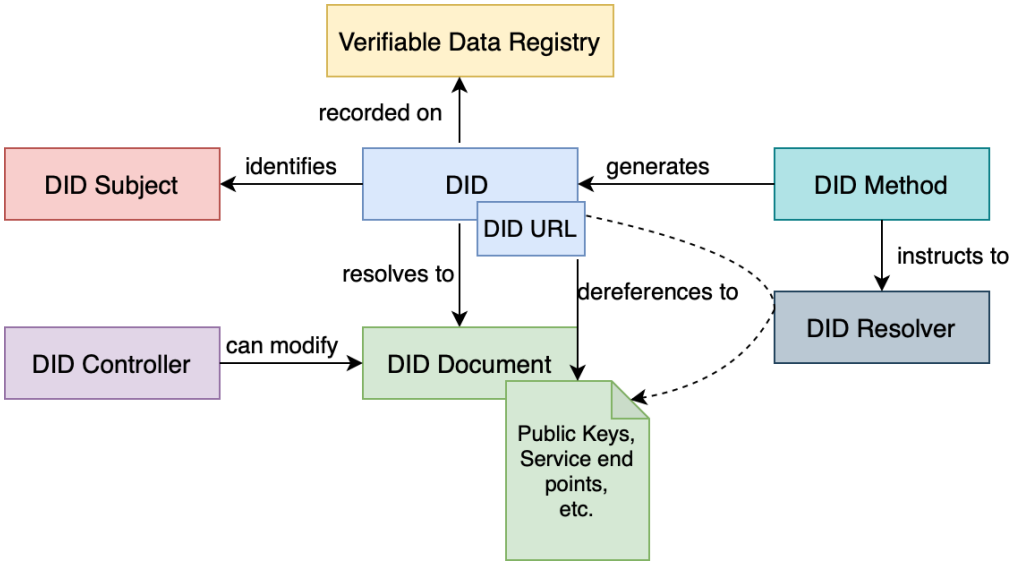


Fig. 2.5 Showing the Overview of DID architecture and the relationship of the basic DID components[Source:W3.org] [2]

```
  "service": [{       //to retrieve VCs
    "id":"did:example:123456789abcdefghi#vcs",
    "type": "VerifiableCredentialService",
    "serviceEndpoint": "https://example.com/vc/"
  }]
}
```

Listing 2.1 DID Document Syntax

A DID resolves into a DID Document using a DID Resolver and this process of mapping a DID to its DID Document shown in Listing 1, is called *resolution*. A DID support four common operations: 'Create', 'Read', 'Update' and 'Deactivate'. The 'Read' operation is the most frequently used operation and is used by the DID Resolver to resolve a DID to its DID Document. DID Doc can contain set of public keys, authentication methods and service-endpoints. The best part about DID Doc is that they do not contain any Personally Identifiable Information (PII) that can compromise the privacy of the subject. The service endpoints provide means of communicating with the subject that owns the DID. DID Docs can be stored directly on blockchain like Bitcoin (*did:btcr*) [54], Sovrin (*did:sovrin*) [55] or they can be stored elsewhere such as relational databases or in a decentralized content-addressable store such as IPFS (*did:uport*) [56].

2.4.2 Verifiable Credentials (VC)

A VC is a set of claims that are machine readable, tamper-proof and cryptographically verifiable using techniques such as digital signatures [57]. The VC can be used to assert the

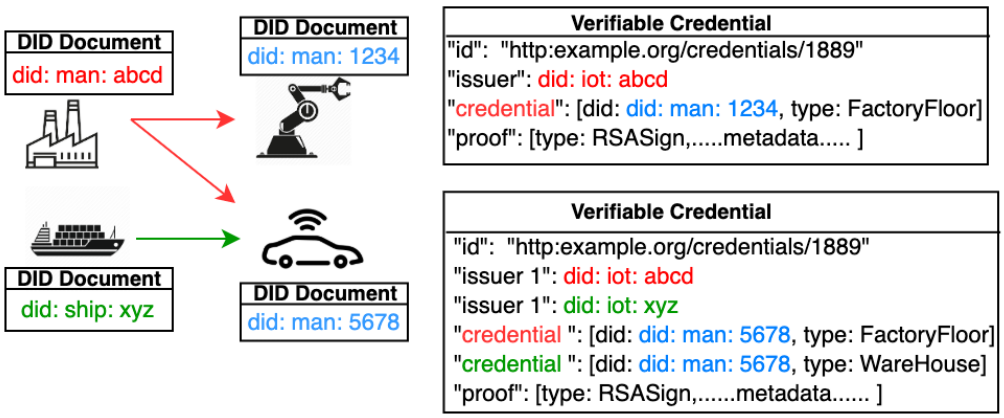


Fig. 2.6 VCs issued to two types of IoT devices (a stationary robotic arm and a mobile autonomous bot) by two separate issuers.

claims of one party to another party. Verifiable credentials are portable, tamper-proof and cryptographically digital claims made about a subject. The claims are issued by a *issuer* after verification of the asserted claims to the *holder*. The claims are requested and verified by *verifier*. For instance, an autonomous car at a charging station can present it's claim that it belongs to the verified owner of the car. The format of a VC is shown in Figure 2.6.

Verifiable Presentation (VP) enables the holder to present claims from one or more VCs. A trusted *Issuer* issues a signed VC to the *holder* and bind it to their DID after verifying their identity. To access a service/resource held by a *Verifier*, the user reveals their DID and the issued VC to the *Verifier*. The *verifier* verifies the claims in the VC presented by the holder using *Issuer's* public key.

VC provides an inter-operable digital format of proving claims that can be stored, cryptographically signed and are under full control of the subject and shared only when necessary. Another major advantage of using VC is privacy-preserving verification of the claims between the subject and verifier. VCs can be stored in several locations, some of the proposed locations being on-chain in smart contracts and off-chain in claim registries.

2.4.3 Limitations and areas of improvement in SSI

Areas of improvement in SSI

Self-Sovereign Identity (SSI) represents a paradigm shift in digital identity management, empowering individuals to own and control their identities. However, despite its potential, SSI has limitations and areas where improvements are needed:

- **Adoption Challenges:** While there are emerging standards, SSI frameworks are still fragmented, with various platforms implementing different protocols. This can lead to interoperability issues and hinder widespread adoption.

- **Entry barrier:** Transitioning to an SSI system often requires significant changes to existing processes and infrastructure, which can be costly and time-consuming for organizations.
- **Data Correlation Risks:** Even though SSI aims to protect user privacy, there is still a risk of data correlation across different services if not implemented carefully. Multiple interactions using the same identity could reveal patterns that compromise privacy.
- **Challenges with Revocation:** Managing the revocation of credentials in a decentralized manner is complex and often lacks a standard approach, which could lead to security risks if credentials are compromised.
- **Key Management:**The security of SSI systems relies heavily on the secure management of cryptographic keys by users. Loss or theft of these keys can lead to permanent loss of access to one's identity, making robust key recovery solutions critical. Implementing secure and user-friendly key recovery solutions, such as social recovery mechanisms or multi-signature wallets, can help mitigate the risks associated with lost or stolen keys.

SSI holds great promise for enhancing digital identity management, but it faces several limitations in terms of adoption, scalability, legal recognition, and security. Addressing these challenges through improved standards, user experience, scalability solutions, privacy protections, regulatory frameworks, and security enhancements will be key to realizing the full potential of SSI in various domains, including IoT.

Relevance of SSI in the context of KYC:

In traditional settings, we often come across the process of KYC(Know Your Customer), which is used to establish the identity of a user who is transacting with an establishment for a service or product. The primary goal of KYC is to prevent fraudulent activities, such as money laundering, identity theft, and financing of terrorism, by ensuring that entities know who their customers are. Key aspects of KYC include (i) identity verification: establishing the identity of the individual wishing to transact with the service provider (ii) due diligence: performing background checks and monitoring customer transactions to assess and mitigate potential risks and (iii) compliance: a regulatory requirement to comply with regulations such as AML.

Self-Sovereign Identity (SSI) and KYC are increasingly interconnected as SSI provides a decentralized approach to identity verification, which can significantly streamline the KYC process. SSI enhances KYC by offering a decentralized, secure, and privacy-preserving approach to identity verification. Through SSI, individuals can manage their own identity data and share verifiable credentials with service providers, simplifying the

KYC process while ensuring regulatory compliance and data protection. This integration of SSI and KYC represents a significant advancement in the way identity verification is conducted, with potential benefits for both users and organizations. SSI can also enable reusable-KYC hence bringing down the cost and time spent by a subject to authenticate with different service providers.

KYC can play a significant role in the digital realm of IoT too. In this dissertation, the concept of KYC is leveraged to convey the significance of secure ID verification on a decentralized platform like a data marketplace too. Just like in the traditional settings, a digital KYC can help in establishing the identity of the user engaging with the platform and over time it can help perform due diligence using the user's activity on the platform. In this work, the emphasis is on performing due diligence while maintaining the privacy of the users and the security of the platform.

2.5 Distributed Ledger Technology Overview

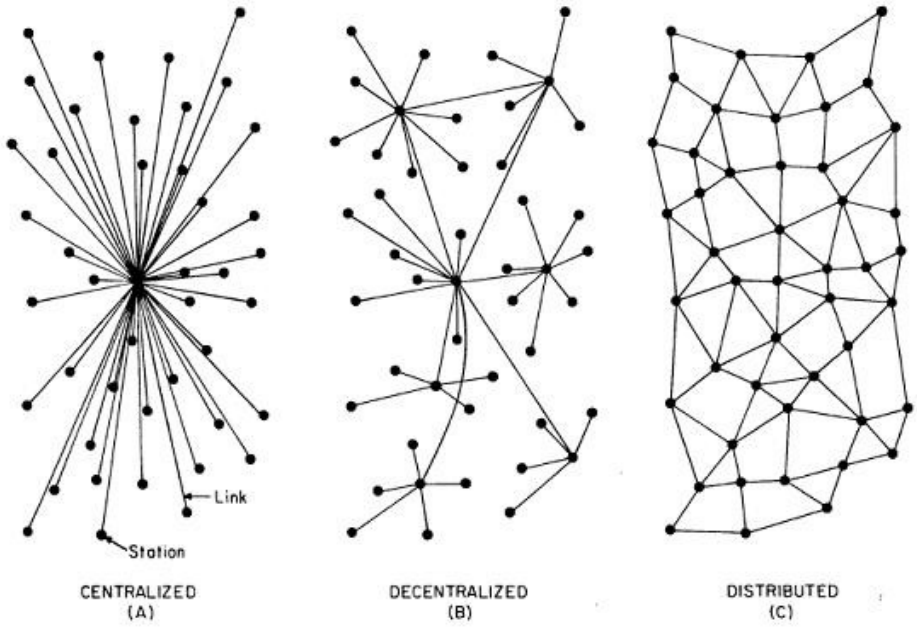


Fig. 2.7 Showing the difference in the models for: Centralized - Decentralized - Distributed types of networks [Source: Medium] [3]

Distributed ledger technology (DLT) is an immutable database that is shared, replicated, and synchronized among the members of a decentralized network. Participants in the network govern and agree by *consensus* on the updates to the records in the ledger. It is argued that such a technology can help to maintain a distributed view of the IoT network access control policies [58]. There a number of implementations for a DLT such as

Blockchain (Bitcoin, Ethereum, Hyperledger Fabric) [59], [60], [61], Directed Acyclic Graph (IOTA) [62], Hashgraph (Hedera) [63]. The most widely used is the blockchain technology. Blockchain technology has gained immense popularity in the past decade in various industries. Blockchain is a decentralized, peer-to-peer technology which consists of cryptographically linked blocks of data storing transactions in the network. The design features of blockchain such as transparency, immutability, non-repudiation and provenance provide a platform to draft a framework that eliminates the requirement of a trusted third party. These stated guarantees are important for authentication and authorization of IoT devices in a network.

Blockchain gained immense popularity due to its decentralized nature. While the earliest application of Blockchain technology was cryptocurrencies, the first one being Bitcoin in 2008, over the past few years, an entire ecosystem of institutions and companies has developed different applications and services around blockchain technology. These applications enable completely new decentralized solutions to some well-known existing issues of centralized network topology in various domains such as health-care, postal service, IoT, supply chain, and others [64]. Figure 2.7 represents the difference in the three major types of architecture topology i.e. centralized, decentralized and distributed. The primary difference is where the major decisions are made and how the information is propagated throughout the control nodes in the system. While in a centralized architecture, there is a single source of decision-making and trust, in a decentralized model the decision-making is distributed among the participants. Distributed means that the processing is shared across multiple nodes, but the decisions may still be centralized and use complete system knowledge.

2.5.1 Blockchain Architecture

Blockchain technology is a decentralized, peer-to-peer technology which consists of cryptographically linked blocks of data storing transactions in the network [65]. These transactions represent the exchange of information among peers and to ensure authenticity, each transaction is added to the blockchain after a rigorous consensus mechanism [58][66]. This ensures a transparent, non-repudiable trail of records which can be verified by all the participants [67]. A blockchain is different from any other ledger in the sense that a transaction is not merely updated/added in the ledger but it is added to a block of transaction only after validation according to a consensus algorithm [68] among the peer nodes. A blockchain consists of the following components:

Nodes

A blockchain consists of several nodes which maintain the integrity of the network and

[3] https://berty.tech/blog/decentralized-distributed-centralized/

perform various functions. Each node maintains an independent copy of the ledger. The nodes together act like a comprehensive database storing all the information in a distributed manner, thereby eliminating the need for a central authority. Depending on the roles, nodes can have different functionalities. *Full Nodes* maintain a full list of all the transactions that ever happened on the blockchain network whereas *light nodes* only store a subset of the transaction and remain connected to full nodes to fetch transactions as and when required. Nodes that verify and validate transactions according to a *consensus mechanism* are referred to as *miners/validators*. These nodes collect the incoming transaction, form a data structure called *block* and validate these transactions in the block to be next appended in the blockchain.

Transactions

Transactions are the fundamental unit as they represent an exchange of assets/information between two parties on the blockchain network. It consists of the sender's address, receiver's address and a value. In most cases, a transaction also includes a fee that is transferred to the miner/validator as an incentive to successfully validate that transaction and include it in a block.

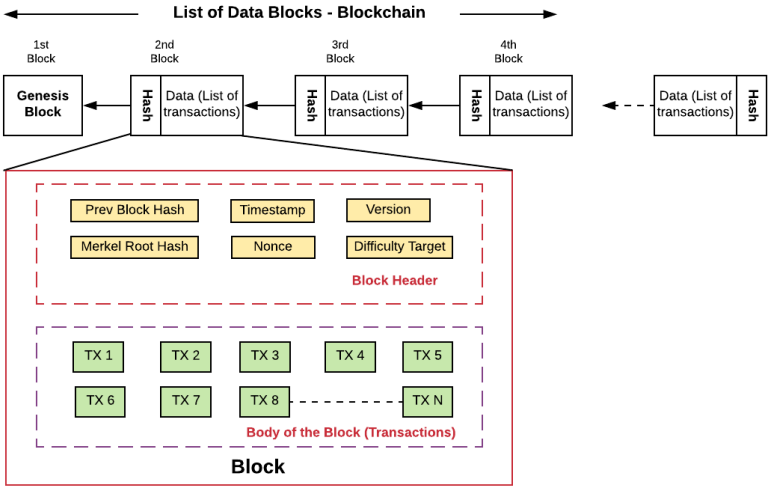


Fig. 2.8 Showing the structure of a blockchain represented by cryptographically linked data blocks and the fields that constitute a block [Source: Data Analytics] [4]

Blocks

A blockchain consists of a series of cryptographically linked blocks of transactions. Blocks are created by miners/validators in the network after verifying the transactions and once verified they are broadcast to all the nodes in the network. When a node receives a new block, it stores it together with other blocks. A genesis block is the first block of the chain of blocks. Figure 2.8 further illustrates the structure of a blockchain. This mechanism of

[4] https://vitalflux.com/blockchain-linked-list-like-data-structure/

linking blocks using the hashing technique provides the important feature of immutability in blockchains since it is computationally challenging to modify a block added to the blockchain. Once a block is added, over time it is joined by other blocks and it becomes increasingly difficult to go back and modify a block. A block generally consists of the following fields: *block number*, *hash* of the previous block, time-stamp, a *nonce* and a transaction representation that comprises a Merkel root and the transactions. The block number and hash together determine the position of that block in the chain. The time stamp denotes when the block was created in the network.

Consensus Mechanism

Consensus mechanisms are a stack of ideas, protocols and incentives that enable a distributed set of nodes to agree on the state of a blockchain. Since blockchain nodes are decentralized with no central authority, the consensus mechanism helps them to agree on the order of blocks in the chain. Blockchain is governed and maintained by its participants making it truly a decentralized technology. Therefore, achieving reliable agreement between its participant nodes is of utmost importance for its proper functioning. Essentially, the consensus mechanism ensures the right ordering of blocks in the chain with agreement from the members of the network, rewards the miners/validators for their work and helps in resolving the forks that may appear in case of disagreement.

One of the key properties of the consensus mechanism is its resiliency and fault tolerance in case of an attack or failure of nodes participating in the consensus. There are two major types of faults that can occur in a distributed network like blockchain. And based on these faults there are two major families of consensus algorithms being used in blockchain projects. The first type of fault is when nodes participating in consensus go down due to failure/crash and this type of fault is easy to recover from. The family of algorithms that solve the problem of nodes crashing simply are called *Crash Fault Tolerant (CFT)* Algorithms. Raft [69] is an example of CFT algorithm. The second type of faults refers to the *Byzantine faults*. The term Byzantine fault takes its name from the *Byzantine Generals' problem* [70]. In this type of fault, the nodes have been compromised by a malicious actor and can provide inconsistent information to other nodes in the network. The algorithms used to solve such faults are known as *Byzantine Fault Tolerant (BFT)* Algorithms. Proof-of-Work (PoW) [61], Proof-of-stake (PoS) [71], Practical Byzantine Fault Tolerance (PBFT) Algorithm [72] are some of the consensus algorithms that belong to the family of BFT type of algorithms. Today, with various types of blockchain implementations being designed, several consensus mechanisms have been proposed. A summary of the chief characteristics of widely-used consensus algorithms has been outlined in Table 2.1. A full discussion of all the different types of consensus mechanism is out of scope of this thesis.

Table 2.1 A comparison of *BFT* based Consensus Algorithms

Characteristics	PoW	PoS	PBFT
BFT	< 51% computing power	< 51% stake	< 1/3 faulty replicas
Energy Saving	No	Partial	Yes
Scalability	High	High	Low
Participation Cost	Yes	Yes	No
Structure	Decentralized	Decentralized	Decentralized

2.5.2 Types of Blockchains

Depending on the ability to join the network and participate in the consensus algorithm, the blockchain networks can be categorized as shown in Table 2.2 and defined as follows:

- *Public Blockchain*: A public blockchain is a non-restrictive, permission-less distributed ledger system. Anyone with access to computing resources and accessibility can download the client package and join the network. Any node in the network can view and participate in the validation of transactions. Such blockchains are suitable for applications like digital assets and related trading services. Public blockchains generally have low transaction throughput and are prone to a wider range of security threats due to the possibility of malicious nodes joining the network. E.g. of such blockchains are Bitcoin, Ethereum etc.

- *Private Blockchain*: A private blockchain is a restrictive blockchain which operates on permission granted only in a closed network. Private blockchains are usually used within an organization or enterprise where only selected members are participants of a blockchain network. These are typically used in private organizations to store sensitive information about the organization. The level of security, authorizations, permissions, accessibility are enabled by the controlling organization. Thus, the throughput of such blockchains is better than the public blockchains. E.g. of such blockchains are Hyperledger Sawtooth, Corda etc.

- *Consortium Blockchain*: A consortium blockchain as the name suggests is managed by a consortium of organizations. Such platforms are mostly suited for government, banks and organizational projects that have more than one stakeholder. More than one organization hosts blockchain nodes and participates in the process of approving transactions in the network. E.g. of such blockchains are Hyperledger Fabric, R3 etc.

Table 2.2 Comparison between different blockchain architectures : Public, Private and Consortium (R: Read, W: Write, C: Consensus)

	Public	Private	Consortium
Access	R: Open W: Open C: Open	R: Open/Permissioned W: Permissioned C: Permissioned	R: Open/Permissioned W: Permissioned C: Permissioned
Security	PoW, PoS other consensus protocols	PBFT, Raft, legal contracts, proof-of-authority	PBFT, Raft, legal contracts, proof-of-authority
Privacy	Medium	High	High
Environment	Untrusted	Trusted	Trusted
Speed	Slow	Fast	Fast
Cost *Architecture*	High Decentralized	Medium Partially decentralized	Low Partially decentralized or centralized
Transaction	Slower	Faster	Fastest
Membership	Anonymous/ pseudonymous	Known Identity	Known Identity
Platforms	Bitcoin, Ethereum, Litecoin	Hyperledger, R3 Corda	Hyperledger, R3 Corda

2.5.3 Blockchain Platforms suitable for IoT environments

As of today, several blockchain platforms cater to various needs of the industry. All blockchain platforms share similar underlying properties such as having a decentralized database that is synced using a consensus mechanism among participating nodes. However, several differentiating factors make each blockchain platform unique. These differentiating features include the capability to host and process smart contracts, incentive mechanism, type of consensus mechanism, ability to control access control, size of block size, having more than one settlement layer etc. It is worth noting that this is not an exhaustive list and they are also not mutually exclusive to each other. It means that the performance of one feature can affect the others. To understand the applicability of a blockchain network one needs to carefully analyse what features the design requires. Some blockchains are also purpose-built for a certain application. Following are some of the general-purpose blockchains that are relevant in an IoT context :

- **IOTA**: IOTA is a lightweight distributed ledger that uses a directed acyclic graph to store transactions on its ledger. This allows transactions to be added in parallel. IOTA's Tangle architecture eliminates transaction fees, making it ideal for IoT applications involving micro-transactions. Theoretically, the Tangle becomes more

efficient as the number of users increases, apparently improving scalability as more devices join the network. However, IOTA is still evolving, and its adoption is lower compared to more established blockchains like Ethereum. The Tangle architecture is less straightforward than traditional blockchains, which might make it harder for developers unfamiliar with the concept. There have been security issues in the past [62], and its unique architecture has not been as extensively tested as more established blockchains.

- **Ethereum**: It is a well-known blockchain platform that is being used for various applications in the industry. Ethereum's robust smart contract functionality allows for automation and can enable complex interactions between IoT devices. Extensive developer support and documentation make it easier to build and deploy IoT solutions. Ethereum is a widely adopted blockchain platform, offering better interoperability with other systems and services. Although the current cost of transactions on Ethereum can be prohibitive, especially for IoT applications involving frequent micro-transactions, Ethereum 2.0 and Layer 2 solutions aim to improve scalability and reduce costs.

- **IoTeX**: IoTeX is touted as purpose-built for IoT environments, offering features like decentralized identity and secure data exchange tailored for IoT use cases. It focuses on privacy and security, incorporating features like ring signatures and lightweight cryptography, which are essential for IoT. However, IoTeX is newer and less tested in large-scale environments compared to more established platforms like Ethereum. Consequently, it has less market adoption and fewer real-world use cases compared to more established platforms. The developer community and ecosystem are smaller, potentially limiting available resources, tools, and integration options.

- **Hedera Hashgraph**: Hedera also uses a directed acyclic graph like IOTA, allowing it to process transactions more quickly and reliably. Hedera's gossip protocol is used to spread information quickly amongst nodes. Hedera can handle a large number of transactions per second, making it suitable for high-frequency IoT environments. While generally lower than Ethereum, Hedera's transaction fees are not fee-less, which can be a consideration for IoT applications with a high volume of micro-transactions. Hedera's governance model is more centralized compared to traditional blockchains, which could be a concern for applications requiring high decentralization. Additionally, the developer and user ecosystem around Hedera is smaller than Ethereum's, potentially limiting available tools and community support.

- **Polkadot**: Polkadot is an innovative blockchain platform designed to enable interoperability between different blockchains, making it a promising candidate for

IoT applications. It offers strong interoperability, scalability, and security features that make it a compelling option for IoT environments, especially those requiring integration across multiple blockchain networks. However, the platform's complexity, resource requirements, and ongoing costs could be challenges, particularly for smaller IoT projects. As Polkadot continues to mature and develop its ecosystem, it could become a more attractive and viable option for IoT applications, provided these factors are carefully managed.

Other purpose-built blockchains for specific use cases such as identity are also being extensively explored in the IoT ecosystem. Two major blockchain projects in this space are Hyperledger Indy and Polygon ID (now known as Privado ID).

Indy is specifically designed for decentralized identity management, making it ideal for IoT environments that require secure, verifiable, and decentralized identities for devices. It supports zero-knowledge proofs, allowing IoT devices to share only the necessary information without revealing full credentials, enhancing privacy. Since Indy adheres to open standards for DIDs and VCs, it is easier to integrate with other systems and platforms within IoT ecosystems. It can be easily integrated with other Hyperledger projects like Hyperledger Fabric and Hyperledger Aries, providing flexibility in building IoT solutions. Additionally, Indy benefits from a strong community and support network, with extensive documentation and enterprise-level backing.

Formerly Polygon ID is also an upcoming popular platform for digital ID. Polygon ID also supports SSI and VCs, allowing IoT devices to manage their identities and share credentials securely. It is compatible with the Ethereum ecosystem, allowing IoT applications to integrate with a wide range of Ethereum-based dApps and services. As a newer technology compared to more established identity solutions, Polygon ID is still evolving, and its long-term security and robustness are less proven. Additionally, deploying and managing decentralized identities on Polygon ID requires an understanding of both Layer 2 solutions and identity management, which may present a learning curve for some.

Choice of blockchain platforms: In this dissertation, three prominent blockchain platforms have been selected to implement PoC in the chapters that follow. Hyperledger Fabric and Ethereum platforms were selected as general-purpose blockchains. These two platforms were selected due to their large developer community, presence of extensive documentation and ease of development. A lot of the blockchain projects discussed above are in their initial phases and hence they often lack the ease of development. A strong developer community helps overcome inevitable roadblocks during development phases. Another reason for selecting these platforms was their ability to host complex smart contracts required for the development of DApps. Since in the proposed dissertation, the capability to embed complex business logic was required Fabric and Ethereum are the best

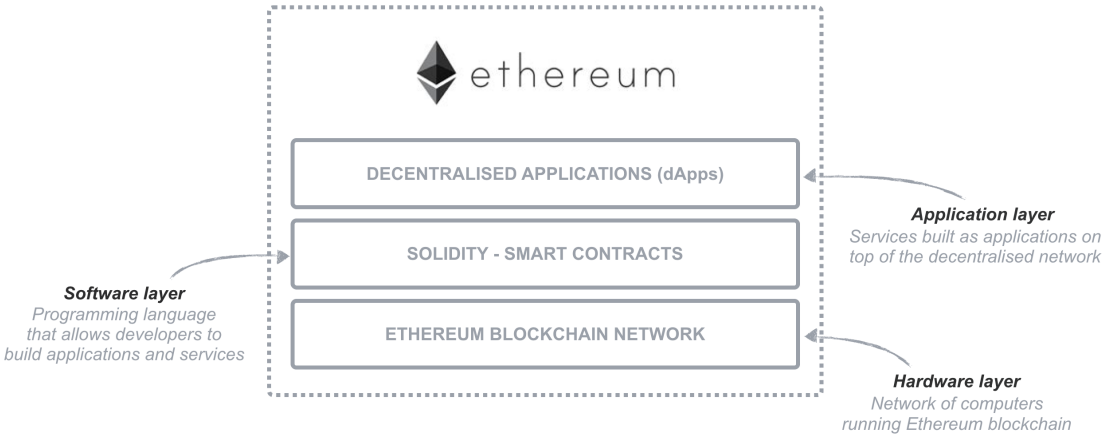


Fig. 2.9 Showing the Ethereum Blockchain Architecture which consists of the base blockchain network, the smart contract layer and the top most decentralized application layer

available platforms for it. The third is the option of adopting each of these blockchain platforms as public or private depending on the use case. Some solutions may require strong enterprise access controls. Therefore, being able to control various levels of access permissions might be a very useful feature.

Similarly, Indy was selected as it is currently a predominantly development-friendly available solution for decentralized identity. It is a mature project with extensive development support and demonstrates various security and privacy features like selective disclosure, separation of concern etc.

2.6 Blockchain Projects used in this dissertation

2.6.1 Ethereum Blockchain

Ethereum [59] is one of the most popular and widely adopted blockchains and the most successful platforms after Bitcoin. They were one of the first platforms to introduce the concept of self-executing machine codes called *Smart Contracts*. A smart contract encapsulates an agreement for transacting in the network and can be deployed by any user. It provides a function-based interface or Application Binary Interface ABI for interacting with the blockchain and performing transactions. Once deployed, the smart contract can be referenced by its address, which is a cryptographic identifier. A user or even an application can interact with the smart contract by sending transactions to the address of the contract and the data payload of the transaction contains the function signature and input parameters. A smart contract can hold and transfer the native coin of the Ethereum blockchain called *Ether*. It can also call functions of other smart contracts. The Ethereum

architecture is shown in Figure 2.9 Ethereum blockchain consists of a "Turing complete" coding system, where the Ethereum Smart Contract can be coded with logic that will be run by the Ethereum Virtual Machine (EVM). The computational capabilities of EVM are prevented from abuse, by charging a fee called *Gas* against each computation. The Smart Contract can be written in supported languages, Solidity, Vyper, Serpent etc[73].

2.6.2 Hyperledger Fabric

Hyperledger Fabric [60] is a permissioned blockchain framework, with a modular architecture. It is a blockchain platform for distributed ledger solutions underpinned by a modular architecture delivering high degrees of confidentiality, resiliency, flexibility, and scalability. It leverages container technology to host smart contract (called Chaincode in Fabric) which contains application logic.

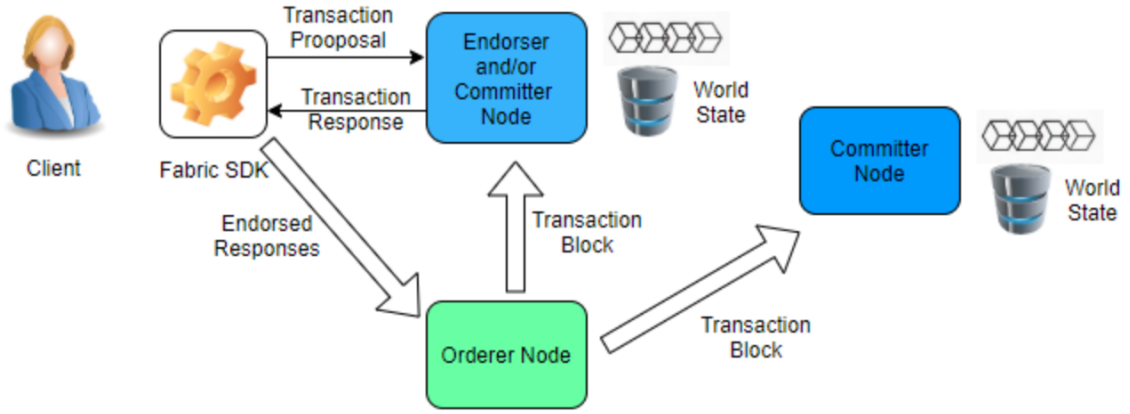


Fig. 2.10 Showing the transaction flow in the Hyperledger Fabric network right from the point it is created by the client to the transaction getting included in the world state

In Fabric, when clients submit the transaction proposal through the Fabric SDK, this proposal is sent to all Endorsing Peers. The endorsing peers check the transaction, verify and execute and generate the Read and Write set as output. Now, this response is again sent to the client. The client collects all responses from all endorsing peers and sends them to Orderer. The Orderer node verifies all transactions and orders them in ascending order to form a block. The verified block is dispatched to all committers which checks the transaction and adds a new block in their own copy of the ledger. The basic transaction flow for fabric is as shown in Figure 2.10. Hyperledger Fabric consists of below major components:

Clients: Clients are applications that act on behalf of a person to propose transactions to the network. These client applications use Fabric SDK to communicate with the blockchain

in order to Read or Write the data in the Fabric blockchain and in-state DB. Each client is issued with a certificate from the CA authority in order to make sure that a valid client has initiated the transaction over the network.

Membership Service Provider(MSP): MSP defines the rules in which, identities are validated, authenticated, and allowed access to a network. The MSP manages user IDs and authenticates clients who want to join the network. This includes providing credentials for these clients to propose transactions. The MSP makes use of a Certificate Authority, which is a pluggable interface that verifies and revokes user certificates upon confirmed identity. The default interface used for the MSP is the Fabric-CA API. However, organizations can implement an External Certificate Authority of their choice. There are two types of MSPs.

- *Local MSP:* It defines users(Clients) and nodes(peers, orderers). It defines who has administrative or participatory rights at that level.
- *Channel MSP:* It defines administrative and participatory rights at the channel level.

Nodes: A "Node" is only a logical function in the sense that multiple nodes of different types can run on the same physical server. There are three types of nodes:

- *Client*: A client submits an actual transaction-invocation to the endorsers, and broadcasts transaction-proposals to ordering service. Thus, clients communicate with both peers and the ordering service
- *Peer*: A node that commits transactions and maintains the state and a copy of the ledger. A peer receives ordered state updates in the form of blocks from the ordering service. Besides, peers can also have a special endorser role that allows them to endorse a transaction before it is committed.
- *Ordering-service-node or Orderer*: a node running the communication service that implements a delivery guarantee, such as atomic or total order broadcast.

Orderers: In a Blockchain network, transactions have to be written to the shared ledger in a consistent order. The order of transactions has to be established to ensure that the updates to the world state are valid when they are committed to the network. Unlike the Bitcoin blockchain, where ordering occurs through the solving of a cryptographic puzzle. Hyperledger Fabric allows the organizations running the network to choose the ordering mechanism that best suits that network. This modularity and flexibility make Hyperledger Fabric incredibly advantageous for enterprise applications. Hyperledger Fabric provides three ordering mechanisms: SOLO, Kafka, and Simplified Byzantine Fault Tolerance (SBFT).

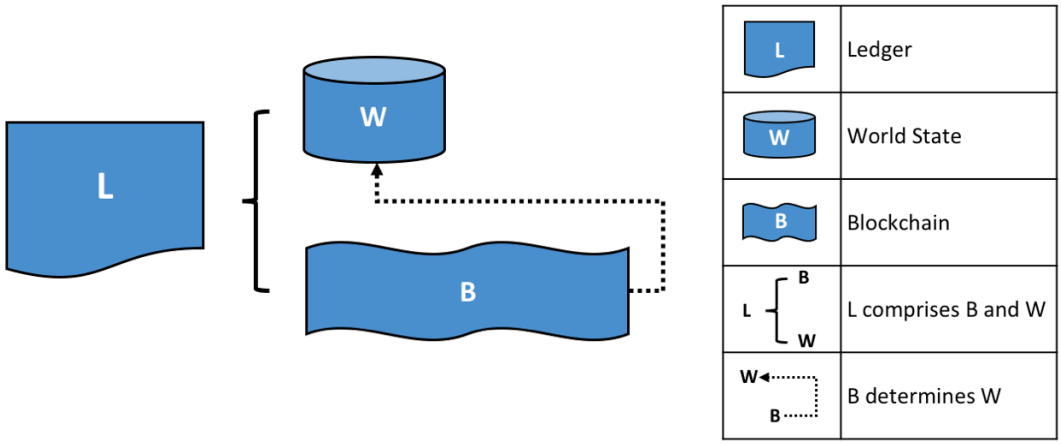


Fig. 2.11 Components of a Ledger in Hyperledger Fabric: A Ledger L comprises blockchain B and world state W, where blockchain B determines world state W

Channels: A fabric network can have multiple channels. Channels allow organizations to utilize the same network while maintaining separation between multiple blockchains. Only members(peers) of the channels are allowed to see the transaction created by any member in a channel. In other words, channels partition the network in order to allow transaction visibility for stakeholders only. Only the members of the channel are involved in consensus, while other members of the network do not see the transactions on the channel. The peer can maintain multiple ledgers. And peer can be connected to multiple channels.

Ledger: A ledger contains the current state of a business as a journal of transactions. A ledger consists of two different parts, a *world state*, and a *blockchain* as shown in Figure 2.11. The world state holds the current value of the attributes of a business object as a unique ledger state. That's useful because programs usually require the current value of an object; it would be cumbersome to traverse the entire blockchain to calculate an object's current value. Whereas a blockchain is a transaction log that records all the changes that have resulted in the current world state. Its data structure is different as once written cannot be removed. It is immutable.

Smart Contract: A smart contract defines the executable logic that generates new facts that are added to the ledger. A smart contract defines the transaction logic that controls the lifecycle of a business object contained in the world state. It is then packaged into the chaincode which is deployed to the blockchain network. Smart contracts are defined within chaincode. Multiple smart contracts can be defined within a single chaincode. When a chaincode is deployed, all smart contracts available within the chaincode is made available to the application. Every chaincode has an endorsement policy attached to it, which applies to every smart contract defined within it. This identifies which organization must sign a transaction generated by Smart contract, in order to consider it valid.

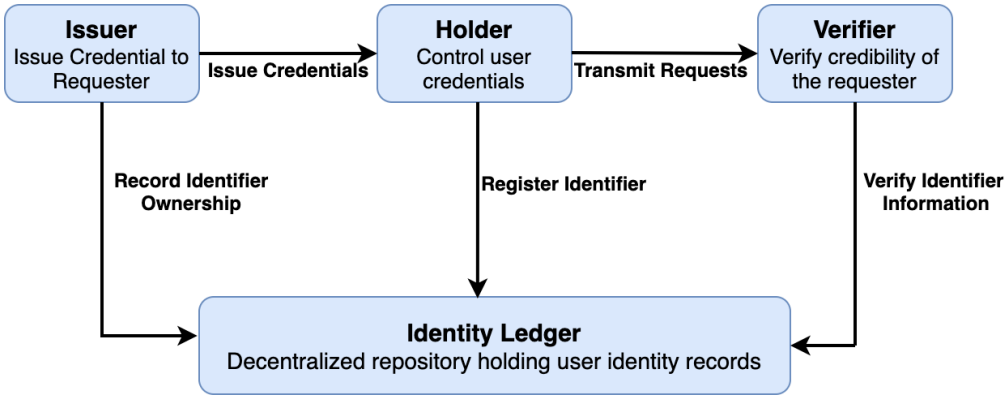


Fig. 2.12 Showing the Hyperledger Indy Architecture

2.6.3 Hyperledger Indy

Hyperledger Indy [74] is one of the projects developed by the Linux Hyperledger Foundation along with a series of blockchain-based projects each for a specific purpose. It is an open-source distributed ledger platform purpose-built for a decentralized, Self-Sovereign Identity system. It is designed to support the creation and management of digital identities that are independently verifiable, and can be used for a variety of use cases such as authentication, authorization, and secure sharing of personal data.

Hyperledger Indy architecture is built upon a distributed ledger platform, which is composed of several key components as shown in Figure 2.12:

Identity Ledger: The ledger is the foundation of the Hyperledger Indy architecture. It acts like a distributed database that stores data of all users on the network, including their public keys, and records of attributes and credentials in a decentralized manner. The ledger is hosted by a set of nodes of the network. Each participating node in the ledger acts like a steward in the network and maintains a copy of the ledger and also participates in the consensus protocol to ensure the integrity of the data.

Identity Wallet: The identity wallet or *agent* is a user-controlled software component that allows the identity holders to interact with the ledger and manage the interaction of their identity information with other parties. With this digital wallet, the identity holders can create, store and manage their verifiable credentials (VCs), private keys and personal data, which can be used to prove their identity to other parties.

Identity Issuer: A software component acting on behalf of an issuer that issues and manages digital credentials, such as degrees, certifications, and licenses. The issuers use their keys to sign the credentials issued by them, and log the event on ledger for future verification purposes.

Identity Prover/Holder: A software component acting on behalf of an identity holder that shares their identity credentials and attributes for verification purposes.

Identity Verifier: A software component acting on behalf of a verifier that verifies the authenticity of digital credentials and attributes presented to them.
Identity Resolver: The identity resolver is a component that helps to map a DID (Decentralized Identifier) to a specific identity holder.
Protocols and Libraries: Hyperledger Indy consists of a set of protocols and libraries that enable interoperability between different identity systems, and support for self-sovereign identity.

2.7 Decentralized Storage

A centralized storage facility such as a proprietary on-premise storage solution or cloud-based storage solution is effectively owned by a single organization/entity. Unlike these a decentralized storage system consists of a peer-to-peer (P2P) network of user-operators who hold a portion of overall data, thereby creating a resilient crash fault-tolerant storage sharing system. Such a decentralized system can be blockchain-based or any P2P-based network [75].

When selecting a decentralized storage system, the following factors should be considered:
Persistence mechanism/Incentive structure: In order to make sure that data persists over time, a persistence mechanism is used. It can be blockchain-based or contract-based. In a blockchain-based persistence mechanism, the whole chain is accounted and each node adds new data at the end of the chain. This leads to chain bloating and is not a sustainable way of designing such systems in the long run. The blockchain must also include some type of incentive structure to make payments to the validators to add data to the chain. Platforms with blockchain-based persistence are Ethereum and Arweave.

In contract-based persistence, the idea is that data cannot be replicated and stored forever by every node and therefore, it should be maintained with contract agreement. In this case, multiple nodes make an agreement to hold a piece of data for a period of time. For this service, they must be refunded or renewed in case they run out to keep the data persisting. For most cases, instead of storing all data on-chain, the hash of location of data is stored. In this way, the complete chain doesn't need to scale to store all the data. Platforms that support contract-based persistence are Filecoin, Storj, 0Chain, Skynet.
Data retention: To prove data retention, the system must have some sort of mechanism to make sure data is retained. One of the most well-known methods is to use a cryptographic challenge that is issued to the nodes to make sure they still hold the data. For example, Arweave's uses proof-of-access to prove data retention. They issue a challenge to the participating nodes to check if they have the data at both the most recent block and a

random block in the past. If the node can't solve the presented challenge, they are penalized. The examples of some known decentralized storage systems with a challenge mechanism are 0Chain, Skynet, Filecoin, Arweave and Crust Network.

Decentrality: Estimating the level of decentralization of such platforms is a difficult problem. There are not many tools available to measure it, but in general, we want to use tools that don't have some form of KYC to provide evidence they are not centralized [75]. Decentralized tools without KYC: 0Chain (implementing a non-KYC edition), Skynet, Arweave, Filecoin, IPFS, Ethereum, Crust Network.

Consensus: Most of these decentralized storage platforms have their own version of a consensus mechanism but generally they are based on either proof-of-work (PoW) or proof-of-stake (PoS). Some of them based on PoW are Skynet and Arweave. And the ones based on PoS are Ethereum, Filecoin, 0Chain, Crust Network.

In this thesis, we use IPFS [76] (InterPlanetary File System) as decentralized storage solution in specific chapters. IPFS represents data as content-addressed blocks, and operates on those data blocks using the following subsystems:

- **Content Identifier (CID)**

  In IPFS, data is divided into blocks, that are assigned a unique identifier called a Content Identifier (CID). The CID is generally computed by combining the hash of the data with it's codec. CIDs are unique to the data from which they were computed, which provides IPFS with the following benefits: Data can be fetched based on it's content, rather than it's location. The CID of the data received can be computed and compared to the CID requested, to verify that the data is what was requested.

- **InterPlanetary Linked Data (IPLD)**

  IPFS uses InterPlanetary Linked Data (IPLD), a type of data model to work with CIDs and content-addressed data. IPFS uses IPLD to represent relationships between content-addressed data, such as file directories and other hierarchical structures, using a Directed Acyclic Graph (DAG) called a Merkle DAG.

- **Content Addressable aRchive (CAR) files**

  IPFS uses Content Addressable aRchive (CAR) files to store and transfer a serialized archive of IPLD content-addressed data. CAR files are similar to TAR files, in that they that are designed for storing collections of content addressed data.

A detailed subsystems and architectural discussion of IPFS is out of scope of this thesis but can be found at [76].

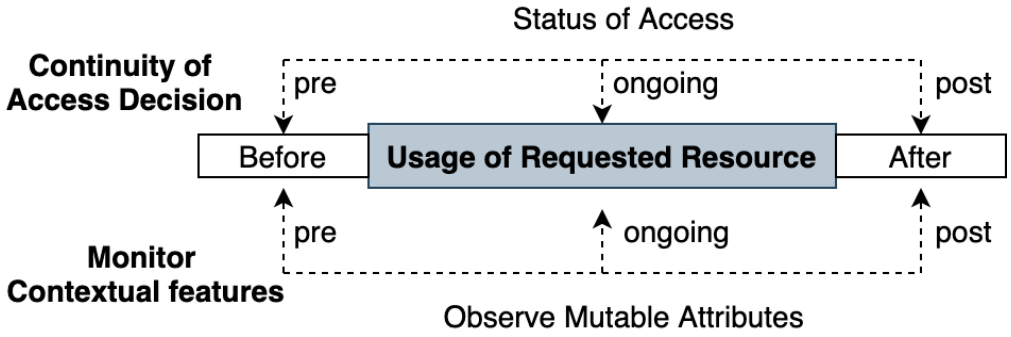


Fig. 2.13 Showing the different stages of UCON system: pre, ongoing and post the usage of the resource requested

2.8 Usage Control

Usage Control (UCON) [21] is a contemporary access control model that extends traditional access control models like Role-Based Access Control (RBAC) and Attribute-Based Access Control (ABAC) to provide more dynamic and fine-grained control over access to resources. UCON focuses on controlling how resources are used once access is granted, rather than just determining who can access what resources. It's designed to address the limitations of traditional access control models in contexts where continuous monitoring and dynamic decision-making are crucial, such as in cloud computing, IoT (Internet of Things), and data sharing environments [13].

Usage-Controlled Access: Usage Control (UCON) (i.e., control the 'intended purpose' or 'expected use' of data) is an extension to traditional forms of access control such as RBAC or ABAC. It is widely used in data control technologies to implement data sovereignty and control data flow between applications [13].

A traditional access control mechanism only authorizes the user once before access is allowed (pre), however, a usage-controlled access system monitors the usage of resources and user attributes on a continuous (ongoing) basis. Such an access control is instrumental for applications that are relatively long-lived and where immediate revocation is required; features required for a real-time IoT data service.

UCON defines three major actors i.e. User (Subject), Resource (Object) and Rights. It defines two types of attributes for both *User* and *Resource*: *mutable* and *immutable* attributes. Attributes are 'features' of an entity that uniquely identifies it. Mutable attributes are user/resource attributes that undergo change as a result of access and immutable attributes do not change during the course of access. UCON defines an access control decision to be an ongoing process and defines three stages of access *pre, ongoing* and *post*. For instance, some mutable attributes change their values because the policy includes

attribute update statements that are executed before (pre-update), during (on-update), or after (post-update) the execution of the access as shown in Figure 2.13.

While a traditional access model only bases its decision-making on *authorization(A)* as a pre-requisite, the UCON decision process defines two more elements to consider when defining access policies *Obligations(O)* and *Conditions(C)* too. Therefore, the UCON decision process is based on three factors: authorizations, conditions and obligations.

Authorizations (A) are the predicates that evaluate the attributes of the subject and object and the requested right to decide whether the subject is allowed to access the object. The evaluation of authorisation predicates can be performed before executing the action (pre-authorisations), or continuously while the action is in progress (ongoing- or on-authorisations) in order to promptly detect policy violations [**?**].

Obligations (O) are decision factors that verify whether a subject has satisfied some mandatory requirements before performing an action (pre-obligations), or whether a subject should continuously satisfy these requirements while performing access (on-obligation). Obligations enforced after access (post-obligations) do not affect the execution of action in any way. For e.g., making a mandatory one-time fee to access the system.

Conditions (C) are system-oriented decision factors and environmental factors; therefore they do not depend on the subjects or objects. The evaluation of conditions involves attributes of the environment such as system load, system temperature etc. It can be executed before (pre-conditions) or during (on-conditions) the execution of the action.

Taking into account the above three factors, a UCON-enabled system provides extended monitoring and control of resources being consumed by a subject in the network.

2.9 Cryptographic Primitives

Asymmetric Cryptography and EdDSA: Asymmetric cryptography is the field of cryptography system that uses a pair of related keys called public-private keypair. The system is used to encrypt or sign data using this keypair to provide security and integrity guarantees that can be verified cryptographically. In this keypair, the secret key is known only to the owner, while others have access to the public key. Elliptic-Curve Cryptography (ECC) is one of the asymmetric cryptography techniques. EdDSA (Edwards-curve Digital Signature Algorithm) [77] is a digital signature scheme based on ECC. In the proposed protocol we use EdDSA, particularly Ed25519 that is based on performance-optimized elliptic curve, Curve25519. Cryptosystems based on ECC have an advantage of using small key and signature sizes. Hence, they have low memory and computational resource requirement.

Due to these properties, they are ideal for resource-constraint devices such as IoT devices. EdDSA signature generation and verification algorithms are shown in Algorithm

Algorithm 1: EdDSA Signature Generation, $Sign_{EdDSA}$ (m,$privKey$) $\rightarrow$ (R,s)
Generate a secret integer $r = hash(hash(privKey) + m) \bmod q$ Calculate $R = r * G$ Calculate $h = hash(R + pubKey + m) \bmod q$ Calculate $s = (r + h * privKey) \bmod q$ *Sign* is (R, s)

Algorithm 2: EdDSA Signature Verification, $Verify_{EdDSA}(m, pubKey, Sign) \rightarrow (Valid, Invalid)$
Calculate $h = hash(R + pubKey + m) \bmod q$ Calculate $P1 = s * G$ Calculate $P2 = R + h * pubKey$ Return $P1 == P2$

1 and 2, respectively. In Algorithm 1, a message m is signed with $privKey$ to get the signature (R, s). In Algorithm 2, the message m, the signature and the public key are the inputs. The signature is accepted or rejected based on its validity.

ECC is considered secure due to the difficulty of Elliptic Curve Discrete Logarithm Problem (ECDLP) and Elliptic Curve Decisional Diffie-hellman Problem.

Elliptic-Curve Discrete Logarithmic Problem (ECDLP): In this problem, one is given an elliptic curve E defined over a finite field $E(F_q)$, a point P of order n on E, and a point Q that is a multiple of P, and one has to find the integer $a \in [0, n-1]$ such that $Q = aP$.

Elliptic-Curve Decisional Diffie-Hellman Problem (ECDDHP): Let E be an elliptic curve defined over F_q, let P be a point of order n at E. The ECDDHP is the problem, given aP, bP and cP, of deciding whether or not c $\equiv ab (\bmod n)$. For Ed25519 the private key is 32 bytes. For Ed25519 the public key is 32 bytes. EdDSA (Edwards-curve Digital Signature Algorithm) [77] is a digital signature scheme based on ECC using a variant of Schnorr signature based on twisted Edwards curves, a type of Elliptic Curve.

The private key ($privKey$) is generated from a random integer, known as seed (which should have similar bit length, like the curve order). The public key ($pubKey$) is a point on the elliptic curve, calculated by the EC point multiplication: $pubKey$ = $privKey$ * G, where G is the generator point for the curve.

Zero-Knowledge Proof(ZKP): The ZKP is a cryptographic way of presenting a knowledge that a *Prover* wants to prove to the *Verifier* without revealing the knowledge itself. The Zero-Knowledge Succinct Non-Interactive Argument of Knowledge(zk-SNARK) is a ZKP construct using which a Prover can prove knowledge of information to the Verifier without revealing the information and without any interaction between them. The zk-SNARK system presented in [78] is the most efficient and widely accepted, therefore, we use this in

Table 2.3 ZKP Algorithms

Algorithm	Description
$(crs,t) \leftarrow Setup(R)$	For relation R, crs and t are generated.
$\pi \leftarrow Prove(R,crs,st,w))$	This algorithm takes crs and $(st,w) \in R$ as input and return π.
$0/1 \leftarrow Verify(R,crs,st,\pi)$	crs, st and π are the inputs for this algorithm and 0(reject) or 1(accept), is the output.
$\pi \leftarrow Sim(R,t,st)$	The simulator takes t and st as inputs and returns π.

proposed protocol. A common reference string is shared between the Prover and Verifier to achieve ZKP [78]. In the algorithms presented in Table 2.3, a relation generator $\mathbb{R}$ returns a binary relation R for a security parameter λ. For pairs$(st,w) \in R$, st is the 'statement' and w is the 'witness'. crs is the common reference string and t indicates the simulation trapdoor. An efficient prover publicly verifiable non-interactive argument for $\mathbb{R}$ is a quadruple of probabilistic polynomial algorithms (Setup, Prove, Verify, Sim) as shown in Table 2.3 such that:

- $(crs,t) \leftarrow Setup(R)$: The setup produces a common reference string crs and a simulation trapdoor t for the relation R.
- $\pi \leftarrow Prove(R,crs,st,w)$: The prover algorithm takes as input a common reference string crs and $(st,w) \in R$ and returns an argument π.
- $0/1 \leftarrow Verify(R,crs,st,\pi)$: The Verification algorithm takes as input a common reference string crs, a statement st and an argument π and returns 0 (reject) or 1 (accept).
- $\pi \leftarrow Sim(R,t,st)$: The simulator takes as input a simulation trapdoor and statement st and returns an argument π.

There are three properties of ZKP [78]: completeness, zero-knowledge and soundness .
Completeness: An honest prover can convince a true statement to an honest verifier:

$$Pr[(crs,t) \leftarrow Setup(R); \pi \leftarrow Prove(R,crs,st,w) :$$

$$Verify(R,crs,st.\pi) = 1] = 1.$$

Zero-Knowledge: The proof does not reveal anything other than the truthfulness of the statement. For all $\lambda \in N$, $(R,z) \leftarrow R(1^{\lambda}), (st,w) \in R$ and adversary A, we can write:

$$Pr[(crs,t) \leftarrow Setup(R); \pi \leftarrow Prove(R,crs,st,w) :$$

$$A(R,z,crs,t\pi) = 1]$$

$$= Pr[(crs,t) \leftarrow Setup(R); \pi \leftarrow Sim(R,t,st)]$$

$$: A(R,z,crs,t,\pi) = 1.$$

Soundness: A prover cannot prove a false statement to the verifier:

$$Pr[(R,z) \leftarrow R(1^{\lambda}); (crs,t) \leftarrow Setup(R);$$

$$(st,\pi) \leftarrow A(R,z,crs) :$$

$$st \notin L_R \wedge Verify(R,crs,st,\pi) = 1] \approx 0.$$

Dynamic Cryptography Accumulator: An accumulator is a short binding commitment to a set of elements and allows for a short (non)membership proof for any element in the set or not in the set. These proofs, also called witnesses (witness to element being accumulated in the accumulator), can be verified against the commitment [79]. A dynamic accumulator that allows addition/removal of values dynamically is proposed in [80]. The proposed protocol implements a revocation policy using dynamic universal accumulator to remove dishonest or malicious *buyers* from the marketplace platform. A *buyer* needs to provide a non-membership witness that they are not in the accumulator before requesting data stream purchase. The properties of such an accumulator are as follows:

Generating an Accumulator: A secret key k_{Acc} is generated. The accumulator generation function $G_{Acc}()$ takes k_{Acc} and the revocation list L as inputs ad returns an accumulator U. The function is $U \leftarrow G_{Acc}(k_{Acc}, L)$

Updating an Accumulator: To add a new value v_{new} to the accumulator, function Upd_{Acc} takes current accumulator U, k_{Acc} and v_{new} to output updated accumulator $U_{new} \leftarrow Upd_{Acc}(U, k_{Acc}, v_{new})$

Generation of a Non-membership Witness: The non-membership witness generation function G_w is implemented as $w \leftarrow G_w(U, k_{Acc}, L, a)$ where a is a value not in L.

Verification of a Non-membership Witness: The non-membership witness verification function $V_w()$ is implemented as $0/1 \leftarrow V_w(U, w, a)$. It returns 0 if a is in L, else 1.

2.10 Summary

The primary goal of this chapter is to introduce the essential background knowledge necessary to understand the research areas investigated and problems formulated for this dissertation. This chapter begins with a brief description of IoT network fundamentals and introduces the challenges of managing the life-cycle of a large number of devices while ensuring security and privacy. The IoT architectural layers with IAM frameworks and classification of identifiers in IoT networks are discussed. The IAM concepts in IoT are highlighted with elaboration on the commonly used identification schemes. Thereafter, the common forms of identification used in the traditional IoT ecosystem are discussed. The chapter later discusses the leading technologies and frameworks proposed in the domain of decentralization, including blockchain, smart contracts, decentralized identifiers, continuous authentication, Zero Trust framework and Usage Control Policy.

The context discussed in this chapter is relevant to Publications I - IV and therefore form the foundation of the work carried out in this dissertation.

Chapter 3

Decentralized Identity Management Framework for Industry 4.0

This chapter is the first of four chapters describing the core research contribution of this dissertation. The key information in this chapter is based on work published in Publication I. To this end, it presents a decentralized identity management framework for Industrial IoT for managing identity and access management policies. The objective of this chapter is to propose a novel smart-contract-based decentralized identity management framework that can improve the handling/communication of IoT devices.

Industry 4.0 heavily uses the connected machine paradigm to automate, track and maintain processes. However, a major challenge in Industrial Internet-of-Things (IIoT) remains managing the burgeoning number of sensors. The centralized framework for managing these devices raises concerns. In this work, a novel proof-of-concept framework for managing identity and access management policies for IIoT is proposed. The proposed framework is a smart contract-enabled and blockchain-based decentralized life-cycle and access management system. It is backed by decentralized storage technology IPFS. The proposed framework is decentralized and scalable unlike the state-of-the-art IIoT management frameworks designed using legacy cloud infrastructure which are by default centralized in nature.

3.1 Introduction

With increased automation in industrial processes, machine-to-machine communication is becoming pervasive. Since IoT provides a connected paradigm of intelligent devices that can sense the manufacturing processes closely, it is being widely exploited in industries like manufacturing, supply chain management and predictive maintenance [81]. This wide use of IoT devices brings with it a number of management challenges. Whilst there are

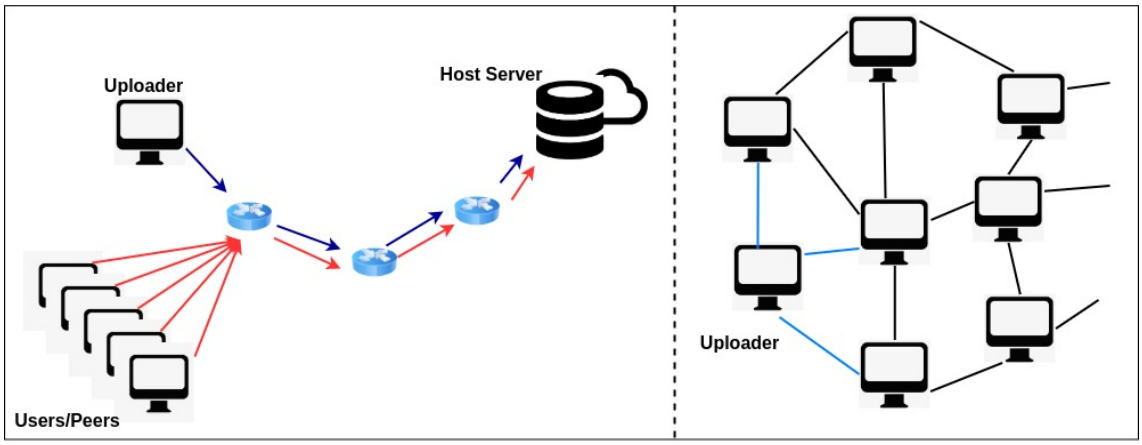


Fig. 3.1 Showing the architectural difference between Client-Server network model and Decentralized network model

several traditional identity management(IdM) solutions available in the market, they are not suited for an Industrial Internet-of-Things (IIoT) setup. Most of the solutions are heavily centralized around cloud architecture which creates a bottleneck in real-time data access and enables access control policies for smart devices [82]. When identities are stored in a centralized location they become a potential attack vector, the confidential information is prone to data breach and manipulation. A single breach not only has digital fatality but can have serious repercussions in a physical environment [83].

Industrial facilities are dynamic environments as compared to home or office workspaces. They have several mobile entities and access control of these devices is shared among various management parties, unlike the home environment which has a single controller. Sensors installed in such industrial facilities gather highly sensitive data and this initiates the need for secure management practices of such devices. In a conventional IIoT environment, the security of such sensitive data is at high risk due to the possibility of a single point of failure. To overcome this problem there is a need to develop a framework based around a decentralized architecture [84].

The blockchain technology has gained immense popularity in recent times. It is a decentralized, peer-to-peer technology which consists of cryptographically linked blocks of data [65]. These blocks of data represents transactions or exchange of information among peers and in order to ensure authenticity, each transaction is added to the blockchain after a rigorous consensus mechanism thus making it a suitable decentralized solution [58][66]. Decentralized systems promote trust and integrity in a system where many key coordinators are managing a set of resources. Figure 3.1 illustrates the difference in data/command propagation in both models. In a client-server model, the flow of information is always towards or away from a central server which stores all the information in the system. Whereas, in a decentralized system, each device in the network is a node hosting and

consuming data simultaneously. Even if a number of nodes fail, the network keeps functioning, unlike a centralized system where server failure means the network comes to a halt.

Although blockchain as a distributed ledger is capable of storing a certain number of transactions, its efficiency impedes as the volume of data stored on the ledger increases. Increased data storage on the blockchain increases the synchronization time among the peers and negatively affects scalability. Currently, the chain is growing at a rate of 1MB per block every 10 min in Bitcoin [85]. In the Ethereum size of the block is measured in two ways: gas usage and block size [86]. Currently as it stands after PoS update it stands at 15 million gases (roughly 1.7 MB) and it is generated every 12 sec. Similarly, in Algorand the block size is 5MB [87]. Therefore, it is not viable technically to store a large amount of data on the blockchain ledger. This becomes an even bigger issue when dealing with IIoT devices which tend to produce large volumes of data in a very short amount of time. The amount of data increases exponentially and the sensitivity of the data dictates that this information needs to be disseminated at the earliest. To overcome this issue, our proposed architecture is built using blockchain in conjunction with IPFS. IPFS is a content-addressed peer-to-peer distributed file system [88]. A file stored in IPFS has a unique hash that is derived from its content. This makes the file secure from tampering by an unknown entity. IPFS eliminates the need for a centralized storage entity in our architecture.

3.2 Blockchain-Based Identity and Access Management Systems: A look at state-of-the-art

Researchers across the world are exploring new technologies to make the IoT scalable, secure and autonomous. The goal is to develop techniques so that the devices can communicate securely with minimum human intervention. A secure IdM framework is fundamental to achieving device-to-device communication for authentication and access management purposes with the least human intervention. Hurrow et al. [89] describe an identity management technique for cloud-based IoT. In this work, the *identity manager* is a component of cloud infrastructure that creates and stores digital identities associated with each device in a cloud database, retrieving it during the device authentication process. Identity creation for new devices and removing existing devices is also managed via *identity manager*.

The state-of-the-art technology for authentication and access management primarily uses cloud-based IdM infrastructures [90]. The major processing tasks and storage are delegated to the cloud network. In these frameworks, there are two main methods of creating identities for IoT devices, first by giving a username-password for devices and by

using PKI (Public Key Infrastructure) to create digital certificates corresponding to each device. In both methods, device identities are stored in a central server and the task of authentication is entrusted to a centralized entity. Often the authentication information is stored in plain text in databases. This led to a large-scale attack launched using the collective power of IoT devices acting as Mirai botnet that infected up to 600k devices to take down many online services [5]. The reason for the attack was simple, using default username passwords, attackers formed a large army of IoT devices to launch one of the largest DDoS attacks. Hence, public-key cryptography can be useful to create identities and bind these identities to the physical attributes of devices.

Some work related to distributed identity and access management of IoT devices has been proposed by researchers. Novo [91] describes a generalized framework for access management in IoT using smart contracts as an enabler for defining access policies. This work mainly describes managing access policies for IoT devices by an intermediary interface. However, no identity creation and management is described in this work. Also, resource access policies are stored on the blockchain platform which can impact the performance of the framework. Omar et al. [92] focuses on identity management in a general scenario specifically ownership management and transfer of IoT devices. They present a framework for identity creation and ownership transfer for IoT devices using blockchain. This work relies on a global registry for storing identities created for IoT devices. Hammi et al. [93] propose a scheme for decentralized authentication in IoT networks. In this work, secure virtual zones containing a few IoT devices are proposed and the devices in these zones trust each other. The inter-zone communication requires authentication which is achieved using smart contracts. However, these trust bubbles create a siloed framework for sensor communication, in which authentication requests are directed to one manager node in the bubble.

Using smart contracts, Papadodimas et al. [94] demonstrate an implementation of DApp for providing data monetization. The devices can buy and sell data by leveraging the smart contract functionality. Li et al. [95] introduce the idea of using multiple satellite chains that form interconnected but independent sub-chains of a single blockchain network. These chains interact with each other with the help of a specific validator node. Each satellite chain implements a business logic. Mell et al. [96] propose a federated authentication of the user to a relying party without a third-party authenticated service. Authentication is achieved between the relying party and the user through a smart contract logic.

The above-discussed works present approaches taken to decentralize the IoT infrastructure using the upcoming blockchain technology. However, most of them rely on blockchain ledger for data storage which can impede the scalability of the system with an increased volume of IoT devices in the network. Additionally, while the above-discussed work offers identity management solutions and some showcase access management too, the procedure

of device authentication remains unexplored in most cases. As discussed, achieving reliable device authentication is an important feature for achieving secure device-to-device communication.

3.3 Case Study: Factory Floor Automation

To understand the functionality of the proposed architecture for identity and access management, a use case from the manufacturing sector is demonstrated. Boeing, a leading aviation equipment manufacturing company is using IoT technology on factory floors to optimize production operations by improving predictive maintenance of the manufacturing sites [97]. The efficiency of such industrial organizations depends on managing downtime and shipment delays. And IoT plays a key role in monitoring the processes closely. However, the company today relies on third-party cloud services to store operational and sensor data [98]. While using cloud infrastructure may seem an efficient tool in these scenarios, the problems associated cannot be ignored. The facility's operational data and sensor data are confidential and must be protected against attacks to maintain accountability of critical components in the aviation industry. The identity and access control in such third-party cloud services are usually managed using digital certificates which again rely on another provider. These certificates are not interoperable and susceptible to attacks as they are a single point of failure.

There are several reasons for using the IPFS-backed blockchain solution for designing IdM in the IIoT scenario. Firstly, there is a need for a decentralized IdM architecture given the large number of sensors. The IoT devices try to connect simultaneously to the server for access control permissions, sending multiple requests at a given point in time to the central server. These servers are often physically hosted in far-off geographical locations, affecting the response rate, hence resulting in potential degradation of performance. Secondly, security risk and trust management issues are reduced by storing identity-related information and access control policies in a decentralized framework instead of using a centralized architecture or distributed one where the owner still remains a single entity. Therefore, in the proposed work, a framework for decentralized identity and access control management is proposed.

3.3.1 Contributions

In this work, a novel architecture based on decentralized technologies namely, blockchain and IPFS is proposed. The proposed architecture uses blockchain technology to manage identities and access control to these smart devices in the network. The main contributions of this work are:

- A novel proof-of-concept decentralized IdM framework is proposed which is based on a smart contract-enabled blockchain architecture that is supported by distributed storage, IPFS.
- It is observed that the framework's performance is at par with current state-of-art architecture and yet provide identity management and security in an industrial setting. It completely forgoes centralization and associated risks.
- It is demonstrated that the proposed decentralized framework can achieve a stable throughput and can be easily scaled up in an enterprise setting which demands security and scalability.

3.4 Proposed Architecture Overview

In the proposed work, blockchain acts as a decentralized identity management authority that maintains the mapping of identities to respective public keys. The smart contract provides the necessary logic to help in the provisioning and revocation of keys of the IoT devices. The majority of policies and relevant metadata is stored on the IPFS while the blockchain only stores corresponding file hash as immutable transactions in the ledger. Every identity creation, deletion and policy update is logged on the blockchain as a transaction. Access control policies defined in the network are retrieved from the local copy of the ledger. IoT devices are inherently resource-constrained and hence they cannot act as a blockchain node. Therefore, in our architecture, the *Wireless Sensor Network (WSN)* formed by sensors is not included in the blockchain network and interacts with the latter through a *protocol translator gateway (PTG)*.

3.4.1 System architecture

Figure 3.2 shows the components of the proposed framework. The framework consists of the following entities:

1. *Blockchain Network*: The blockchain network in the proposed architecture is a consortium blockchain as it closely mimics the real-world industrial scenario. In this network, the participating parties agree on a set of logical business operations, recorded in the smart contract. Hence, it allows fair participation of the parties as compared to a private blockchain. It also promotes traceability. Furthermore, it has better transaction throughput and low setup costs when compared to a public blockchain. As a result, it offers advantages of both types of blockchain and still achieves decentralization. Each node stores the ledger and accesses global IPFS storage to access the identity and access-related policies for IoT devices.

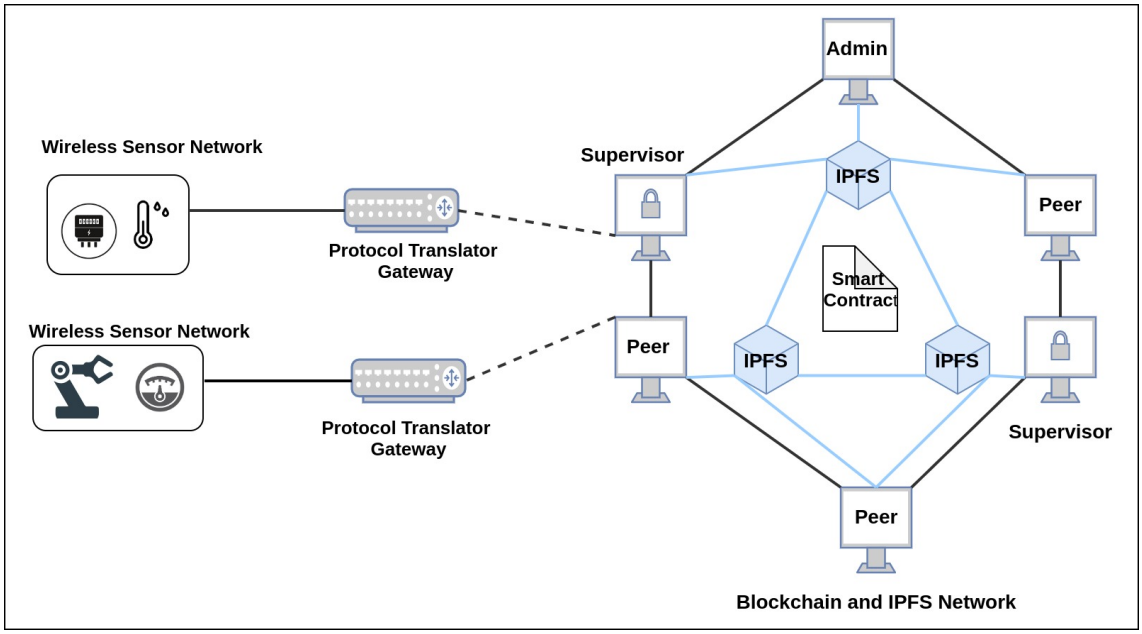


Fig. 3.2 Decentralized framework for Industrial IoT. Each industrial facility hosts a blockchain ledger and IPFS file system as a local copy.

2. *Administrator Node*: The administrator node is a special node that initiates the blockchain network and invites other nodes to be a part of the network. This node deploys the smart contract that defines the functionalities of the entire system. Since this node deploys the smart contract in the blockchain, it receives the address of this contract on the network. Later, this node shares the address with supervisor nodes in the network which allows them to interact with the smart contract.

   It must be noted that as explained above, the framework establishes a consortium blockchain network in which the identities of nodes are known (unlike a public blockchain where the nodes are anonymous and the real identity behind an account/node operator is unknown) as it usually consists of known entities/organizations coming together to jointly host a blockchain network. Therefore, an Admin node in such a setting is an identified blockchain node from either member of that consortium. Depending on the governance structure of the consortium network, a node from any member group can become an *Admin* node and can be tasked to carry out the network and smart contracts setup after a business/legal level agreement is achieved between the consortium members. It does not necessarily mean that such a network becomes centralized but rather this is how the governance structure of this consortium network might be defined. The consensus in such a blockchain network is still achieved in a decentralized manner.

Table 3.1 A table summarizing all the symbols used in the algorithm for this chapter

Symbol	Explanation
K^S_{pub}	Public Key of the Supervisor
K^S_{pri}	Private Key of the Supervisor
K^D_{pub}	Public Key of the Device
K^D_{pri}	Private Key of the Device
τ_S	Supervisor ID
τ_D	Device ID
$Sig_{K^S_{pri}}(\tau_S)$	Digitally Signed Supervisor ID
$Sig_{K^D_{pri}}(\tau_D)$	Digitally Signed Device ID
$\Im$	Ticket with device metadata stored in IPFS

3. *Smart Contract*: The smart contract contains the core logic of the architecture as its functions. A set of operations that allows entities in the network to provision identities to IoT devices, manage their access control and later revoke identities are defined in the smart contract. These operations can be triggered by the smart contract transactions sent by participating nodes in the network. A security feature is introduced in the contract function that authorizes only the registered supervisor nodes to send transactions to those functions. This feature restricts unauthorized access request load on the smart contract.

4. *Supervisor Nodes*: Any trusted peer node in the network can register itself as a supervisor node. Once these nodes are registered as supervisor nodes with the smart contract, they can now register IoT devices in the blockchain platform. Supervisor nodes have higher computational and storage power and hence they can choose to host a copy of the blockchain. However, if these nodes are some lightweight node in the system they can choose otherwise. Each IoT device should register itself with one registered supervisor node. It can also be registered under multiple supervisor nodes. That will ensure robustness in case of any faulty supervisor node.

5. *Wireless Sensor Network (WSN)*: The IoT sensors installed in the facility form a WSN that interacts with the blockchain network through the gateway in the architecture. These sensors have limitations in terms of memory, power and computational capabilities. Each IoT device in the network is uniquely identified by its *SensorID*.

6. *Protocol Translator Gateway*: Since IoT sensors are resource-constrained devices they use low-power communication protocols like MQTT (Message Queuing Telemetry Transport) and CoAP (Constrained Application Protocol) within *WSN*. However,

blockchain network uses RPC protocol among their peers. In order to provide interoperability between the blockchain network and IoT sensor network *PTGs* are used. They convert incoming MQTT requests from sensors to JSON-RPC message format before sending it to a supervisor node and vice-versa.

7. *IPFS File System*: IPFS file system is a content-addressable peer-to-peer file system. On uploading a file to IPFS, a hash of the content of the file is returned. Using IPFS in our architecture provides two benefits, first, it removes the need for centralized storage and second the content of the file is cryptographically secured using the hashing mechanism. Any tampering with a file stored on IPFS can be detected and traced back.

3.4.2 Smart Contract Methods

The methods/programming functions defined in the smart contract are as follows. The symbols used in the explanation are defined in Table 3.2.

- *RegisterSupervisor*(K^S_{pub}, τ_S): to register any node S with Supervisor_ID τ_S as supervisor.
- *RegisterDevice*(K^D_{pub}, τ_D, K^S_{pub}): to register a device D with help of a supervisor S .
- *AuthenticateSupervisor*(K^S_{pub}, $Sig_{K^S_{pri}}(\tau_S)$): before a supervisor S can register/deregister a device, it needs to authenticate itself.
- *AuthenticateDevice*(K^D_{pub}, $Sig_{K^D_{pri}}(\tau_D)$): before a device can request/access a resource or even give a command to other IoT device, it needs to authenticate itself.
- *AddResourceControl*(*ResourceList*): to add access control for a resource list of a device.
- *PermitResourceAccess*($D1$, $D2$, R): permit a device $D1$ to access a resource R hosted by device $D2$.
- *RemoveSupervisor*(K^S_{pub}, τ_S): to deregister a node S as supervisor.
- *RemoveDevice*(K^S_{pub}, τ_S, τ_D): to deregister a device D from the network by the supervisor S.

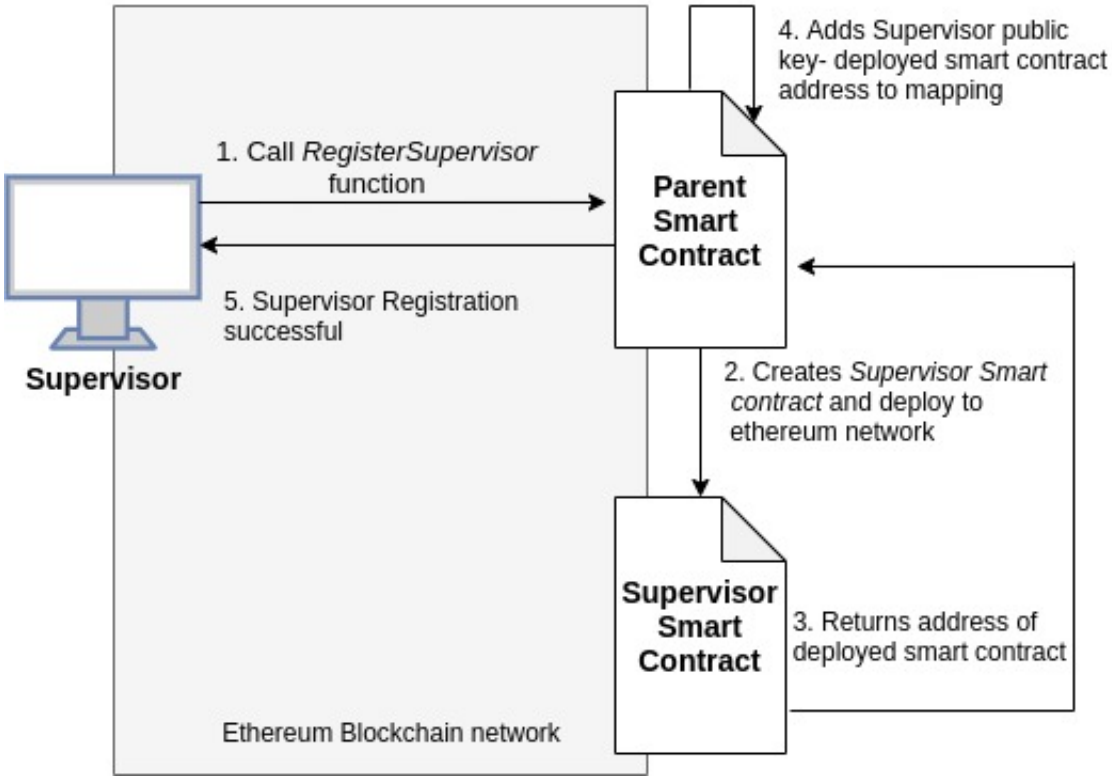


Fig. 3.3 Showing the process of Supervisor Registration as part of network setup

3.4.3 System Interaction and WorkFlow

Network Setup

In this phase, the blockchain network is set up. Trusted peer nodes are added in this phase to help maintain the distributed ledger and participate in voting. Some of the trusted peer nodes will register as supervisors. These peer nodes also host file systems for IPFS since they have high computational and storage capabilities. The administrator node in the network deploys the smart contract on the blockchain. It receives the blockchain address of this smart contract which it stores in a *Setup* file in IPFS and the hash of this file is stored on the blockchain. The address of this smart contract can be queried by any valid node from the IPFS which will allow them to send transactions to the smart contract.

Registration of the entities

Any trusted node in the network added by the consortium can register itself as a supervisor. To register, the node generates a public-private key pair (K^S_{pri},K^S_{pub}) using ECDSA algorithm which identifies it uniquely in the network. The Keccak hash function [99] is used to generate a 20-byte supervisor ID, τ_S from the public key. As shown in Figure 3.3, the supervisor sends a transaction to function *RegisterSupervisor* defined in the smart contract.

This function deploys a smart contract for the registered supervisor and obtains the address of the deployed smart contract. The deployed supervisor's smart contract contains the supervisor's metadata which includes the supervisor's public key, ID and an array of devices registered under the supervisor. The parent smart contract will contain a mapping of every registered supervisor's public key to the address of their respective deployed smart contract. This completes the registration of the Supervisor node.

Figure 3.4 shows the process of device registration. For device registration, the device will possess a public-private key pair (K^D_{pri},K^D_{pub}) and device ID, τ_D derived from it. The device will send a registration request to the nearest supervisor, with its public key and τ_D. The supervisor will first invoke function *RegisterDevice* in the parent smart contract, this function calls another function *AuthenticateSupervisor* which takes as input K^S_{pub} and signature of supervisor over τ_S. From the received public key, the supervisor's smart contract will be retrieved taking the address from the parent smart contract mapping. The received signature will be validated against the τ_S stored in the supervisor's smart contract. Only a registered supervisor can add the device to the network. Once validated, the supervisor will be allowed to create a *device ticket*, $\Im$ that will be stored in IPFS as shown in Figure 3.5. The $\Im$ contains all the metadata related to the device which includes K^D_{pub}, τ_D, hashed device attributes $H(A)$ and resource list associated with this device.

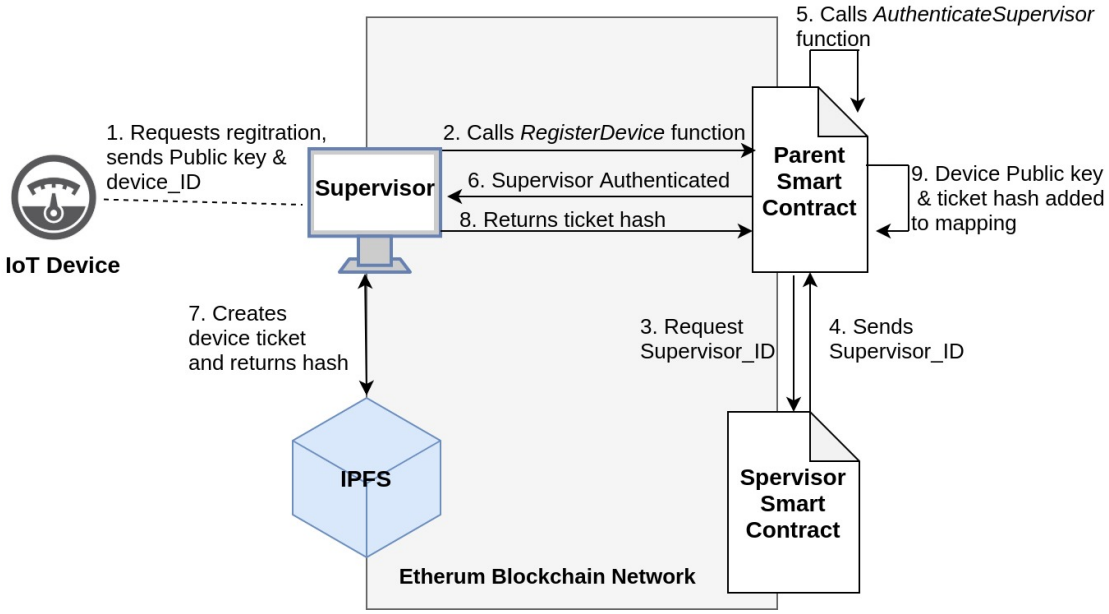


Fig. 3.4 Showing the process of IoT Device Registration as part of network setup

Device attributes A_i where i represents parameters like manufacturer details, firmware version, hardware version, etc. $A = A_1||A_2||A_3.....A_n$, are hashed together to give a unique fingerprint of the device $H(A) = UniqueDeviceFingerprint$. The Unique Device Fingerprint prevents any tampering with the device. On uploading the $\Im$ in IPFS, a hash string

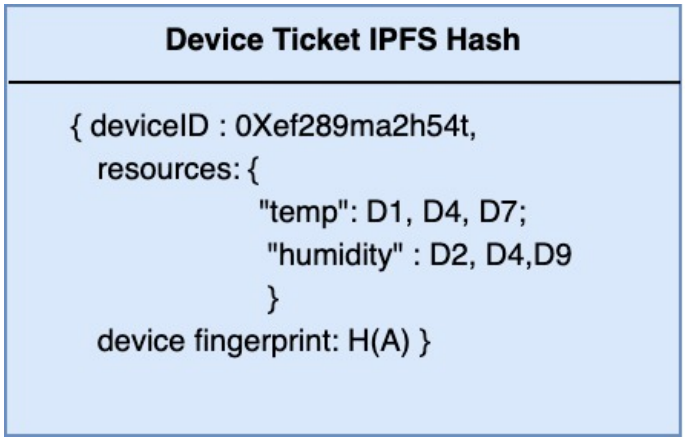


Fig. 3.5 Illustrating the format of a device ticket $\Im$ that contains a unique fingerprint of each device

is returned. This ticket hash is stored in a mapping in parent smart contract along with K^{D}_{pub} and a dictionary of supervisors under which this device is registered. This marks the complete registration of a device in the system.

It must be noted here that taking into account the device firmware to calculate the device fingerprint is a choice that owners of device can make. Having firmware versioning as part of the device's unique fingerprint can help ensure tamper-resistant firmware updates. Hence, only the device owner will be able to update the device fingerprint after a secure firmware update. However, it would also mean that the device would have to re-register every time its firmware is updated if this attribute is kept intact as part of the device's fingerprint, resulting in additional steps each time. So, it's a design feature which can be selected depending on the level of security expected.

Add Resource policies

Supervisors can add resource access policies for IoT devices registered under them. After a device is registered under a supervisor, the access policies can be defined for that device. These policies can be defined in several ways. In our work, all the resources of a device are listed in their respective $\Im$. To set the access permissions, the supervisor first authenticates itself as explained in the device registration process. After successful authentication, the supervisor is given access to $\Im$ location in IPFS to set the resource access policies of the device. The new hash of the ticket is updated in parent smart contract mapping which contains the device public key mapped to $\Im$ hash in IPFS. This creates a distributed registry of device resources which can be accessed by any trusted node of the network to grant access to the devices.

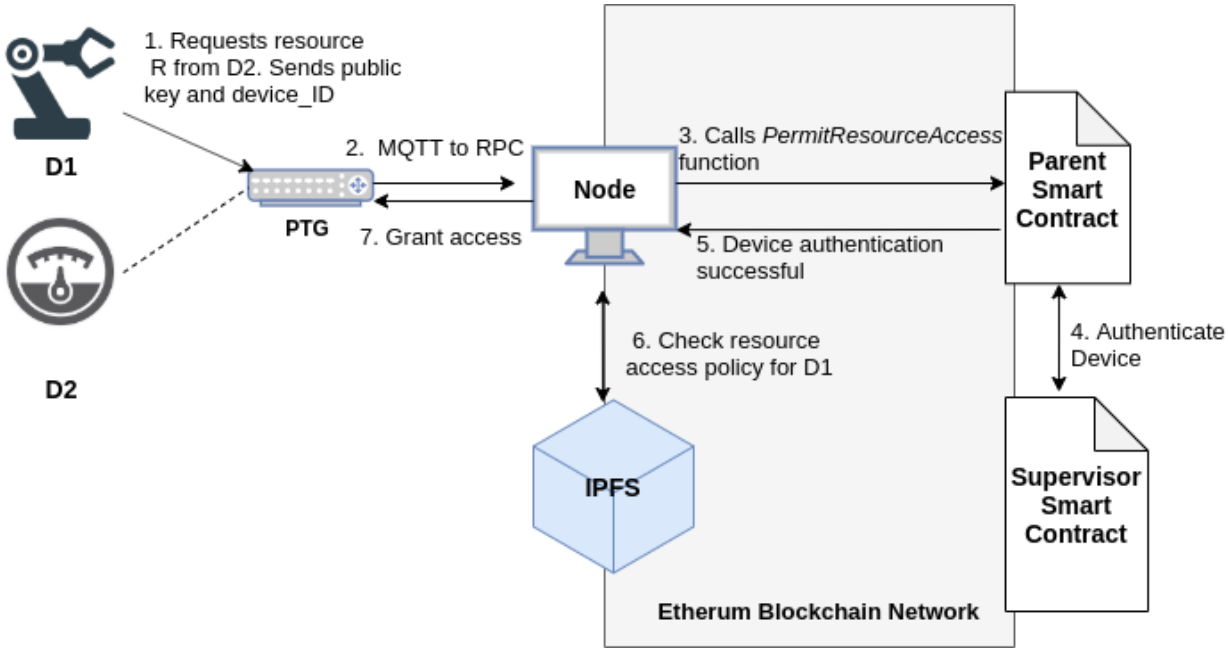


Fig. 3.6 Showing the steps involved in granting access to a device for a requested resource hosted by another IoT device

Permit Resource Access

When a device $D1$ wants to access a resource R hosted by device $D2$, $D1$ sends a request containing its signature over τ_D, $Sig_{K^D_{pri}}(\tau_D)$ and public key to a PTG in the form of an MQTT message requesting the resource. The PTG transforms MQTT message to RPC-JSON request and forwards it to the nearest blockchain node. The node calls *PermitResourceAccess* function defined in the parent smart contract. This function first authenticates device $D1$ and if authenticated, allows the node to query the resource access policy for $D1$ stored in $\Im$ of device $D2$ in the IPFS file system as shown in Figure 3.6. If $D1$ has an access policy defined for resource R it sends the permission to PTG which sends it to $D1$. This allows device $D1$ to connect to MQTT broker hosting resource of device $D2$. Since the query is made to the copy of IPFS hosted on the node, the process is completed immediately. In this case, there is no reliance to receive the information from a centralized storage.

3.5 Implementation and results

3.5.1 Experimental setup

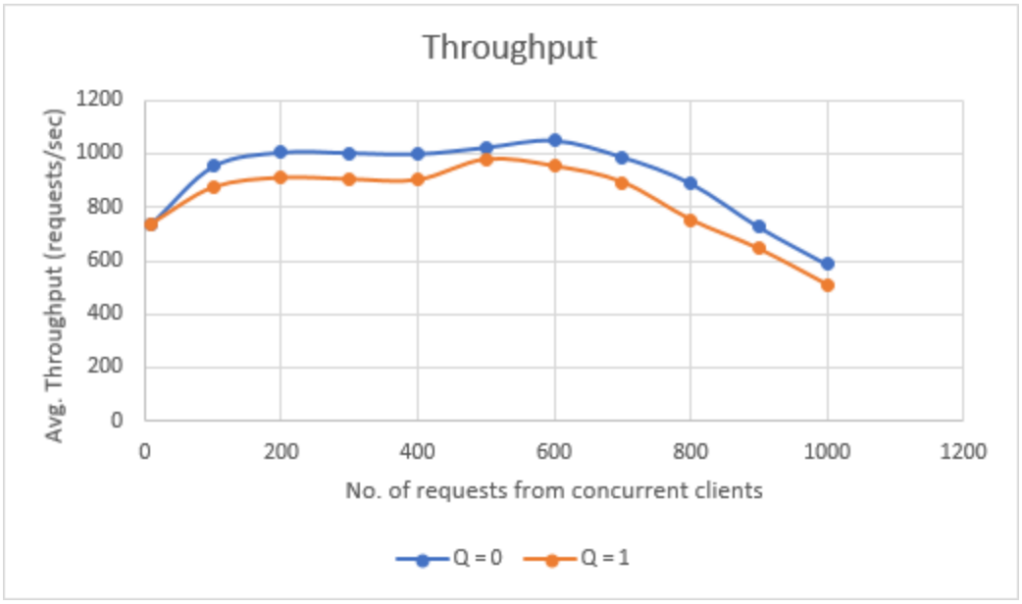
The experiment was set up on a Linux machine installed with the following hardware specifications: Intel® Core™ i5-7200U CPU@2.50GHz and 8 GiB RAM. A private

Ethereum blockchain was setup. Such a private or consortium network can be deployed by a single organization or a group of legal entities coming together on agreement in a real-world scenario. Later smart contracts explained in Section 3.4.2 are developed using Solidity programming language. As explained in Section 3.4.2, each function performs a specific task in the protocol. The developed smart contract is deployed by the *admin* node in the network. Once the smart contract deployed, it is ready for accept transactions for various functions.

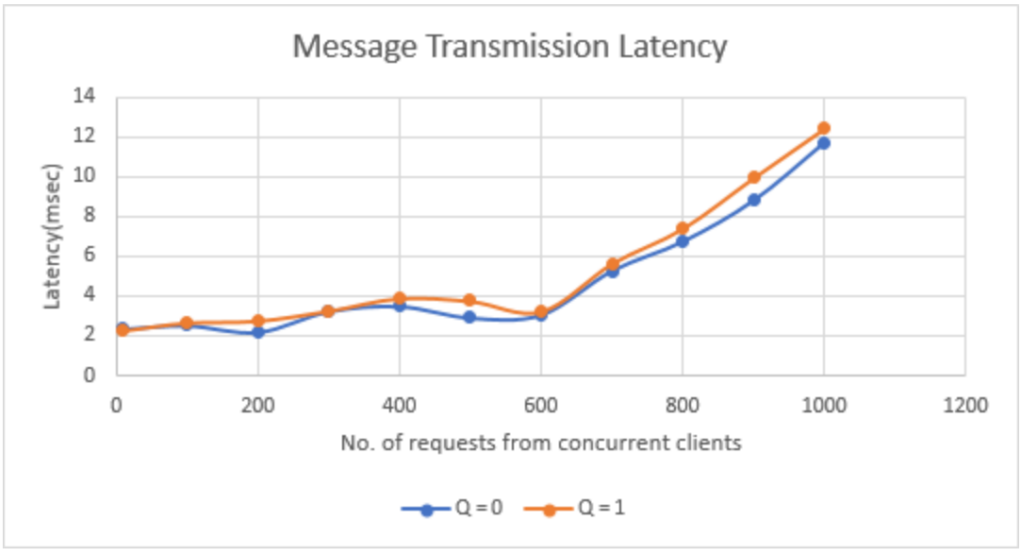
Later, an IPFS network using Go v1.12.0 was installed on the system. In a real world scenario such a network can be an IPFS cluster hosted in an industrial zone if a single party is deploying the framework or just like the blockchain network a group of entities/organizations can host multiple nodes of IPFS network. Once the blockchain network and IPFS network is setup, IoT devices were onboarded in a facility by sending transaction request to PTGs which then send it to the supervisor nodes.

IoT devices are mostly sensors that collect data and this data needs to be hosted for feeding it in various pipelines. This data is the resource that is later requested by another device, an application or even a user. In order to host this data in a distributed fashion, a Mosquitto MQTT message broker v1.6.9 from Eclipse Foundation is installed on the testing machine. MQTT protocol works in a publish-subscribe model where the clients (IoT devices) publish to a *topic*, another term for a specific data stream from sensors. Publishers are clients that publish the data to the broker in specific *topics* and subscribers are the clients that subscribe to the *topics* the application is interested in. This way a continuous stream of sensor data can be published by IoT devices and it can be consumed by other devices or applications. MQTT protocol also provides the ability add an authentication and access control layer on top of the publish-subscribe communication to make it secure. This capability of the protocol is leveraged to add the devised identity framework.

MQTT protocol is selected to implement the IoT devices' communication as it is a lightweight protocol and closely resembles the communication requirement of a real-world industrial setup where real-time data is used to drive machines and make decisions. In order to analyze the performance of the setup, an MQTT benchmark tool written in Python 3.6 called PyMQTTBench from Python Packages is used. The tool was modified to incorporate an authentication process via blockchain and IPFS. This benchmark tool has ability to create several virtual IoT clients concurrently and simultaneously spur resource access request in the network once the devices are onboarded. Such a concurrent set of IoT devices were spurred using the testing framework to replicate the real-world industrial setup where multiple devices are simultaneously communicating with each other or other applications.



(a) Throughput performed on subscribers requesting access control for a resource hosted by a device in the network through *PTG*



(b) Latency experienced in resource access message delivery for the tests performed in Figure 3.7(a)

Fig. 3.7 Performance Results illustrating the Throughput and Message Transmission Latency varying with the number of concurrent client requests. Q = Quality of Service (QoS)

3.5.2 Performance Evaluation

The test is executed as a sequence of three steps. In the first step, the network is set up, a smart contract is deployed, the IPFS node is initialized and keys are generated for supervisors and devices. In the second step, the supervisors are registered followed by the registration of devices. In the third step, MQTT broker is set up and virtual publishers start publishing data concurrently to a *topic* like *"/resource/temperature/D1"*. A large number of subscribers send resource access requests simultaneously to the *PTG* for connecting to MQTT broker for *topic* subscription. Once the authentication is obtained from the blockchain network with the help of device metadata on IPFS, the subscriber can connect to the broker and subscribe to the *topic* to start receiving the requested data. Each test was run ten times and average values were taken.

The goal of the benchmark is to evaluate the impact of a number of concurrent client requests on the network in terms of first, delivered throughput i.e. message rate on the subscriber side and second, message transmission latency i.e. time required to transmit a message to the client. In order to evaluate both these parameters, a parameter intrinsic to MQTT protocol is leveraged and their performance against this parameter is observed as shown in Figure 3.7. This parameter is known as Quality of Service (Q/QoS). MQTT Quality of Service (Q) is an agreement between the message sender and receiver that defines the level of delivery guarantee for a specific message [100]. Q is crucial in MQTT due to its role in providing the client with the ability to select a service level that aligns with both the network reliability and the application's requirements. MQTT's inherent capability to handle message re-transmission and ensure delivery, even in unreliable network conditions, makes Q essential for facilitating seamless communication in such challenging environments. By offering different Q levels, MQTT empowers clients to optimize their network usage and achieve the desired balance between reliability and efficiency. An analysis of the result against parameter Q helps us understand how the system will perform in various level of guarantees.

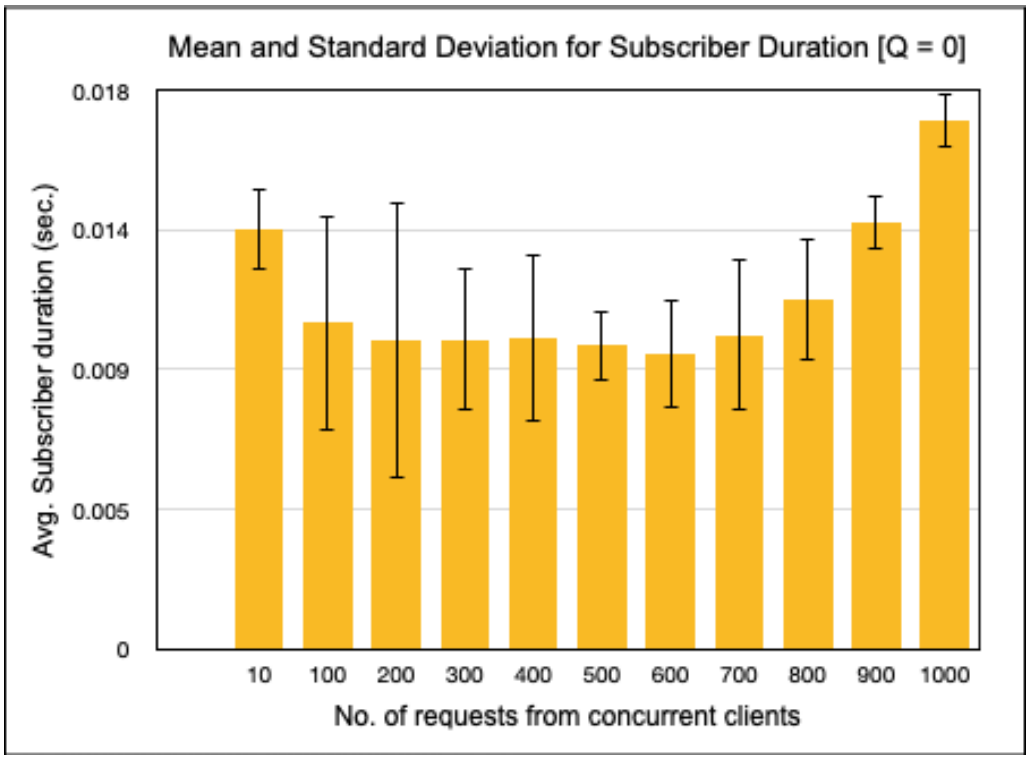
MQTT provides three levels of Q: At most once (Q=0), at least once (Q=1) and exactly once (Q=2). At the lowest level, Q=0 in MQTT offers a best-effort delivery mechanism where the sender does not expect an acknowledgment or guarantee of message delivery; it means that the message may be delivered once or not at all. In Q=1 of MQTT, the focus is on ensuring message delivery at least once to the receiver. This is achieved through an acknowledgement exchange between the subscriber and the broker. In the proposed work, the results are evaluated for first two level of delivery reliability to gauge the performance of the framework.

Figure 3.7(a) shows the results for throughput. The throughput for client requests steadily increases from 750 requests served per second until it achieves a throughput

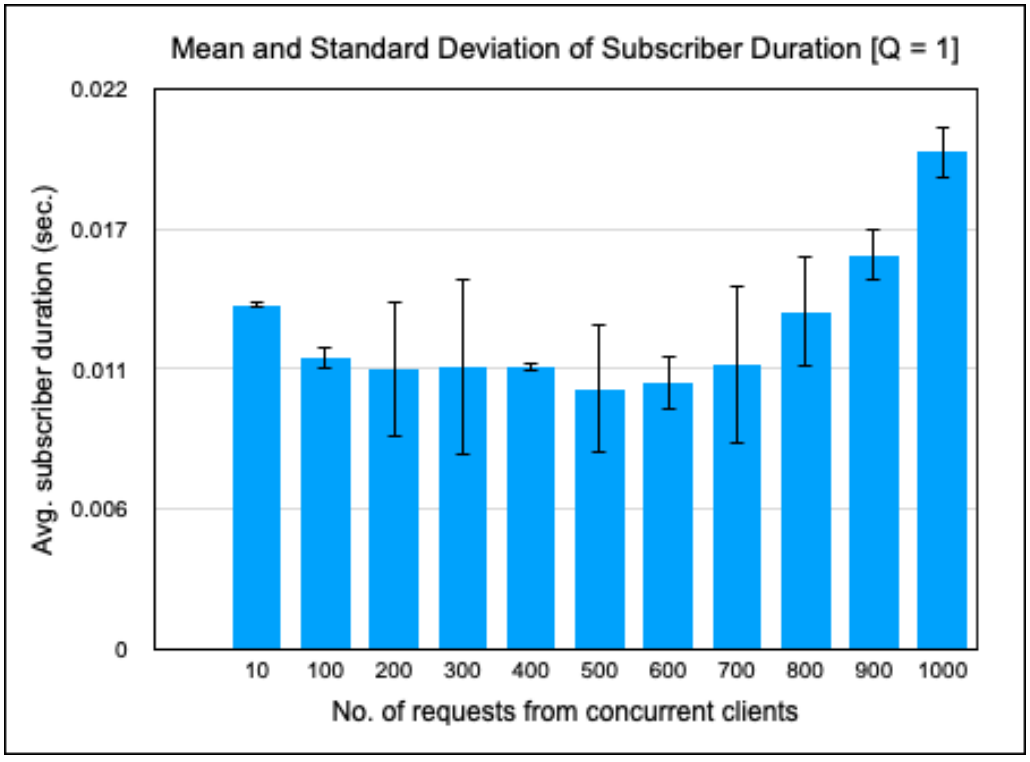
of 1000 requests per second for approximately 600 client requests in the case of Q = 0. Similarly, for Q = 1, the throughput increases from 720 requests to 930 requests per second. However, the overall throughput for Q = 1 remains smaller than Q = 0 due to the fact that after each message delivery, the broker waits for acknowledgement in Q = 1 resulting in lower throughput values. The decrease in the performance is directly related to the increase in message transmission latency as shown in Figure 3.7(b). The latency for Q = 1 is always greater than that for case Q = 0 due to the acknowledgement that takes place after each message exchange and authentication process completion in the blockchain network. The latency in both cases remains well below 4 msec when the number of requests is below 600 but it increases with an increase in the number of requests.

A PTG in the proposed protocol is acting like a protocol translator. IoT devices cannot be part of a blockchain network due to a number of reasons like lacking in capability to host blockchain ledger and constraints regarding participating in the consensus mechanism. Hence, it is not possible for IoT devices to communicate directly with a blockchain network in order to register itself with the supervisor nodes. Therefore, IoT devices need an intermediary to communication with the blockchain network. PTGs perform this role and handle the two way communication between IoT devices and supervisor/peer nodes. In the test bed setup only one PTG is considered to generate results and its effect can be observed. A greater throughput and better latency can be achieved with a higher number of *PTGs* in the network and faster authentication processing in blockchain and IPFS networks. Nevertheless, the performance of the overall setup performs well with up to 1000 requests served per second for 600 clients. It is possible to forgo PTGs if the IoT devices directly wish to communicate with the blockchain network, in which case these devices should be able to act as nodes in the blockchain network which would require them to be significantly computationally competent. Therefore the inclusion of PTGs (an intermediary) in the design provides a tradeoff between efficiency while keeping an intermediary or a computationally intensive design pattern.

The reliability of any data set is very important to gauge the authenticity and reproducibility of results. Therefore, the dependence of subscriber mean duration on an increase in total requests made by clients is tested. Figure 3.8 is plotted with respective standard deviations in the form of error bars over each set. The average subscriber duration is the mean time taken by the subscribers to receive the resource requested. The observed behaviour shows in Figure 3.8(a) that for Q = 0, the mean subscriber duration is lowest between 200-700 and the standard deviation is also relatively small for this range. Henceforth, the system performance will be better for this range and this can be confirmed with Figure 3.7(a). A similar pattern is observed for Q = 1 in Figure 3.8(b). However, the mean value of subscriber duration remains on the lower side for Q = 0 as that of Q = 1. This directly affects the throughput of the system as seen in Figure 3.7. However, the standard



(a)



(b)

Fig. 3.8 Statistical Results illustrating the Mean and Standard Deviation(SD) of subscriber duration against the number of concurrent client requests

Table 3.2 A table summarizing susceptibility of each component to a set of threats under STRIDE model

	S	T	R	I	D	E
IoT Devices					X	
PTGs	X	X	X		X	
Supervisor Nodes						
IPFS nodes						

deviation for Q = 1, is always less for a particular number of requests as compared to Q = 0, which means that the mean values for Q = 1 are more clustered around the mean as that in Q = 0. The smaller the standard deviation, the more reliable the data. Hence, the results are in accordance with statistics.

The overall performance of the system can be considered acceptable considering that with the increase in *PTGs* the performance will improve. And a WSN will have several *PTGs* interacting with supervisors.

3.6 Security Analysis

Security is a prominent feature of any architecture and our proposed architecture should be no exception to this. Even though the aim of our design is to facilitate access control in constrained scenarios, the solution should provide a satisfying level of security. In this section, the main possible threats in the architecture are identified and solutions are provided to achieve the best level of security. To identify the threats in our system, the STRIDE [101] model is used by asking whether one or more threat types apply. STRIDE classifies the threat into six categories, and its acronym derives from it: spoofing, tempering, repudiation, information disclosure, denial of service and elevation of privileges. In our architecture, there are four main entities: IoT devices, *PTGs*, Supervisor nodes and IPFS nodes. The susceptibility of each to the above threats is identified.

Spoofing attack takes place when a malicious entity tries to impersonate another device in the network. The IoT devices and supervisor nodes are not susceptible to this attack as they possess a public-private key pair, which they use in all the interactions with the blockchain. However, since IoT devices are not a part of the blockchain network and their communication with Supervisor nodes depends on *PTGs*, a malicious *PTG* can change the information to and from IoT devices. To overcome this issue, the *PTGs* can be assigned signed certificates by a trusted authority in the network. Each IPFS node has its unique identity created at the time of installation and hence can prevent spoofing.

Tampering of access control information can be done by a malicious entity posing as *PTG*, but as proposed earlier, signed certificates can help mitigate the attack. Tampering of device metadata in IPFS and transaction records in blockchain is difficult because both are cryptographically protected. In IPFS, each change to the *device ticket* $\Im$ is stored on the blockchain as a hash of the file and any tampering with the data will result in a different hash. Similarly, in blockchain, each block is cryptographically linked to the previous block making tampering highly difficult.

Repudiation refers to threat action aimed at performing prohibited operations in the system that lacks the ability to trace the operations. Non-repudiation property is a prime feature of blockchain technology as the sequence of events is logged in the form of transactions in the blocks. In our architecture, direct interaction between IoT devices is not encouraged, and this will prevent any spoofing attack by one device on another. A malicious *PTG* can repudiate (can be claiming to have not performed an action) and signed certificates can solve this issue.

Denial-of-service (DoS) attack is possible on IoT devices and to avoid it, the devices must be non-reachable from an open network. To ensure this, firewalls can be installed in the network to prevent any malicious entity from outside the network from launching DoS attack. Also, the unique identity of devices can help mitigate DoS attacks on the network. If devices do not have unique identities and are bootstrapped with default usernames-passwords, they can become an easy target of malicious attackers trying to flood the network with fake connection requests. The blockchain network in general provides a certain level of security against malicious nodes. Moreover, since the blockchain in our architecture is a permissioned network, only trusted nodes are added to the network.

3.7 Conclusion

This chapter presents the design, implementation and evaluation of a novel decentralized identity and access management framework. The chapter addressed the issues in current Industry 4.0 implementation where IoT technology is completely based on a centralized ecosystem hence causing scalability issues on horizontal and vertical scales. In this chapter a novel identity management solution is proposed which is decentralized and based on decentralized technologies like blockchain and IPFS. Due to the resource-constrained nature of the IoT device, it is not made a part of a blockchain network and communication with it is achieved using *PTG*. This makes the integration of IoT technology and blockchain technology adaptable with no major changes to the existing infrastructure.

To facilitate the discussion of the proposed analysis, first the recent state-of-the-art models in IoT is presented and their limitation is discussed. The implementation of a

proof-of-concept prototype and its thorough evaluation have proved the feasibility of our approach. In particular, the solution performs as expected in terms of throughput and latency, along with the ability to provide a small code footprint while handling high loads. The goal of this work was to provide a decentralized and secure architecture for managing the identities of IoT devices in the network with no single point of failure. Such an architecture has been proposed in this work with results.

Chapter 4

A Decentralized IoT Identity Framework based on Self-Sovereign Identity and Blockchain

This chapter constitutes the second contribution of this dissertation. The key information in this chapter is based on the work published in Publication II. The state-of-the-art analysis in Chapter 2 of the thesis outlines the several types of identifiers used for IoT devices. While each of these identifiers is unique, interoperability among devices using such identifiers cannot be guaranteed. Such identifiers are also purpose-limited and vertically as well as horizontally siloed. This essentially means that for each layer in IoT architecture, a new identifier is defined which is not interoperable among these layers or with other outside networks. This chapter presents a new standard of managing and implementing digital identity, the self-sovereign identity standard (SSI). This marks an improvement over the last chapter that formulates the hypothesis of a decentralized digital identity framework. This chapter proposes a novel framework to adopt such a model in the IoT ecosystem. The framework is designed to be independent with regard to the underlying IoT protocols, and its performance has been assessed to study the underlying overheads.

4.1 Background and Motivation

The fundamental requirement for interaction between digital entities is a secure and privacy-preserving digital identity infrastructure. Traditional approaches usually rely on centralized architectural components such as Certificate Authorities (CAs) and credential storage databases that have drawbacks like a single point of failure, attack-prone honeypot databases and poor scalability. Cross-domain interaction between smart devices in different industrial settings also remains a challenge. Siloed communication channels and secure

exchange of credentials are some of the existing challenges in machine-to-machine (M2M) communication. Self-sovereign identity (SSI) is a novel decentralized digital identity model that uses Decentralized Identifiers (DIDs) and Verifiable Credentials (VCs). Such a model does not require a centralized trust authority to hold validation evidence. In this work, a novel decentralized identity framework is proposed for Industrial Internet-of-Things (IIoT) based on the SSI model. The proposed framework is implemented on two well-known blockchain platforms *Ethereum* and *Hyperledger Indy* to study the underlying overheads. These platforms were selected for implementation as they have a large development community and a mature code base to test various functionalities.

With an increase in automation, machine-to-machine (M2M) communication has become pervasive in various sectors. However, in complex IIoT settings, interactions are not limited to intra-device or intra-user communication but extend to many heterogeneous forms. Hence, with these evolving interaction paradigms, the idea of digital identity for human and non-human entities needs to broaden too. In the case of users, a common method to assign identities is to create password-protected digital accounts, with the former stored on a centralized web server. So, when smart IoT devices started making their way into markets they were also assigned with default weak passwords which led to large-scale attacks like Mirai botnet [14]. In today's scenario where humans and machines interact seamlessly unlike never before, the idea of a centralized identity infrastructure poses a challenge.

Since devices are largely unattended in IoT scenarios, password-based manual approaches are unsuitable for inter-device communication. In fact, around 80% of IoT devices fail to implement passwords securely [102]. Hence, an automated solution like Public Key Certificates (PKC), which binds the identity of an entity to its public key became popular among IoT vendors. However, certificate standards like PGP and X.509 were not designed with privacy in mind, as is evident with the use of subject names in their certificates. With PKC, lack of privacy is an inherent challenge. Another disadvantage is the high level of dependency on centralized architectures to maintain the root of trust. Therefore, there is a need for a decentralized approach to digital identity for smart devices, such that, devices from multiple stakeholders can interact seamlessly with other devices, users and organizations in a collaborative environment.

A new model of identity management is shaping in the community termed "Self-Sovereign Identity" (SSI) [103]. The key concept behind SSI is that subjects (identity owners) can fully create, build and control their identity and associated credentials rather than relying on a third party to manage them. It has advantages like better subject privacy and control of when and how their identities are shared. To realize SSI, new standards have been proposed by W3C [104], namely, Decentralized Identifiers (DID) and Verifiable Credentials (VCs). For example, in the case of individuals, their passport is a physical

national identity and provides them with a claim to secure travel permissions off-shore. Similarly, DIDs allow subjects to own online identities and VCs are online representation formats of cryptographically verifiable digital credentials. DIDs in most cases leverage the decentralized blockchain technology as distributed public key infrastructure (DPKI), although other options are open for adoption.

There are four major design principles of Industry 4.0 [105] namely, *interconnection* that enables machines, sensors and people to communicate with each other, *transparency* in decision making by using comprehensive collected data to take an informed decision, *decentralized decision* that will allow cyber-physical systems to act on their own and take autonomous decisions as much as possible and *technical assistance* to assist humans in informed decision making to ensure safe industries. These principles clearly imply that IoT technology requires a decentralized and autonomous framework.

In this chapter, a decentralized digital identity framework is described using the *principles of Self-Sovereign Identity* to securely bootstrap IoT devices in the network. The proposed model facilitates faster on-boarding of a large fleet of devices in a secure, scalable, privacy-preserving and easily verifiable manner. The focus of this work is to decentralize the way in which machine/smart device identities are created, provisioned and managed. Blockchain technology is used as a DPKI and a decentralized storage platform is used to store large files off-chain and put immutable, permanent links in transactions time-stamping and securing content without having to put the data itself on-chain which can impede its performance. The key contributions of this work are:

- A novel proof-of-concept solution is proposed for IoT devices bootstrapping that supports digital identity creation as DIDs, issuing of VCs, verification of VCs, revocation of DIDs and revocation of VCs all in a decentralized manner by leveraging an innovative combination of blockchain and a decentralized data storage layer.

- The proposed model presents a systematic way of applying DIDs and VCs to IIoT for its identity lifecycle management as it extends the capabilities of machines to function autonomously in complex real-case operations.

- The usability of such a system is demonstrated by explaining a use-case scenario i.e. to establish secure remote connections for M2M communication in IIoT.

- A proof-of-concept (PoC) is implemented to test the feasibility of such a solution. The testing on two different blockchain platforms is performed for a realistic comparison. These platforms are Ethereum [59] and Hyperledger Indy [106]. IPFS [76] is leveraged as the decentralized off-chain data storage layer.

- The proposed framework is modular and can be easily embedded in current systems. The performance is studied to present computational and storage overhead incurred for the proposed methodology.

4.2 Current Practices and State-of-the-art Identity Management Models

Today, PKI certificates are predominantly used in IoT devices for registration to assign unique identities that are later used for encryption and authentication purposes [107]. They enable devices to perform data transfer, remote communication, firmware updates etc. There are numerous types of certificates [108]. *SSL certificate* authenticate servers to devices/machines. *Code-signing certificate* are used to sign software and prove the legitimacy of widely distributed software packages. *Client Certificate* identify users/devices/gateways over intranet/extranet and access resources like databases. These certificates proved scalable for small and medium-sized deployments but for large-scale IoT deployments they face issues [109] like high cost and performance bottleneck. Certificate expiration and certificate revocation is another major drawback that stems from the fact that certificates are firstly derived from a single trust anchor called root certificate and secondly, they are stored in a centralized location. When these certificates expire a CA can revoke the device certificates by putting them in the Certificate Revocation List (CRL) without informing the applications they are interacting with. In case of incorrect revocation or a malicious act, the devices will be abandoned, where the enterprise will have no control over them [110]. Since IoT devices are mostly deployed in physically inaccessible and remote locations, frequent re-installation of such certificates on revocation is an expensive task. And for critical infrastructures like safety and surveillance, this can prove fatal and cause massive losses.

In the early years, online entities were primarily created for individuals, however, in recent years there has been a paradigm shift towards M2M communication. Such transformation in communication topology has kindled the quest for practical, secure and privacy-preserving digital identity infrastructure. Over the years identity management models have also evolved. The first model comprised centralized authorities acting as issuers and authenticators of digital identity. Organizations like ICANN determined the validity of domain names and later CAs created and stored identities for entities. This led to excessive authority vested in a few corporations with little control left to the users.

The second model known as federated identity allowed multiple service providers to form a federation with one of the identity providers, allowing the user to use the same credential across these platforms. This led to great control and usage of data tracking by

technology giants. There have been multiple cases of identity breaches where data of a large number of users was compromised [17], [15].

The third model known as user-centric identity focused on two major elements: user consent and interoperability. The user was kept at the centre of Identity Management Standards (IDMS), and was free to decide with whom and when to share their identity. Standards like OpenID (2005), OAuth (2010), FIDO (2013), OpenID Connect (2015) and CTAP 2.0 (2018) were introduced for creating user/device authentication. However, these standards use OpenID servers to store passwords/credentials. The most recent model called Self-Sovereign Identity (SSI) [111] gives complete autonomy of its identity to the subject itself. SSI is built on ten core principles, namely: existence, control, access, transparency, persistence, portability, interoperability, consent, minimization and protection. These principles are drafted by Christopher Allen and are influenced by Kim Cameron's laws of identity [51].

4.2.1 Use of DIDs for IoT Devices Identity

Even though SSI is a relatively new digital identity standard still under development, the feasibility and advantages of using SSI have become a topic of research in various contexts. Soltani et al. in [112] propose a client onboarding and KYC (Know Your Customer) process using SSI and distributed ledger technology. They make use of Hyperledger Indy to propose a KYC framework and evaluate their framework against SSI and GDPR principles. Othman et al. [113] present a *Horcrux* protocol that combines DID with Biometric Open Protocol Standard (BOPS). It enables the SSI subjects to control the process of accessing their identities by giving consent to this verification process via biometric authentication. It is claimed that the protocol helps in the secure exchange of user-controlled biometric credentials.

In IoT, several use cases for SSI have been proposed by researchers for instance facilitating interaction between users/devices, automated authentication and authorization between users and devices, part life-cycle management etc.[114]. Lagutin et al. [115] propose the use of DIDs in an IoT scenario through a use-case of a trusted printing service in a university, that needs to be accessed by authorised users without leaking any confidential information. In the proposed work they use a proxy approach and delegate the processing of DIDs from IoT devices to the OAuth Authorization server by extending the ACE OAuth flow. In [116], the authors present a comparative study for using SSI for IoT environments against standards such as PGP (Pretty-Good Privacy) Keys and X.509 certificates. The comparison is made in terms of the types of identifiers used, the uniqueness of the identifier, endorsement policy, privacy-preserving techniques and any dedicated service points available.

The use of DID for IoT applications was discussed in [117].The work indicates the feasibility of DID even on devices with limited resources. IoT-as-a-Service (IoTaaS) concept to offer IoT device services using DID and VC on a demand basis has been proposed by [118]. A solution based on DID and VC that enables a distributed V2X (Vehicle to everything) access authorization mechanism to allow a vehicle owner to prove its identity without compromising privacy was proposed in [119]. While DID enables authentication and user identification, VC can verify a set of claims that can be verified against access policy to allow secure and granular authorization [120]. The authors in [121] present a privacy-sensitive information protection and management scheme to allow the automated distribution of information.

The above discussed works clearly establish the merits of use of DIDs and VCs for IoT devices identity lifecycle management. SSI via use of DID and VC offers several advantages in a multi-stakeholder environment where parties inherently do not trust each other and wish to achieve transparency, and non-repudiation and avoid siloed identity vertical. This theme and several aspects of SSI have been explored in the above discussed works. It has been used in conjunction with biometrics and other identifiers to allow seamless integration of IoT devices for human facing applications. The authentication flow of DIDs and VCs is also explored with traditional protocols like OAuth. Although the above discussed works try to address different issues related to IoT device authentication they are mostly focused around device to user authentication and vice-versa. Device to device authentication in a remote setting is not explored in depth in these works. Additionally, a majority of these works perform computation on a single blockchain platform and do not present any comparison among different decentralized identity supporting blockchain. The concept of wallets and DID methods and its importance is also not discussed in these works. In some of these studies, lack of evaluation and simulation results does not allow for critical comparison of work with other similar approaches.

4.3 Proposed Model

This section describes the proposed approach and discusses it in detail. Figure 4.1 illustrates the reference architecture. The idea behind our scheme is to move away from a centralized PKI for interconnected device identity life-cycle and provide a decentralized digital identity model leveraging SSI standards. Such a model is described to illustrate remote M2M communication.

4.3.1 System Actors

The primary actors with their respective roles in the system are as follows:

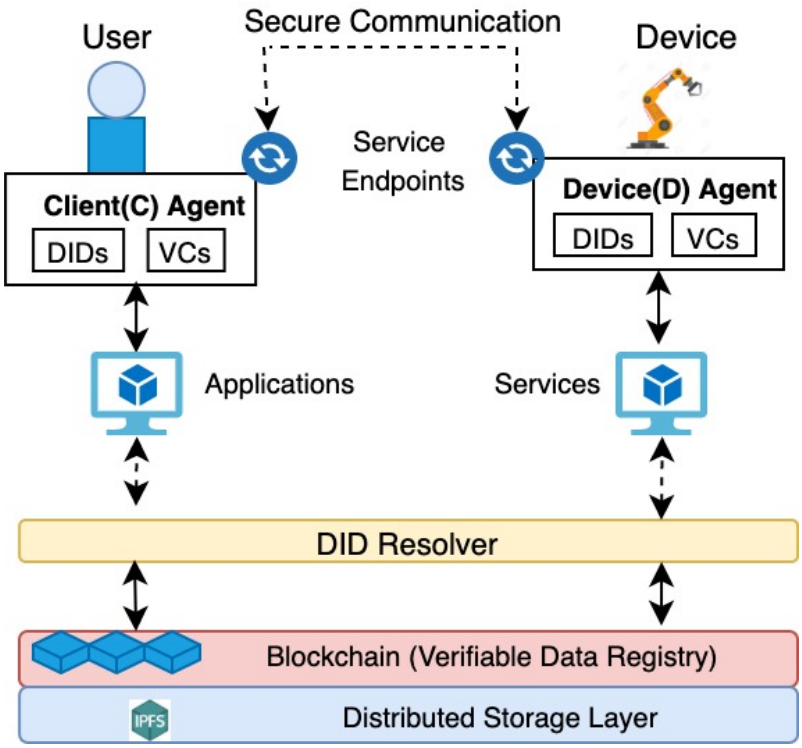


Fig. 4.1 Reference Architecture for proposed framework showing M2M communication carried out by IoT device agent

1. **Subject/Holder** (*S*) can be a machine (including sensors, mobile machines, manufacturing units, network gateways etc.), a user or even an enterprise that holds a unique identity in the network. These heterogeneous actors will henceforth be called by a common term, *subject* (*S*). These actors connect to each other in order to exchange messages/data, authenticate to prove their identity, gain authorization to access restricted services, remotely log in to applications/devices in isolated environments etc.

2. **Issuers** (*I*) are trusted actor in the system that issues unique identities i.e. DIDs to the *subjects*. *I* also holds the authority to issue digitally verifiable claims, VCs to the *subjects*, which can be used to prove their access rights in the network. The issuer can be a government agency, a manufacturing unit, an OEM (Original Equipment Manufacturer) etc. and each issue statement is logged in a verifiable data registry (VDR) as a transaction.

3. **Verifier** (*V*) are software actors that form the interface of a resource and verify the VCs presented by *S* to grant them access. It can be a cloud or edge application that performs verification with the help of VDR.

4.3.2 Components of the architecture

This section discusses the role of each component in the architecture:

Agent

Each *subject* hosts an *agent*. An *agent* is a software process that acts on behalf of the *subjects*. It offers persistent addressable service endpoints to securely communicate with the *subjects*. The *agent* also processes *subject's* request to the distributed ledger. The agent's capabilities and structure will depend on various factors like where it is deployed, how many endpoints it needs to serve, how many VCs it needs to store and process for presentation and other such operational features. While the term *agent* generally represents a process that initiates and handles all the communication for a subject, a "wallet" has better operational inference. Hence the term "wallet" has been used interchangeably with "agent" henceforth and they both refer to same underlying process in the context of this work.

Verifiable Data Registry (VDR)

A verifiable data registry stores references to the DIDs and VCs issued by *I* to *S*. Depending on VDR implementation, it can include some or all of the following data: DID, credential schema, credential definition and a revocation registry. In some VDR implementations, issuers may put a schema on the VDR that says in effect, "When I issue a credential, it's going to have these fields in it". Similarly depending on VDR, before an issuer can issue a credential using a schema, they must put a credential definition on the VDR which says "I am going to use this schema for my credentials and here are the public keys that I am going to sign the claims when I issue a credential" [122]. In this work, two different VDRs are used for a comparative study. One is Ethereum blockchain [59] and the other is Hyperledger Indy blockchain platform [106].

Smart Contracts

These are automated computer programs or a transaction protocol embedded in blockchain which triggers when a transaction is sent to them. They can execute, control and legally automate events defined in them according to the terms of an agreement between transacting parties. For the Ethereum blockchain implementation, the *Ethereum DID Registry* contract is used as shown in Listing 2 that acts as a registry for key and attribute management of lightweight blockchain identities. The *Verifiable Claims Registry* contract as shown in Listing 3 is a secure place to hold VCs issued to *subjects*. In order to implement this system, ERC1056 (Ethereum Lightweight Identity) [123] and ERC780 (Ethereum Claims

Registry) [124], two standards proposed by the Ethereum community for SSI are leveraged. As shown in Listing 4.1 ERC1056 describes a standard for creating and updating identities with limited blockchain resources. As shown in Listing 4.2, ERC780 allows persons, smart contracts and machines to issue claims about each other, as well as self-issued claims.

Hyperledger Indy does not host smart contracts. Rather than storing data in the ledger and then providing access to that data using smart contracts, Indy enables subjects to own the data and share it in a way that preserves their privacy. Hyperledger Indy identities can be referenced in smart contracts from other systems which enhances its interoperability with other decentralized identity system [106].

Distributed Storage Layer

This is a layer of peer-to-peer connected nodes that form a decentralized system for storing, linking and transporting data. In the proposed model IPFS is as the distributed storage layer.

```
contract EthereumDIDRegistry{
  % add owner of the DID holder
  function addOwner(address owner, bytes32 ipfsHash)
  % returns owner of the identity holder
  function identityOwner(address identity)
  % change the owner of DID
  function changeOwner(address identity, address newOwner)
  % check signature of identity holder
  function checkSignature(address identity, bytes32 signature, bytes32
   hash)
}
```

Listing 4.1 Ethereum DID Registry

```
contract VerifiableClaimsRegistry{
  % add claim to subject, key = type of claim, its value
  function setClaim(address subject, bytes32 key, bytes32 value)
  % retrieve claim for a subject
  function getClaim(address issuer, address subject, bytes32 key)
  % remove claims for a subject
  function removeClaim(address issues, address subject, bytes32 key)
}
```

Listing 4.2 Verifiable Claims Registry

4.3.3 Framework Explained

Let us analyze our use case to understand the working of the proposed model in an IIoT setting. A high-end surveillance IP camera is deployed in a remote location and it uploads location-sensitive images to a server in a secure facility. There can be a network of such cameras that interact with the server to upload the data. There are two primary functionalities served by a camera in such a setting. First, an authorized device/application can remotely access this camera to make changes to its firmware or tune setting parameters. Second, it is important to make sure that only a verified camera uploads images to the server, as this data will further be used for critical decision-making on the server end.

At the time of provisioning, *subjects* create a unique identifier, *DID* in their wallet, creating an immutable record of the operation on the blockchain. Every time a DID is generated by the subject, a corresponding public-private key pair is generated. After this, a DID is registered with a chosen DID method as explained in Section 2.4. The methods with which a DID is registered defines a set of specifications by which a particular DID and its associated DID Document is created. Consequently, a DID Document is next generated by the subject. This document enlists an array of public key records containing public keys used for authenticating, authorizing updates to the DID Document, or establishing secure communication with service endpoints as and when they are generated. The respective DID Document for each *DID* is created and stored on IPFS. It is to be noted that in order generate and process DIDs and VCs, an IoT device need to have a firmware capable of handling related operations. An IoT device must be equipped with storage, computational ability and capability to establish connectivity with other network entities. Additional security capabilities like secure boot, key management (creating and storing private keys) using HSM and TPM/SE will allow secure functioning of the devices.

In Ethereum, IoT devices and users (or any *subject*) create their *DIDs* in the form of Ethereum Externally Owned Accounts (EOA). The created *DIDs* are registered on *EthereumDIDRegistry* with the hash of DID Doc content returned by IPFS using the *addOwner* function. The hash provides integrity of DID Docs and cryptographic bindings to the identifier defined by the smart contract. This smart contract also provides a means to change the identity owner and verify the identity owner. In Hyperledger Indy, *subject* creates a new DID record in the wallet and then sends an *NYM* transaction to the blockchain to make a record of this identity creation. *NYM* transactions are special types of transactions in Indy for the creation of new DIDs, setting and rotation of verification key, and setting and changing of roles.

Once, the *subjects* in the network are equipped with unique identities, *VCs* can be issued to the devices and applications attesting to its legitimacy, access privileges and roles. Claims are issued by an *issuer* signed with their private keys, to a *subjects* with a *key* that

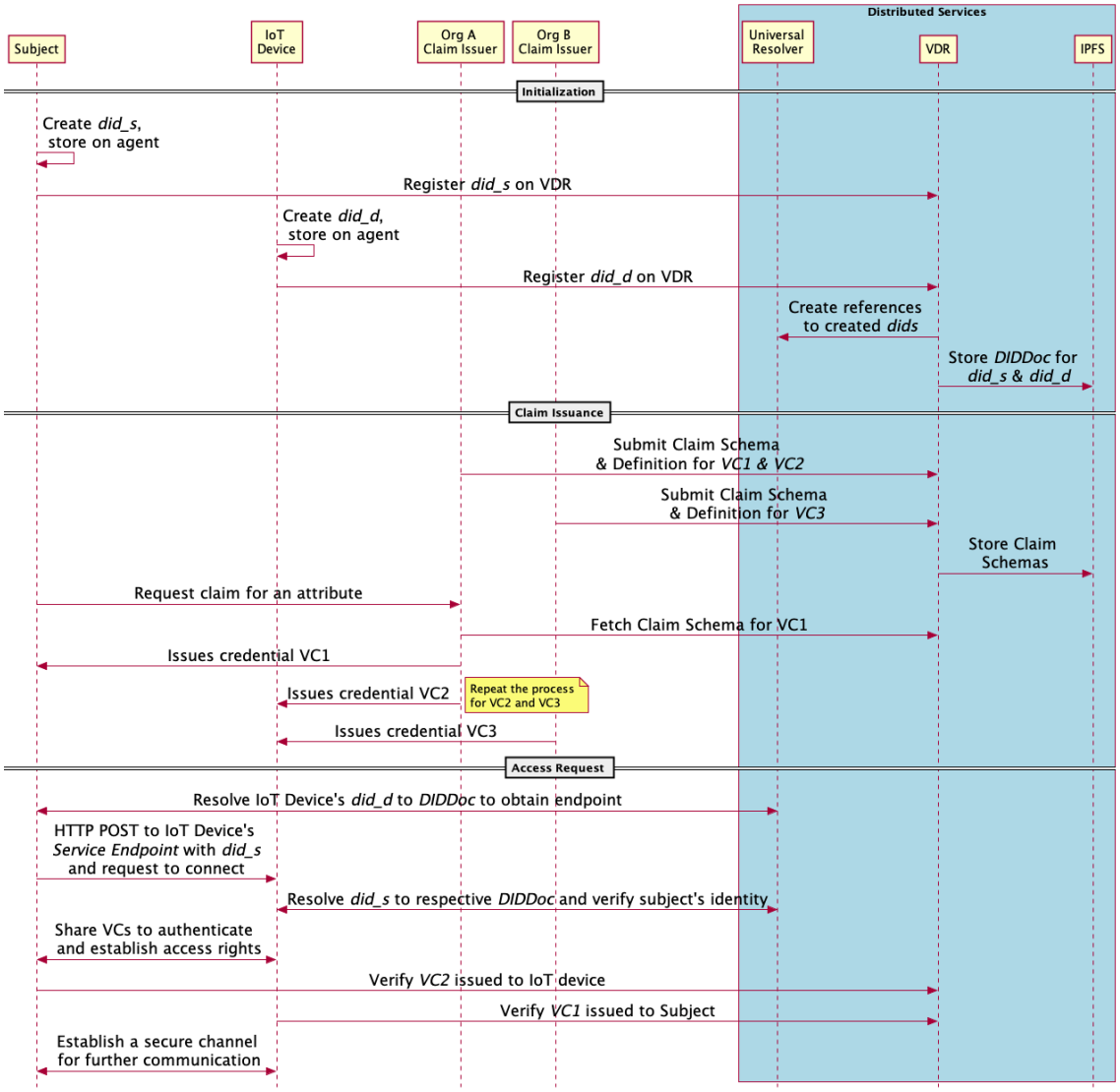


Fig. 4.2 Flow Diagram illustrating the interaction between the different components of framework

indicates the type of claim. The claims are portable and reusable enhancing interoperability. The same process is repeated for other *subjects*. These steps are performed during the first time provisioning of the *subjects* in the operator's network and before the device leaves the operator's premises. Different blockchains have different mechanisms to issue claims. For Ethereum, the claims are registered on the *VerifiableClaimsRegistry* contract using the *setClaim* function. This contract has an interface to add, get and remove claims and all of these tasks are performed automatically unlike a digital certificate ecosystem. Whereas in Indy, the *VCs* are issued by *TrustAnchors* or trusted *issuers* in the form of *NYM* transactions recorded on the ledger.

Once the device is deployed in a remote location, it's an essential part of device management to remotely communicate with the device for tasks like data access, device

parameter tuning, firmware upgrades etc. For a *subject*, in order to access an IoT device remotely, it executes the processes as listed below, also shown in Figure 4.2:

1. The *subject* first resolves the *DID* of the IoT device it wants to remotely communicate to, through the *Universal Resolver*. The DID Doc of the IoT device is returned to the *subject* by the IPFS. The list of device *DIDs* can be maintained in the form of a directory by the provisioning party.

2. The *subject* sends a request to the IoT device for establishing a secure connection on the endpoint described in DID Doc of the IoT device under the "Service Endpoint" list. The *subject* sends its *DID* in the request.

3. The IoT device resolves the received *DID* of the *subject* through the *Universal Resolver* and verifies the identity of the *subject*. Until now, the device only knows that this *subject* is a part of the operator network as they share a similar *DID method*.

4. The IoT device and subject request each other to share their verifiable claims i.e. *VCs* for mutual authorization. The *VCs* are shared and both the subject and IoT device verify them through the underlying VDR.

5. The IoT device verifies if the *subject* possesses the authority to make any changes to device settings and *subject* verifies if the device is what it claims to be.

6. If shared claims are legitimate, the IoT device sends another endpoint to the *subject* on which it can communicate securely with the device.

Through this series of interactions using DID and VC, two discussed functionalities for secure remote communication can be achieved by a remote camera. Firstly, it can verify any incoming communication in order to make changes in its firmware. A new VC can also be assigned to the remote camera using these steps by the owner of the device. Secondly, using its VC, the camera can verify its authenticity each time it tries to upload an image on a remote server. In the second case, the subject is the server and it can skip the steps to fetch the DID document each time a camera tries to upload data.

4.4 Evaluation and Discussion

4.4.1 Evaluation Results

Extensive experiments have been performed on the proposed framework to understand the feasibility and robustness of the network. The experiment was set up on a Linux machine installed with the following hardware specifications: Intel Core i5-7200U CPU@2.50GHz

Table 4.1 A table to summarize the architectural difference between the blockchain projects used to conduct tests for the proposed scheme

Features	Ethereum	Hyperledger Indy
Access Type	Public	Public
Validation Type	Permissionless	Permissioned
Smart Contract	Applicable	Not-applicable
Mining	Applicable	Not-applicable
Consensus Mechanism	PoW	RBFT
Wallet/Agent Support	No	Yes

and 8 GiB RAM. For Ethereum, a private Ethereum blockchain was used to deploy smart contracts, perform transactions and run tests. Ethereum has the functionality of smart contract execution on an Ethereum Virtual Machine (EVM). The smart contract was developed in Solidity v0.5.16. For Hyperledger Indy, a test network of Indy Pool Nodes was deployed that has three *TrustAnchor* that act as *issuer* to issue claims and act as *verifier* to verify each other's claims. There is a *holder* that gets VCs issued from an *issuer* and tries to verify it at a *verifier*. An IPFS node written in Go v1.12.0 was installed on the system. The performance of these two separate platforms is now discussed. One thing to note here is that while Ethereum is mostly a permissionless public blockchain network, Indy is a permissioned public blockchain network.

Ethereum: Three parameters are compited to understand the performance of the framework namely, *GasCost*, *ConfirmationTime* and *StorageCost*. *GasCost* refers to the fee required to successfully execute transactions and smart contracts on the EVM and is calculated as shown in equation 4.1. It's unit is denoted as *Gwei*, equation 4.2. The gas is used to allocate resources of the EVM so that smart contracts can be executed in a secure manner. *GasPrice* is the maximum price the user is willing to pay in the network. The *gas cost* is calculated with an average *gas price* of 15 *Gwei*. *ConfirmationTime* shows the time required for a transaction or smart contract to be mined in the network, only after which it will be valid in the network. *StorageCost* will represent overhead incurred on EVM for contract storage and IPFS for DID Doc storage. Table 4.2 and Table 6.2 show the results.

$$GasCost(Gwei) = GasUsed * GasPrice \tag{4.1}$$

$$1Ether = 10^9 Gwei \tag{4.2}$$

Table 4.2 A table showing results of simulation for the parameters: Gas Used, Gas Cost and Confirmation Time

Txn/Contract Deploy	Gas Used	Gas Cost(Gwei)	Time(sec)
Identity Creation	0	0	0
DID Doc creation	32918	493770	31
EthereumDIDRegistry	1923852	28857780	172
VerifiableClaimsRegistry	410148	6152220	121
DID Registration	42800	642000	31
Issue Claim	44765	671475	31

Table 4.3 A table showing results of simulation for the parameter: Storage Overhead for Smart Contracts and DIDDocs

Txn/Smart Contract	Storage Cost (KB)	Stored At
EthereumDIDRegistry Contract	950	EVM
VerifiableClaimsRegistry Contract	350	EVM
DID Doc for each subject (approx.)	10-20	IPFS

Table 4.4 Framework Performance on Hyperledger Indy

Transaction	Time(sec)
Network Node Pool Setup	1.983
Steward Setup	2.607
Create wallet and DID for OEM	2.668
Create wallet and DID for Organization 1, $O1$	3.086
Create wallet and DID for Organization 2, $O2$	3.0197
Creating wallet and DID for $D1$	3.015
Creating wallet and DID for $D1$	3.392
OEM creates 2 schema definition on VDR	6.085
O1 to D1 Credential Issue	0.299
O2 to D2 Credential Issue	0.252
Time taken by D2 to verify D1 credential	0.2039
Time taken by D1 to verify D2 credential	0.331

Note that Identity creation has no associated cost and takes no time as it only involves creating an EOA account on the blockchain and the keypair. These identities are unique and lightweight. DID Doc creation is a relatively less computationally expensive task. The contract deployment takes the maximum computational resource on EVM and hence has greater *Gascost* and respective confirmation time. However, these are one-time costs and therefore practically feasible. DID registration and issuing claims are also less resource and time-consuming transactions in the network. As far as storage cost is concerned, the overhead is relatively inexpensive, particularly DID Doc storage on IPFS is realistic as indicated by data.

Hyperledger Indy: For the Indy setup, a pool of 4 nodes is deployed in the network that form the underlying VDR. They store the transactions related to DID creation and VC issued by *TrustAnchors* and later enable verification of DIDs and VCs. First, a *Steward* is created in the network, these entities have the authority to write transactions to the ledger. Later, the steward on-boards three *TrustAnchors*, an OEM manufacturer and two organizations participating in business. These organizations may share an IoT device or the devices of two organizations may interact with each other over M2M communication. Then two IoT devices are on-boarded and issued with DIDs and VCs. Device $D1$ was issued a credential with 5 claims and $D2$ with 7 claims. When these devices try to communicate with each other, they exchange a verifiable presentation of these credentials to authenticate each other as explained in section 4.3.3. The time taken for each of these transactions is shown in Table 4.4. It's important to note that while one-time transactions like network setup, *Steward* setup, wallet and DID creation for *TrustAnchors* take practically feasible time, the time taken to verify VCs is practically negligible, as it only involves verifier querying the ledger for $subject's$ entities.

4.4.2 Discussion and Analysis

After performing the above discussed simulation, there are a few key takeaways from this study. In general using DID and VC for identity management in IoT devices can offer several benefits. It provides separation of concern between the issuer, holder and verifier of the credentials. This helps in decoupling the dependability of the verifier on issuer for verification and vice-versa and hence reduces the perimeter of harm in case of a security breach. Using DIDs, devices can be issued credentials that can be verified with public keys controlled though the DID, called VCs. Additionally, several advanced features of VCs like selective disclosure and zero-knowledge based proofs can help achieve privacy without compromising security.

A decentralized identity infrastructure can be designed with or without blockchain. There are some key advantages of using a blockchain based distributed public key infras-

tructure (DPKI). These benefits include tamper-resistance, user control and privacy, better interoperability and elimination of intermediaries. A blockchain based DPKI strengthens the principle of data minimization and provides granular access control using VCs. The choice of the blockchain platform will depend on requirements of any project as each of them offer various security features. For example the two blockchain platforms used in this chapter i.e. Ethereum and Indy are both renowned projects in the community. However, Indy has a modular architecture which allows for quick adoption. It offers various developments tools and boilerplate code to tweak and adapt the platform to specific user needs. Hyperledger Indy is also actively being developed to be interoperable with other identity based blockchain projects. Hence, interoperability of also a key selection factor for large enterprise projects and Indy supports it.

4.5 Security of the proposed framework

The security of our identity framework relies on some underlying practical assumptions: (a) the channel used for message transmission is secure, and (b) the underlying permissioned blockchain network is secure. Blockchains provide security advantages to a variety of applications by removing/reducing the need for trusted third parties. Table 4.5 further describes how various malicious entities can affect the network accompanied by corresponding prevention.

Table 4.5 A table showing Malicious Entities, Threats and Inbuilt Prevention for blockhain peer and IPFS node

Malicious Entity	**Explanation**	**Inbuilt Prevention**
Blockchain Peer	• Modify/Delete Subject data • Approve a false transaction	• Consensus Algorithm • Voting policy
IPFS Node	Tampering of uploaded subject's DID docs	Hash of DID document is maintained in registry

In the past, DDoS attacks on IoT devices have been prevalent among hackers. This was possible due to weak passwords and theft of device certificates from certificate servers. The proposed framework is resistant to such an attack as device identity is stored on its *wallet* and is not stored anyplace centrally. This work has no centralized *subject* identity storage that can jeopardize devices. *Identity Theft* is motivated by the fact that identity stores exist where substantial device identifiers are stored. Malicious actors have a considerable incentive to hack such servers. In the proposed work, the lack of a central device identity

server makes it less appealing for malicious actors to resort to a single device identity theft. Moreover, the DID alone is a pseudo identifier, the control of its corresponding private key proves its legitimacy by a device. *Social Engineering* attacks are possible in IoT scenarios when personally identifiable information (PII) is disclosed by devices in their communication. The proposed work is resilient to such attacks as no PII is disclosed in either DID Doc or VC. *Ransomware* attacks are targeted on IoT devices to gain control of device functioning and ask for some benefit in return for giving device control back to the legitimate owner. As shown in our use case a device first verifies the identity of the *subject* and only if verified through the VC, allows it to access device resources. Hence, the proposed framework is safe from ransomware attacks. *Private Data Leak* remains a constant challenge for IdMS. This can be minimized by minimal disclosure mechanisms like zero-knowledge protocols. *Metadata Tracking* by applying pattern analysis of on-chain metadata through possible interception of parties may be applied. This correlation can be minimized by decoupling devices from persistent identifiers and they may use session pseudonymous identifiers. *Oracles and top-layer protocol compromises* are possible when blockchain IdMS are integrated with off-chain data, logic and processes in the form of oracles. As researchers say a system is as secure as its weakest link. An overall check and balance should be in place when designing such systems. Detailed security analysis is beyond the scope of this chapter, and will be part of the extended work.

4.6 Conclusion

This work discusses the issues faced by the current IoT identity ecosystem due to a large number of deployed devices and the devices trying to communicate asynchronously and heterogeneously with each other to achieve autonomous, faster communication. The centralization and use of PII in digital certificates and its drawbacks are analysed in this work. This chapter proposed a novel distributed-ledger-based M2M digital identity framework for implementing autonomous remote device-to-device communication. With the new forms of digital identifiers centralized identity databases can be eliminated and interoperability will improve promoting cross-silo communication between devices for innovative use cases. Experiments conducted demonstrate the feasibility of such an architecture that deploys lightweight smart contracts and uses distributed storage services. The device meta-data on IPFS can be further encrypted to provide better security.

Chapter 5

A Fair, Secure and Trusted Decentralized IoT Data Marketplace enabled by Blockchain

This chapter constitutes the third contribution of this dissertation. The key information in this chapter is based on the work published in Publication III. In the previous two chapters the notion of decentralized identity for IIoT is proposed, designed and implemented as stand alone solutions using two different approaches: smart-contract based decentralized identity and self-sovereign identity. The various aspects of decentralized identity are discussed including how it can be leveraged to design a secure, interoperable and scalable decentralized identity management infrastructure with fewer points of failure and less reliance on a centralized ID controller. In this chapter, the concept of decentralized identity has been adapted to apply to a real-life challenge such as an IIoT data marketplace. The objective of this piece of work is to apply the concept of decentralized identity to an "IIoT data marketplace" to ensure fairness, security and a trusted setup. In this chapter, design for a fair, secure and trusted data marketplace is proposed in which data buyers and sellers interact on a blockchain-based, smart-contract-enabled framework. The idea of *identity* is central to this work, as it is used to derive trust and reputation in the network. Therefore, this chapter explores the concept of decentralized identity using a proof-of-concept of application.

5.1 Background and Motivation

Sensing-as-a-Service has emerged as a rapidly growing industry that has spurred the interest of firms and enterprises that invested heavily in the installation of IoT infrastructure for the past few decades. They have realized that the data being generated in their siloed

environment is a tradeable commodity with resale value and it can provide an additional monetary benefit. On the other hand, there are companies that are not inclined to invest in setting up IoT infrastructure but are willing to pay for reliable data streams. Occasionally, these companies require data from heterogeneous domains like weather, population density, soil type etc. In such a scenario it is more profitable to buy this data from corporations who already have such sensors installed. These use cases demand a platform which makes it possible to sell and purchase data seamlessly. However, wide-scale adoption of this idea still remains a challenge due to uncertainties around trust, cross-domain trading facility, fair transactions and security. Vendors coming together to exchange data as a commodity are looking for assurance on the validity of IoT devices and enterprises engaged in business. Siloed identity verticals, unauthorized malicious entities posing as data producers, centralized control of the platform and fear around equitable payment settlement are challenges that require a new approach to make IoT data trading a reality. It is essential to ensure that, unlike any marketplace, the platform is not controlled by a powerful organization.

In this work, a model is presented that is based on the *principle of data sovereignty* in which independent IoT business ventures can sell and purchase data reliably and in a trustworthy manner with undisputed compensation to all the parties. Such an effort requires a platform that disintermediates the central storage operator. To achieve transparency, accountability and fairness it is necessary to forgo centralized control from a trusted third party (TTP) in terms of fair listing and seller discovery, identity validation and payment settlement. Therefore, the proposed digital marketplace uses a blockchain platform as a decentralized trusted party that facilitates fair listing of data streams, identity verification and payment settlement using smart contracts. According to GDPR Articles 16 & 17 (Right to Rectification and Erasure), data modification/erasure must be possible to comply with legal requirements. Hence, critical data should not be stored on an immutable storage such as blockchain, which does not allow for modification of data, to ensure data integrity and trust. Therefore, in the proposed framework no private data is stored on the blockchain. Moreover, blockchain is not a viable option for bulk data storage. Hence, the proposed platform leverages a decentralized data storage layer which is replicated across participating nodes to provide data delivery in a fault-tolerant manner. The authentication of actors is executed using Self-Sovereign Identity (SSI) concept, wherein each entity holds a unique Decentralized Identifier (DID) [125] to authenticate on the platform.

The key idea of this work is to present the feasibility of a decentralized IoT data marketplace that employs a system of peers to collectively provide services ranging from data listing to data delivery. The proposed digital marketplace uses blockchain as a decentralized trusted party that facilitates fair listing of data streams giving preference to trusted sources and settlement of compensation using automated smart contracts.

Several blockchain based solutions are present in the market today for various industry verticals such as finance [126], healthcare [127], energy [128], automotive [129] etc. While data sharing marketplace has been proposed by startups [130], [11], significant contributions have been made in the trust and fairness guarantees in the proposed platform. It is estimated that the value of the IoT data marketplace will reach 3.6 trillion USD by the year 2030 [130]. Eliminating data sharing friction will serve industries like healthcare, smart city and advanced mobility infrastructure, agriculture, smart-grid and electricity, research etc and promote interoperability.

5.2 Related Work and State-of-the-Art

Digital Marketplaces: A range of tangible and non-tangible goods are part of digital marketplaces today including commodities, e-books, music, images and videos. These marketplaces termed eCommerce are not bound by physical boundaries and provide ease and competitive rates to consumers [131]. Such platforms have encouraged trading ecosystems like Software-as-a-Service (SaaS) and Infrastructure-as-a-Service (IaaS) [132]. However, since most of these digital platforms lack in listing and pricing transparency, the seller has minimum to no right to determine the terms of the licensing agreement. Moreover, these platforms are owned and regulated by a centralized authority which dominates the decision-making process [133].

Peer-to-peer (p2p) data sharing network: P2P exchange of information has emerged as a growing fraction of the world's digital economy [134]. This form of value exchange has expanded dramatically to include services, information trading and P2P lending. However, such networks suffer from regulatory challenges and building a marketplace using these platforms alone is challenging. Platforms such as BitTorrent [135] scale well yet building a marketplace on them is infeasible due to a lack of properties like fairness, trust guarantees and payment settlements among network users.

Decentralized and Distributed Storage: As opposed to the centralized cloud servers, owned and regulated by a single organization, decentralized storage solutions consist of a P2P network, hosting and sharing data in a resilient and fault-tolerant manner. Such a form of storage has garnered attention both from the research community and commercial organizations due to the advantages such as fault tolerance, collusion resistance and attack resistance. Decentralized storage solutions like IPFS [136] have inspired projects like Filecoin [137] and OrbitDB [138]. IPFS is a content addressable storage which prevents duplication and provides high availability of content. Storage platforms like IPFS when used along with a technology like blockchain can help in creating holistic solutions like Storj [139], Filecoin [137], Sia [140]. Although these solutions have an incentivization

mechanism for participants, they lack business logic, trust and fairness guarantees that are pivotal for a digital marketplace.

Decentralized IoT Marketplaces: The state-of-the-art marketplaces are based on centralized cloud infrastructure owned by a single organization [141], [142]. Due to research interest around decentralized marketplaces, there have been attempts to design marketplaces for data sharing [143], [144], [145]. However, IoT data marketplaces remain a relatively untapped ecosystem. Xu et al. [146] explore the use case of blockchain marketplace in the electricity and smart grid domain. They present a theoretical business model archetype for the energy market in accordance with EU's Smart Grid Architecture Model (SGAM). However, their work lacks a concrete architecture design involving incentive mechanisms and security features. Missier et al [147] propose a decentralized infrastructure for trading IoT data between producers and consumers exchanging a data unit "cube" with the help of trusted edge gateways. They analyse the feasibility using cost for "cube" settlement. However, it is difficult to establish the scalability and trust metric of the system. Ramchandaran et. al [148] present a simple smart contract-based implementation of a decentralized registry for IoT data trading for smart cities. However, their work largely overlooks the analysis and illustration of essential elements of a data marketplace like payment schemes, trust and privacy. Niavis et. al [149] present a decentralized data-sharing infrastructure for off-grid networking using a blockchain network and a distributed file system. While their work takes into consideration a great deal of factors like identity management and private data exchange, elements like payment settlement, trust among trading entities and fairness remain unexplored. Drawing motivation and understanding from the limitations of discussed works, an attempt is made to design a fair, transparent, reliable and trust-less system for IoT data trading in a decentralized manner.

5.3 Case Study

Sensing-as-a-Service is about creating new revenue streams from existing assets in the IoT world and many commercial ventures are already investing [130] [11] [129] due to its promising future prospects. It is estimated that the value of the IoT data marketplace will reach 3.6 trillion USD by the year 2030 [130]. Eliminating data-sharing friction will serve industries like healthcare, smart cities and advanced mobility infrastructure, agriculture, smart grid and electricity, research etc. and promote interoperability. FlashLabs Inc. [150], a partner of Hyundai group is developing autonomous vehicular technology and is exploring traffic patterns, road markings, environmental parameters, charging stations, and geographical data for an area. However, it does not want to invest in installing such heterogeneous sensing infrastructure. In such a case, it can turn to government agencies,

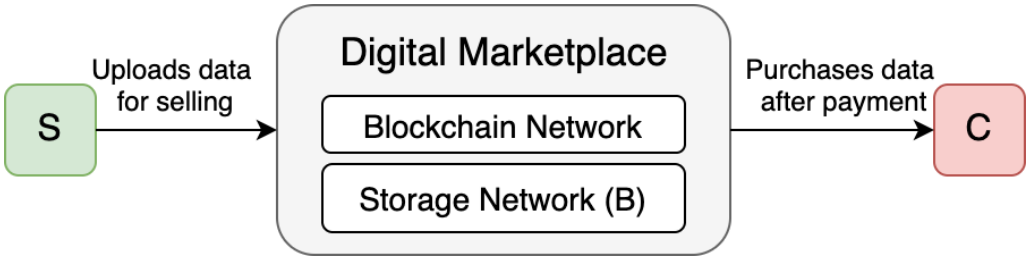


Fig. 5.1 Showing the high-level view of sub-components of the proposed digital marketplace

organizations that installed smart electricity poles, and open data collection agencies for all its myriad data needs. There are numerous such use cases that form the motivation for developing a marketplace for IoT data trading. While there are certain solutions focusing on centralized architectures using cloud infrastructure [151], this work attempts to design and analyse decentralized solutions for IoT data marketplaces.

5.3.1 Contributions

The key contributions of this work are as follows:

- A novel proof-of-concept in presented as a decentralized solution to the IoT data marketplace that supports actor verification, managed trust metric and a secure and transparent transactional model for paying the sellers all in a decentralized manner by leveraging an innovative combination of blockchain, DIDs, p2p data stream storage and end-to-end encryption.
- The proposal guarantee (i) fair settlement of compensation to all the involved parties, (ii) unbiased content listing using calculated trust metric, (iii) security of the content from peers hosting and delivering data streams (iv) fault tolerance and assured data delivery in case of faulty peers.
- A proof-of-concept implementation is developed with distributed applications (DApp) and blockchain layer using Hyperledger Fabric, and the decentralized real-time stream data storage layer using Apache Kafka. The performance of the system is discussed and it demonstrate that the system scales well with rational overheads.

5.4 System Elements and Design Features

5.4.1 System Actors

The actors of the system as shown in Figure 5.1 are discussed with their respective roles in detail below:

1. **Seller** (*S*) wants to use the marketplace to trade the IoT data generated in its facility. It may not be able to host the data itself and create an infrastructure around it. Hence, it uses the service of the digital marketplace to monetize its otherwise siloed machine data. *S* wants the buyers to discover its offers and pay fairly for the data. *S* is willing to pay the broker nodes in exchange for their data hosting and processing service.

2. **Broker** (*B*) are data facilitators of the system. They are a cluster of decentralized p2p nodes that host data in the form of data streams and are incentivised for their service. The independent server providers with a stake in the system can choose to act as broker nodes and provide storage services in the system at competitive rates. These nodes are modelled in publish-subscribe topology. Brokers facilitate hosting and managing important security infrastructure that is required to oversee such a service. Each seller in the marketplace may not have the potential to manage such an infrastructure. They can also perform services like authentication and authorization to allow access to their data stores on behalf of sellers.

3. **Client** (*C*) comes to the marketplace platform to browse, purchase and get access to a myriad range of on-demand data sets which otherwise is difficult for them to procure in a trusted manner.

5.4.2 Design Features

In the proposed model, the broker nodes are synchronized with each other to serve data under various topics. Each topic is replicated to multiple partitions to avoid centralization and ensure fault tolerance and scalability. Given the above actors in the system, several questions arise with regard to how the funds will flow in the system, the payment scheme and intervals of payment. While there are many variants possible, two broad variants are discussed here:

- **Batch Mode:** In this mode of trading, the client wants to access IoT data in batches at regular intervals for a stipulated period of time. For example: once every fortnight, for a period of a few months. The agreement in this mode involves payment of a flat fee periodically before delivery of each batch of data. This mode is called “batch” mode as data is not served in real-time but rather as small packets of information.

- **Stream mode:** This mode is meant to serve data as real-time streams subscribed by the client. The client pays the seller and broker periodically for the subscribed streams. This payment is made in advance by the client before the streaming of data can be initiated. Stream mode is an important mode when a client wants to build a real-time application by consuming the data generated continuously.

In both the above-discussed modes, it is expected that the seller will upload the data on the broker network in real-time. The proposed solution works with both variants. In both cases, a smart contract ensures fair and smooth data trading. It achieves a reliable trading flow in which the seller and broker are paid for data and service respectively whereas the client is assured of data delivery without losing any deposited payment. In case of failure to deliver data, the smart contract refunds the remaining funds back to the client. All the data exchanges happen in a publish-subscribe data model. Such a model is well suited to this design pattern, as a large number of sellers and clients will access the data streams simultaneously.

Identity: Identity verification plays an important role in the design. It is assumed that the participating entities in the network are adopters of decentralized identity in the format of the Self-Sovereign Identity (SSI) framework. Each participating entity holds Decentralized Identifiers (DIDs) [125], short identifiers resolvable to a DID Document containing more information about the holder. It is assumed that the actors be they individuals, organizations, agencies or research groups, interested in trading on the marketplace have their identities verified on one of the decentralized platforms. There are several such examples of online digital identity initiatives like Microsoft ION, IBM Trusted Identity, Evernym, Sovrin [152] etc. It is similar to an eKYC process done by an organization to onboard customers. This ensures that each seller on the platform is a verified organization/agency/facility. An identity verification step is mandatory for the seller in the proposed work. However, a client can opt to undergo DID verification. If a client chooses to undergo the DID verification, it will have a wider range of data streams to purchase as some sellers might want to give access to their data to legitimate clients only.

Data Security: Security by design is a vital aspect of any secure system. In this work, it is ensured that data originating from a seller is encrypted before it is outsourced to a broker and later decryption keys are only available to a client who has made upfront payment. No middle party should be able to decrypt it in transit or at rest. For this reason, an end-to-end encryption (E2EE) is applied on IoT data in this design. All the communications are TLS/SSL encrypted.

Trust: Trust is emerging as a critical influencer of consumer behaviour on digital platforms such as marketplaces [153]. It infuses network effects into the system, which further drives the participating parties to act fair. The marketplace maintains a TRUST SCORE ($\mathbb{TS}$) metric for every actor in the network. $\mathbb{TS}$ is derived from two primary factors, reputation and credibility. No centralized authority is involved in trust score calculation.

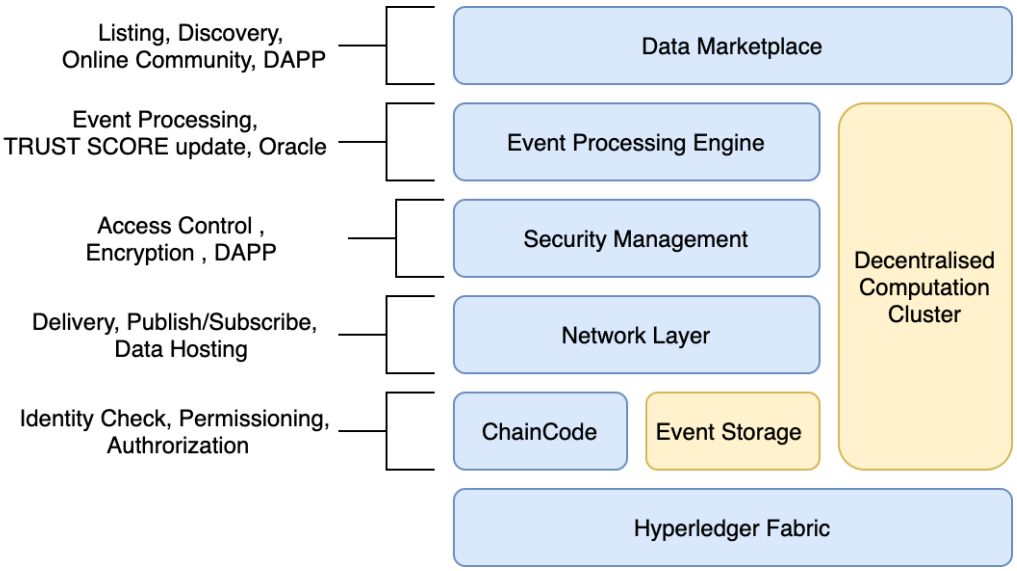


Fig. 5.2 Showing the architectural layer diagram outlining the various layers of the framework and their respective roles

5.5 System Components and Roles

In this section, the role of each layer of the proposed architecture is broadly discussed. Figure 5.2 shows the layers discussed below:

5.5.1 Data Marketplace

The data marketplace is a listing portal implemented as a DApp. A DApp is an application which runs on a decentralized peer-to-peer network like the blockchain. It is a decentralized portal/website that interacts with the blockchain back-end through smart contracts. Data Marketplace DApp enables listing and discovery of data offers (DO) from the sellers. Data is offered as data streams and they can be subscribed in the following modes 1) Batch Mode 2) Stream Mode. A data offer as shown in equation 5.1, consists of the data topic, payment mode (batch/stream), price for the data topic in selected mode and related terms of data purchase. Data is offered in a publish-subscribe data model as it is suited for the IoT domain since both the sellers and buyers are potentially large and are constantly changing.

$$Data\_Offer = Topic + PaymentMode + Price + Terms \tag{5.1}$$

In the marketplace, the listing will be available from potential sellers accompanied by their respective $\mathbb{TS}$ and payment offers for modes of subscriptions.

5.5.2 Event Processing Engine

Events are an important component of any software architecture as they give useful insights into the process state. Events are generated from various sources e.g. at the origin of data streams, from DApps, from client applications, execution of Smart Contract functions etc. Events are recorded in a separate "event storage" location on the broker nodes cluster. Since this storage is hosted on the broker nodes, it provides a distributed service to read/write events on the marketplace. Engine listens to the events on the network, processes them for consumption by DApps/Smart Contracts and takes real-time actions on them. It also registers events related to $\mathbb{TS}$ metric, processes them and updates the $\mathbb{TS}$ for participating entities in the marketplace.

5.5.3 Security Manager

This layer in the system ensures that data exchange takes place in a secure manner. It envelops the broker nodes and monitors all the communication made by an entity with them. The seller encrypts the data stream and uploads it to brokers through this DApp. Later, when a client wants to download the data, it first verifies its identity with the smart contract, makes payment and obtains an access token. This layer verifies the token with the blockchain and allows the client to access the data stream for the allotted time frame.

5.5.4 Network Layer

It is the primary data transport layer of the architecture. It is composed of a collection of a p2p network of broker nodes that facilitate data streams and decentralized management of events. These nodes are compensated for their service in the network. In order to handle the decentralized messaging taking place in the system, this layer uses policies defined in the smart contract for its functioning. Policies and permissions defined in the smart contract are the guiding principles for this layer. Consensus and coordination among the nodes are achieved using the underlying blockchain ledger.

- *Data Streams:* Data streams are hosted by verified sellers on the broker nodes. The seller encrypts the data using the end-to-end encryption (E2EE) technique before uploading it to brokers to ensure the security and privacy of the data. A single stream is replicated across multiple brokers to ensure fault tolerance. When a stream is hosted by a seller, it makes a data offer on the DApp that is registered on a smart contract against the seller's public key. When a client wants to access a stream, it enters into an agreement with the seller, the terms of which are detailed in the form of a license. The fields of the data stream are *streamID, streamName, topic/type, owner/seller, permissions, conditions*.

- *Broker Nodes:* Brokers are the fundamental enablers of the proposed decentralized architecture. They host data streams, deliver encrypted data to clients, forward event streams to the processing layer and communicate with the security manager DApp to securely process the authorization of clients requesting data stream access. Broker nodes will expose all these functionalities via APIs. Broker nodes provide storage and computation services in lieu of monetary benefits in the proposed architecture.

5.5.5 ChainCode

ChainCode or "Smart Contracts" are self-executing computer programs that enforce and govern the terms of agreement in a blockchain. In the proposed data marketplace architecture, smart contracts support autonomous operation. The network uses smart contracts for coordination between sellers and buyers, identity verification (using SSI), payments, incentivization and permissioning. These contracts generate events to alert the system on the completion of transactions.

5.6 Problem Definition

The problem addressed in this work is to ensure fair, secure, and trusted delivery of IoT data from a seller to a client in a decentralized manner. Let Π be a protocol executed by a cluster of broker nodes $\mathbb{B}$, with each node $B_i \in \mathbb{B}$ hosting data for multiple topics $\mathbb{T}$ and a client downloading a batch d of data stream of a topic $T_i \in \mathbb{T}$ and paying price p.
Fair Trade: Protocol Π is fair if the following hold:

1. **Client Fairness:** If C pays p according to Π, it is assured of receiving the subscribed batch d of data stream of topic T_i for the agreed mode and time period before termination of Π.

2. **Seller Fairness:** If C pays p according to Π to purchase a batch d of data stream of topic T_i published by P, then P is assured of getting their share of payments.

3. **Broker Fairness:** For all $i \in |\mathbb{S}|$, if S_i gives access to d of T_i according to Π, it is assured to receive payment.

Privacy: Protocol Π is considered secure if the execution of Π does not disclose the data on the marketplace to any broker. The access of data should be given only to a client who has purchased it by making payment p. It is important to note that without a privacy guarantee, fairness cannot be assured, as any broker that gains access to data without prior payment is a violation of the Π.

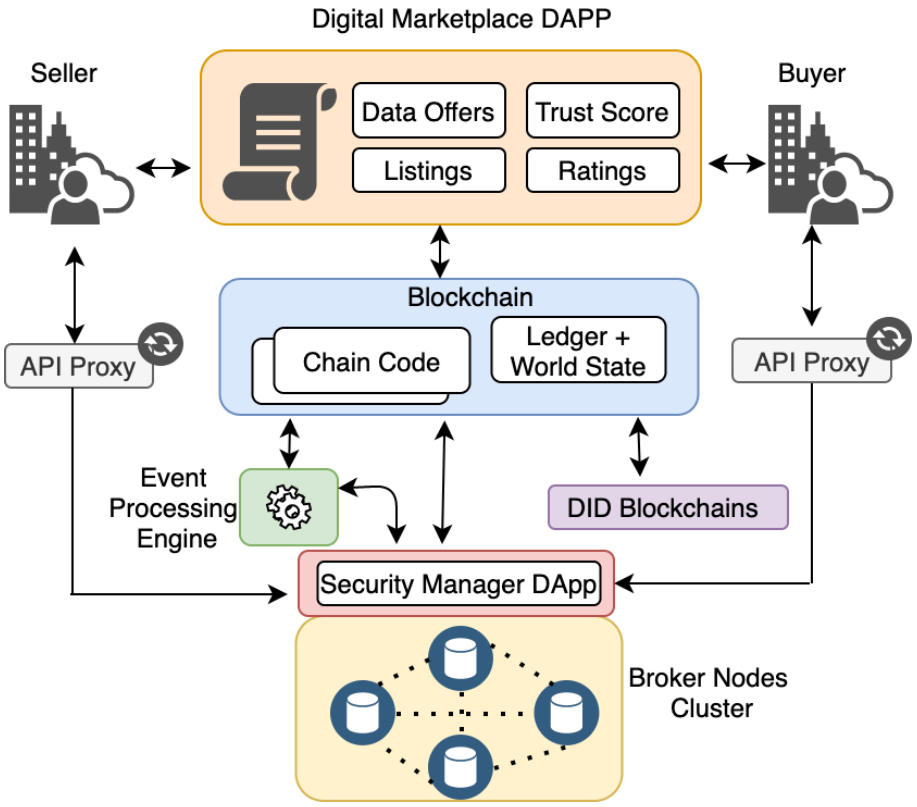


Fig. 5.3 A detailed diagram illustrating all the system components of the architecture

Trusted Exchange: Protocol Π is considered secure if it allows only verified sellers to host data on the platform. After verification, an honest seller should be rewarded with a higher Trust Score ($\mathbb{TS}$) based on transaction history and feedback from trading parties. Sellers can place restrictions on certain data streams to be available only for verified clients.

5.7 Solution

In this section, the proposed protocol flow is described in detail. The system features and provided guarantees are also analysed.

5.7.1 System Architecture Overview

A data marketplace with no central authority can encounter critical challenges like ensuring fair trade settlement, privacy and security guarantees [154]. A decentralized blockchain network of consensus-abiding peer nodes can supplant the trusted third party, enforce equitable trading rules and manage autonomous payments. In addition, it ensures transparency, non-repudiability and shared participation in the network. However, using blockchain for storing IoT data is not feasible and would impede performance and scalability. Therefore,

a data storage layer is needed to host data and supplement the blockchain layer in the marketplace. The two layers can be managed independently by different entities as long as the broker nodes in the storage layer can query a peer in the blockchain layer.

The storage layer lacks intelligence and needs additional services to monitor security and privacy policies. The *Security Manager* DApp regulates the authorized access to stored data on this layer. It allows seller S and client C to interact with the storage layer only after authorization. Seller S and client C will communicate with the blockchain through Data Marketplace DApp to invoke the ledger to make queries and later transactions.

The storage layer cannot be trusted with unencrypted data as it will violate data storage compliance like GDPR, HIPAA etc. Even though a decentralized network may be collectively honest, any individual broker node acts maliciously. Therefore, before uploading data to a broker node, the seller S encrypts the data.

The progress in the blockchain ecosystem offers a wide range of blockchain platforms with various functionalities. The use of blockchain in the proposed architecture remains very generic and most systems with support for smart contracts can be used. However, a permissioned blockchain with robust support for identity and access control is better suited for the proposed work. In this work, Hyperledger Fabric is used as the blockchain platform. Figure 5.3 shows a detailed architectural diagram with all the system components.

5.7.2 Notations and Preliminaries

The blockchain ledger hosting smart contracts is denoted as $\mathbb{L}$. All the transacting parties will register with $\mathbb{L}$ before any trading takes place on the platform. $KeyGen\ (1^{\lambda}) \rightarrow (sk, pk)$ is a generator algorithm that generates a private key sk and its corresponding public key pk given a security parameter λ. Each actor will possess a public-secret key pair. Each party will send signed transactions to $\mathbb{L}$ and the peers in the network will verify the signatures before including transactions in $\mathbb{L}$.

Data streams are identified by their respective unique *streamID* which includes *seller's name (S), stream name* and *time-stamp* of origination. Let GenerateID(.) be the procedure that returns a *streamID* given a batch d of data stream,

$$streamID_d \leftarrow GenerateID(S||streamName||timestamp)$$

HMAC-SHA256(k,P) is a keyed-hash message authentication code where k is the secret cryptographic key and P is the plain text for which *MAC*, σ needs to be generated. This cryptographic primitive is used to verify both the data integrity and authenticity of the message as compared to the hash *H(P)* that only guarantees message integrity. The receiver

with the key when receiving data (encrypted or not) along with the corresponding σ, regenerates σ on its side and matches it with the received one to ensure integrity.

$S_{sk}(m) \rightarrow \phi$ is a signature algorithm that generates the signature ϕ of the message m with the private key *sk*, and $V_{pk}(\phi, m) \rightarrow \{0,1\}$ is a verification algorithm that verifies whether the signature ϕ on message *m* is valid or not using the public key *pk* and returns 1 if ϕ is a valid signature or 0, otherwise. Let (Enc, Dec) be a secure symmetric encryption scheme. *AES(k,P)* in *CBC* mode is a widely used encryption scheme which is faster than its asymmetric counterparts. This scheme will be used by the sellers to encrypt the data stream '*d*' before uploading the encrypted data stream '*e*' it on broker nodes. This helps in protecting the privacy of data at rest from a possibly malicious broker node.

5.7.3 Smart Contract Functions

There are two DApps in the architecture through which smart contract functions are invoked. These functions are categorized as follows:

1. Data Marketplace DApp

    - *RegisterAndVerifySeller*(did_S, pk_S): seller *S* sends its DID and public key.
    - *RegisterAndVerifyClient*(did_C, pk_C): client *C* sends its DID and public key.
    - *RequestUploadToken*(*DO*, pk_S): *S* requests authentication token to upload *DO* on broker nodes.
    - *RecordStreamDetails*(*streamID*, *DO* , k_m, k_e, σ_d) : record stream details on blockchain.
    - *RequestDownloadToken*(*p*, *DO*, pk_C): *C* makes payment *p* and requests exchange token.

2. Security Manager DApp

    - *UploadDataStream*(ϕ_S, pk_S, *e*): *S* send encrypted data *e* with signed authentication token.
    - *DownloadDataStream*(ϕ_C, pk_C): *C* sends signed exchange token to receive data from broker nodes.

5.7.4 Protocol Flow

The protocol is divided into three phases. The onboarding phase deals with verifying the identities of the actors in the network using *did* and registering them in the marketplace. The data upload phase will allow the sellers to complete all the necessary steps to get their data up in the market for sale. The data purchase phase will explain the steps involved from the time a client picks a data offer to the settlement of payments. The complete protocol flow is illustrated in Figure 5.4.

1. Onboarding Phase: In this phase, the entity seller/buyer makes an account, verifies its identity and registers itself for trading activities on the blockchain through marketplace DApp. Each entity uses key generator algorithm $KeyGen\ (1^{\lambda}) \rightarrow (sk, pk)$ to obtain a key-set. The generated key set is used to obtain a DID (did). The actor sends the did associated with its identity to the DApp, which is forwarded to the blockchain. The blockchain resolves the did through the underlying decentralized network/blockchain which was initially used to issue it to the entity. If the did is successfully verified, an account is created for the actor and its $\mathbb{TS}$ is incremented on the DApp. Later, the seller registers the data offers DO it wants to make in the marketplace. These offers along with payment details and terms are recorded on the blockchain.

Algorithm 3: Onboarding/Actor

1 **func VerifyandRegister**(did, pk){
2 **Send** did, pk **to Data marketplace DApp**
3 **If blockchain successfully verifies with issuing blockchain, register on** $\mathbb{L}$
4 S **uploads** DO **on Data Marketplace DApp**
5 }

Algorithm 4: Data Upload/Seller (S)

1 **func DataUpload**(d, DO, k_m, $k_e, S, streamName(sN), timestamp(ts)$) {
2 **Generate a unique stream ID for data stream** $streamID_d \leftarrow$ $GenerateID(S||sN||ts)$
3 **Generate** k_m **and create MAC of data stream** $MAC(d, k_m) \rightarrow \sigma_d$
4 **Encrypt the data stream using the encryption key** $Enc(d, k_e) \rightarrow e$
5 **Upload request sent for listed** DO **, receive token** $Upload\ Request\ (DO) \leftrightarrow \omega$
6 S **signs the received token using its secret key** $S_{sk}(\omega) \rightarrow \phi_s$
7 **Verify signed token with Security manager DApp,** $V_{pk}(\phi_s, m) \rightarrow \{0,1\}$ **and upload the data stream** e **to the broker node** B
8 **Record details on blockchain by calling** $\mathbb{L}$.
DataAdded($streamID_d, DO, k_m, k_e, \sigma_d$)
9 }

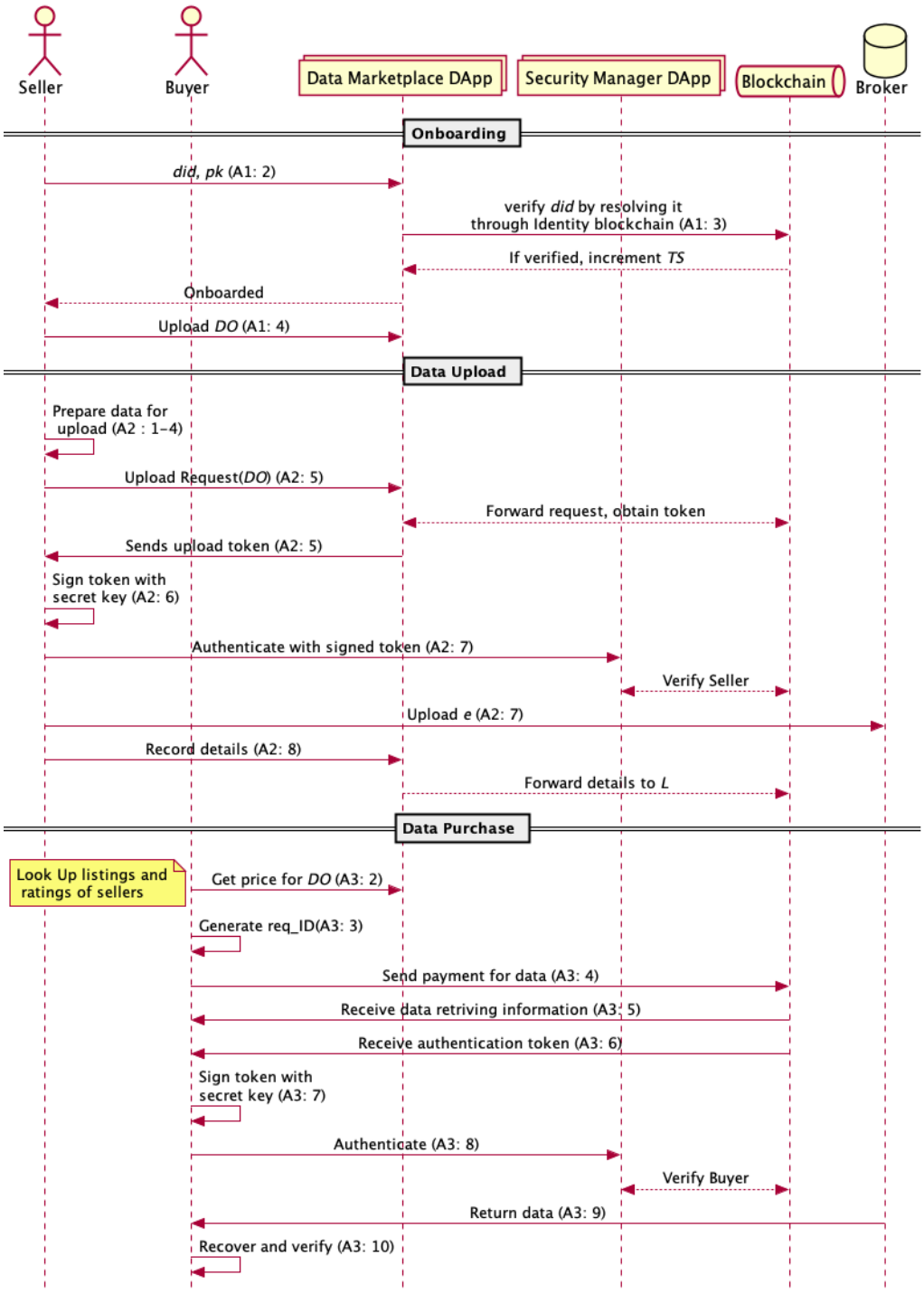


Fig. 5.4 Showing the protocol flow outlining interactions between all the components of the data marketplace

Algorithm 5: Data Purchase/Client (C)

```
1  func DataPurchase(DO, streamID_d) {
2  Retrieve price p for streamID_d by calling 𝕃
3  Generate a request ID, reqID
4  Send the payment p to 𝕃 along with reqID. Payment(streamID_d, reqID, p)
5  Get K_m, K_e, σ_d from 𝕃 for streamID_d.
6  Receive authenticating token δ from 𝕃 for retrieving data from B
7  C signs the received token using it's secret key S_sk(δ)→ φ_c
8  Authenticate at Security Manager DApp through API proxy V_pk(φ_c, ω) →
     {0,1}
9  If verified, receive e from B
10 d' ← Dec(e, k_e) , σ'_d ← MAC(d', k_m). If σ'_d == σ_d , d' == d, data verified.
11 }
```

2. Data Upload Phase: The seller S wants to monetize a batch d of IoT data and needs to upload the data to a broker node B. First, it generates $streamID_d$ which uniquely identifies d. Then, S generates a key K_m for obtaining the message authentication code of the data, $MAC(d, k_m) \rightarrow \sigma_d$, so that any potential client can verify the integrity of purchased data. Later, S generates a secret key K_e and encrypts data batch $Enc(d, k_e) \rightarrow e$. Once the data is ready, it initiates data upload by sending a request to the marketplace DApp. The DApp sends the request to the blockchain which responds by sending a token ω to S. The token is used to uniquely identify an upload request by the S, therefore, it signs ω with its secret key sk to generate ϕ_S. Using this signed token, S verifies with the *Security Manager* DApp through *API Proxy* layer, a service to help clients interact securely with the storage layer. The encrypted data e along with related metadata is stored on B, and secret keys, σ_d, $streamID_d$ and corresponding DO are recorded on the ledger.

3. Data Purchase Phase: In this phase, C browses through the listing of available DO on the DApp. It selects a batch d of data with unique $streamID_d$ on the marketplace. It generates a request identifier $reqID$ and sends it to the blockchain with the payment p for the requested data stream through marketplace DApp. The ledger will first verify if the owner S has placed any restriction on the client it wants to sell data to, for example, the client C needs to be a verified client. If the terms specified in the DO are met and payment p is received successfully, the process to establish data transfer is initiated. C receives two secret keys from the ledger, encrypted by its pk, one for verifying data integrity and the other for decrypting the data. Later, the blockchain sends an exchange token δ to C. The δ is signed by C using its sk to generate signature ϕ_C. C approaches the *Security Manager* DApp through *API Proxy* service with ϕ_C. Upon successful authentication, it receives the data. It decrypts the data, and creates a MAC from recovered data, if the generated MAC matches with the received one, the client can be assured of the integrity of the data.

At every stage of the above-described phases, event generation takes place in the network. These events are continually sent to the *Event processing Engine* for processing. They will further help in settling payments and managing a common view of DApps for all the parties in the system. Another point to note is that after a client successfully completes a transaction, it is prompted for submission of rating for S and B. This rating will be later used to calculate $\mathbb{TS}$.

Payment: In section 5.6, the term fairness from a seller, client and broker perspective is defined. In the proposed design, the client can fetch the requested data stream only after making payment to the blockchain. Since, the interaction between the client and a broker contains an intermediary layer, *Security Manager* DApp, ascertaining that a broker has served the data is easy. This layer keeps a check on the malicious brokers who otherwise can claim to have served the data without doing so to maximize profit. Similarly, in our design, it is difficult for a malicious client to claim that they have not received the data, while they have already downloaded it.

5.7.5 Design Analysis

In this section, it is demonstrated that the proposed design satisfies the goals outlined in section 5.6. The assumed model allows adversaries to keep a check on malicious parties together. It is to be noted that since a broker B is randomly assigned to S, collusion among the client C and broker B_i is unlikely. Also, there is no direct interaction between S and C in the proposed protocol, hence their collusion cases reduce to individual malicious behaviour rather than a group phenomenon.

Fair Trade

Client Fairness: The proposed protocol Π ensures that an honest client who has made payment p to $\mathbb{L}$ should receive access to data stream d for agreed time frame. Specifically, it should be able to access the data stream d and its respective decryption and integrity-check keys. Under the assumption that blockchain is tamper-proof and broker nodes do not have direct access to unencrypted data, honest broker nodes will follow the protocol and keep serving the data stream. Since, the execution of smart contracts is autonomous, tamper-resistant and unbiased, therefore, malicious parties cannot affect the key release by $\mathbb{L}$ after a successful payment made by C. Thus, Π guarantees client fairness.

Seller Fairness: For every successful data exchange, the respective payment is made to $\mathbb{L}$. As the smart contract holds logic to directly pay S, execution of Π guarantees seller fairness under the assumption that smart contract execution cannot be tampered with.

Broker Fairness: In the proposed protocol, each broker B_i when delivers a data stream d for an agreed time frame, an event is generated and it receives the payment from $\mathbb{L}$ corresponding to it. Under the assumption that the smart contract execution is tamper-

resistant, the payment to all broker nodes is guaranteed as it is controlled by $\mathbb{L}$, only when they served C. Thus, the protocol Π guarantees broker fairness.

Privacy: According to Π, no broker can look up the data stored on broker p2p network. This is due to two reasons, first, due to end-to-end encryption scheme nodes being oblivious to the content of the data, and secondly, because keys are not available to any broker node. Therefore, a disconnect is maintained between the data and key set, which can only be simultaneously obtained by a client C who has made an upfront payment to the smart contract. Under the assumption that a smart contract is fairly deployed, an adversary will not be able to obtain the data. In this way, the privacy and security of the data are maintained by Π.

Trusted Exchange: Trust management in a decentralized setting with no central authority is a critical and at the same time a challenging design problem. When multiple unknown parties transact with each other, there should be a metric to ascertain their honesty and integrity. In the proposed system, TRUST SCORE ($\mathbb{TS}$), a metric associated with each seller and client on the platform to assess their trustworthiness is used.

The trust metric of PeerTrust's in [155] is modified to suit the proposed platform's functionality. Reputation and credibility values are used to compute the trust values. In this work, reputation is derived from the verification status of the entity's identity which is denoted by decentralized identifiers (DIDs). Credibility represents the confidence an entity has garnered from its transacting parties. Its value is derived from ratings and feedback an entity received after the completion of each transaction in the past. Given a recent time window, let $I(u,v)$ denote the total number of transactions performed by an actor u with v and I(u) denote the total number of transactions performed by actor u with all other actors, $a(u,i)$ denote the other participating actor in actor $u's$ ith transaction, $S(u,i)$ denote the normalized amount of satisfaction actor u receives from $a(u,i)$ in its ith transaction, $Cr(v)$ denotes the credibility of the feedback submitted by v, $TR(u,i)$ denote the transaction rating v gives to u for ith transaction. For an actor v, actor u's credibility at a given time can be computed using the following equation:

$$Cr(u) = \sum_{i=1}^{I(u)} (S(u,i)) * Cr(a(u,i)) + TR(u,i) \tag{5.2}$$

The actor u's $\mathbb{TS}$ at a given time can be calculated as:

$$TS(u) = \sum_{i=1}^{V(n)} VI(k,u) * Cr(u) \tag{5.3}$$

where $V(n)$ is the total number of identifiers submitted by actor u, $VI(k,u)$ is the reputation value derived from the identifier organization k issues to u, and $Cr(u)$ is actor u's credibility as calculated in equation 6.1.

Availability Availability is synonymous with reliability in a marketplace and it applies all the more to a data marketplace. To ensure the availability of the data in case of faulty broker nodes, the data streams are replicated into multiple partitions across the broker nodes.

External Sharing: In cases where the seller has no objection to reselling the data, they will define terms of reselling in Data Stream Licenses as defined in Section IV-B-4. However, when Seller does not want to resell the data, a game-theory-based pricing model needs to be developed to make external data sharing economically unattractive. Such a practice will be discouraged as in order to make a profit, the party will quote a higher price for reselling data. Whereas, the marketplace always offers a wide variety of data at competitive prices. Yet another inhibiting factor is that IoT data, unlike other formats of data (videos, software, images) derives its value from real-time generation and consumption. Therefore, for a malicious party, it will be increasingly difficult to get hold of a constant stream of IoT data at profitable prices from a party externally sharing data without delay.

5.8 Implementation and Result Evaluation

5.8.1 System Components

For the development of the working prototype, Hyperledger Fabric was selected as the blockchain layer and VerneMQ cluster nodes as the storage and transport layer, as they closely match the intended support features in a digital marketplace. A Storage Operator may run a VerneMQ broker and/or a Fabric peer with the smart contract as described below:

Hyperledger Fabric

For the blockchain layer, a permissioned blockchain platform with support for smart contracts that would scale well in terms of transaction throughput and commit latency was required. Hyperledger Fabric (henceforth called Fabric), fits the requirement well and is easy to set up and test, hence Fabric platform was selected for implementing the PoC. Fabric is an open-source, modular permissioned blockchain platform which is developed and maintained by the Linux Hyperledger Foundation. Fabric permits pluggable components for consensus, data storage and membership service management among other modules.

Each Fabric peer in the PoC network runs an instance of the *data marketplace* smart contract. This contract maintains the listing of data offers (as key-value pairs in the ledger) including terms and mode of data streams, and client details and provides an interface to make and verify payments. The *security manager* smart contract handles the authentication and access management processes to enable access to data streams from the VerneMQ broker network.

As a core feature Fabric allows the creation of channels (logical separation of business groups) to deploy specific smart contracts and make transactions within the channel. This helps organizations avoid disclosing business sensitive information to all the participants in a channel. While for the PoC the system has been modelled as a single channel, it will be relevant to have geographic, category and organization-specific marketplaces depending on the business use-case. Each stakeholder participating organization, auditors, and government agencies could be contracted to run the ordering service used to order the transactions occurring on the Fabric blockchain.

In the PoC implementation, three types of organizations (a private firm, a government agency and a research lab) have been assumed, each contributing peer nodes as a resource in the network. For the governance structure, in Fabric there exists a Membership Service Provider (MSP), that binds the identity of each participant to a root identity provider. Each of these three types of organizations will also define an *admin peer* using their MSP which defines policies and roles in the network. And the network is governed by these policies including transaction ordering, execution and processing. In the results section, for blockchain nodes = 3, each organization type contributes one peer each and for nodes = 6, the contribution consists of two nodes from each organization.

VerneMQ Cluster

VerneMQ [156] is a distributed MQTT(Message Queuing Telemetry Transport) message broker, that ensures low latency and fault tolerant guarantees. MQTT is a lightweight, publish-subscribe network protocol deployed to transport messages between devices. The encrypted data streams are stored on the VerneMQ network nodes that are modelled in a crash-tolerant fashion. As a whole, VerneMQ provides reliable data hosting capable of tolerating faults. VerneMQ with some additional security access policies constitute the data storage layer in the proposed architecture.

Ecosystem Setup and Interaction Details

The experiment was set up on Google Cloud Computing Platform (GCP) running an instance of Linux Ubuntu-2004-focal-v20210325 configured with 16vCPUs (16 CPU cores), 64 GB RAM and 100 GB HDD storage. The complete application setup was built

using *Docker* containers. This enabled a peer-to-peer setting with each service running in its dedicated environment with compute and memory resource to depict a distributed system. A Fabric blockchain network was setup using dockerized containers for each blockchain node. Similarly, a docker container was used to run VerneMQ cluster. These docker containers communicate with each other with the help of Docker Swarm, an orchestration service, in a peer-to-peer fashion as they are on same network. After the blockchain and MQTT cluster are ready, next the DApp i.e. smart contract are developed.

Data marketplace and *Security Manager* DApp are implemented as Golang applications interacting with blockchain networks as smart contracts. Sellers and Buyers are implemented as Golang application running in docker containers. A large number of Sellers and Buyers can be generated using docker containers capable of sending simultaneous queries and/or transactions to the blockchain network by invoking the respective API/functions in the DApp. Both these entities can publish and subscribe to IoT data streams hosted by VerneMQ brokers after authenticating at the security manager DApp. All the components are run as Docker containers orchestrated using Docker Swarm for the experiments. Once all the components are ready including the Fabric, VerneMQ cluster, Smart Contracts/DApps, Seller/Buyer application, a predefined set of calls are made to the smart contracts according to Algorithm 1 and 2 of the proposed framework to complete onboarding phase and data upload phase.

To analyse the performance of the experimental setup, a benchmark tool written in Go v1.15.0 is used. The benchmark tool helps to create several virtual sellers and buyers that send simultaneous requests to the blockchain network. The IoT data used for the testing is generated by virtual clients written in Golang. The generated data is uploaded on data streams hosted by VerneMQ clients using publish service of MQTT client. Depending on the file size given as input by the user, a script recursively generates bytes of data at constant intervals to emulate IoT sensor data generation from a hardware sensor. Similarly, using the subscribe service of an MQTT client a buyer is able to access the data on broker service. The complete open-source implementation of the data marketplace project is available online at Github [157].

5.8.2 Evaluation

The evaluation experiments were run by varying several parameters to understand the dependency on each of them. The parameters are the number of MQTT cluster nodes, the number of blockchain nodes per organization and the file size to be uploaded and downloaded. In each experiment, the values of a single parameter is changed keeping the other parameters constant, to understand its impact on performance. For the variable parameter number of blockchain nodes, two cases are considered namely when blockchain nodes = 3

and blockchain nodes = 6 nodes. It is assumed there are three types of organizations (a private firm, a government agency and a research group) each contributing peer nodes as a resource in the network. Hence. when blockchain nodes = 3, it means that each of the above specified types of organizations contributes one peer node each in the network and when blockchain nodes = 6, each organization contributes two nodes each in the network. The Fabric blockchain network was run in the form of Docker containers. This allows each node to run independently of other nodes. Each experiment is run 5 times and the averaged observations are reported.

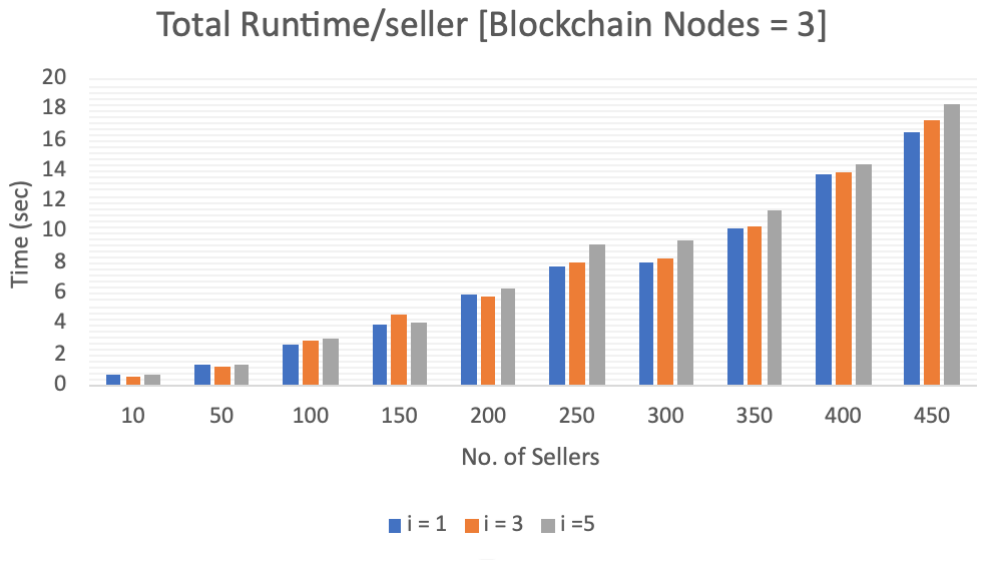


Fig. 5.5 The graph represents the variation of Total Runtime/Seller against no. of sellers when blockchain peer nodes = 3

Varying load It is marked how latency observed by a Seller varies with overall system load both for blockchain node count 3 and 6 each for storage cluster (i) count as 1, 3 and 5. The latency is measured using two parameters i.e. total runtime for each seller as shown in Figure 5.5, 5.6 and upload runtime as shown in Figure 5.7 and 5.8. The total runtime includes the time taken by a Seller to register on the blockchain network and add data offers on the network while upload time is the time taken by a client to authenticate and upload data on storage operators. With the increase in concurrent sellers, the total runtime increases however, since this is a one-time process, it is feasible for each seller in the network. The upload runtime is comparatively low as compared to the total runtime which is a good practical advantage as data upload will be done more often by the sellers.

Varying Blockchain Nodes It can be seen from Figures 5.5 and 5.6, that with an increase in the number of blockchain nodes the total runtime and upload runtime do not come down sharply. This is due to the fact with an increase in the number of blockchain nodes, the synchronization time among nodes increases. The real bottleneck is transaction processing,

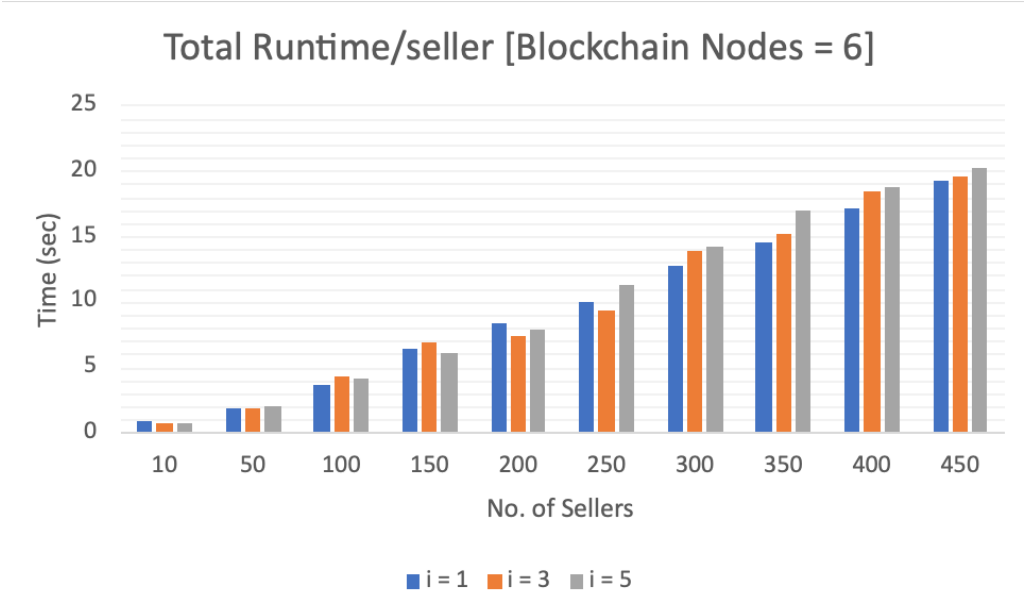


Fig. 5.6 The graph represents the variation of Total Runtime/Seller against no. of sellers when blockchain peer nodes = 6

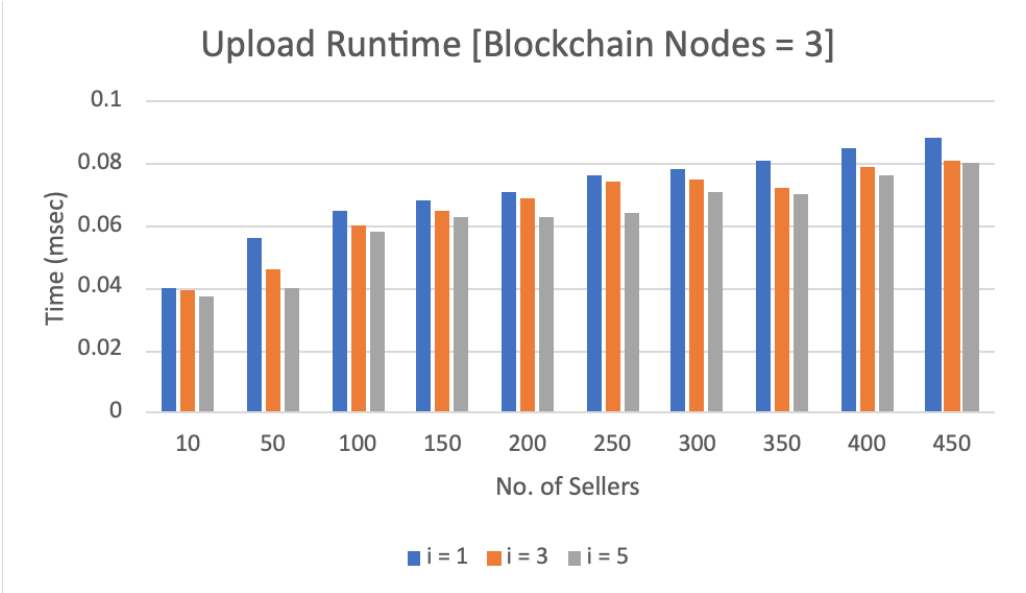


Fig. 5.7 The graph represents the variation of Average Upload Time against no. of sellers when blockchain peer nodes = 3

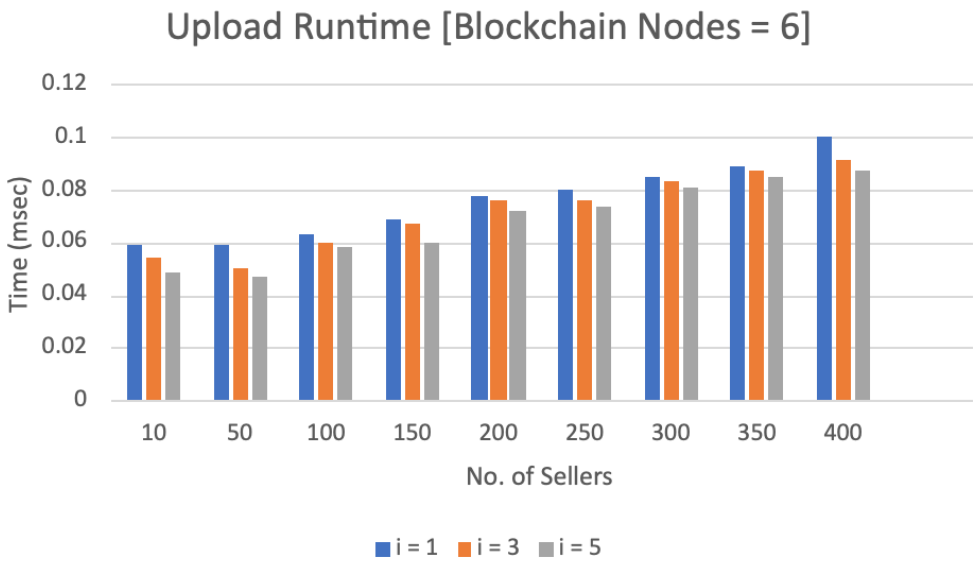


Fig. 5.8 The graph represents the variation of Average Upload Time against no. of sellers when blockchain peer nodes = 6

not the consensus protocol as elaborately explained in the work [158]. However, this only comes into play for state-changing transactions and not for query-based queries.

Varying number of storage operators The results for variation in the number of storage operator clusters can be witnessed for all the results. The upload runtime shows positive trends for average upload time with an increase in storage operator clusters as shown in Figure 5.7 and 5.8, which will improve the publishing speed of the sellers hence increasing the overall throughput as shown in the Figure 5.9 and 5.10. Therefore horizontal scaling helps in the case of storage operator nodes. For blockchain nodes, vertical scaling along with sparse peer node that selectively commits transactions as proposed in [158] will be advantageous.

Upload throughput is the total bandwidth (msgs/sec) processed by the system under the concurrent load of Sellers. The upload throughput shows similar performance trends as total runtime/seller. However, the performance increases with an increase in the number of storage operators.

Varying file upload frequency The impact of data stream sizes is observed on client latency. Figure 5.11 and 5.12 shows the latency impact for both upload of data streams, for file sizes ranging from 100 B/sec to 1 MB/sec. As expected, the latency increases sub-linearly with file size. The overhead is within 10% of the baseline.

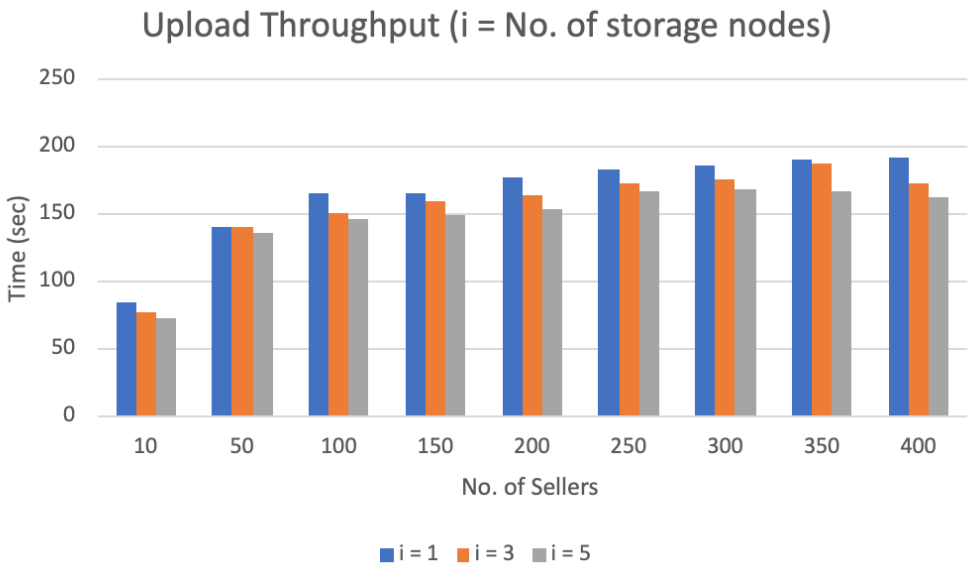


Fig. 5.9 The graph represents the variation of Upload Throughput against no. of sellers when blockchain peer nodes = 3

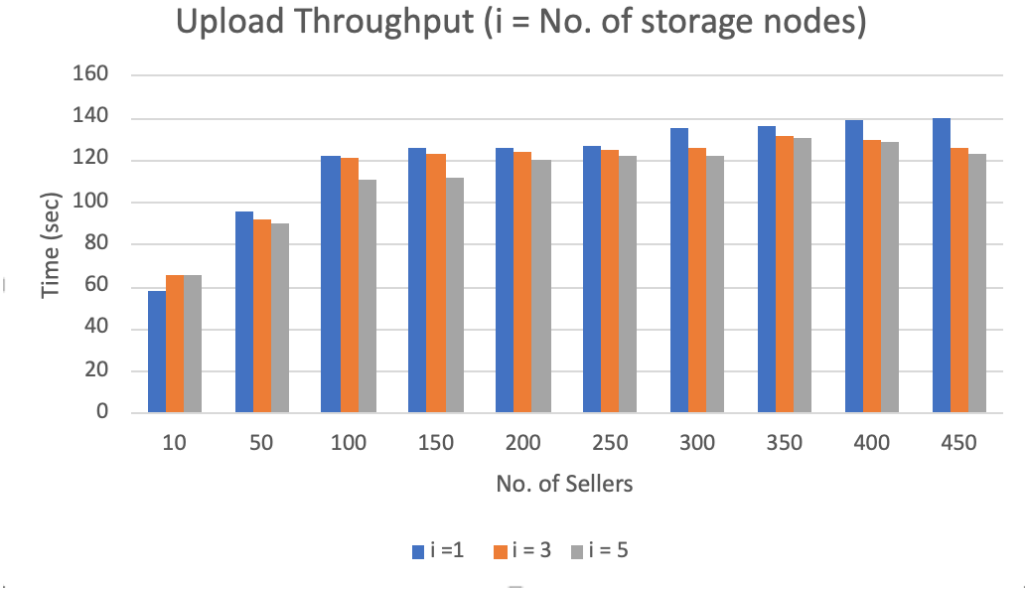


Fig. 5.10 The graph represents the variation of Upload Throughput against no. of sellers when blockchain peer nodes = 6

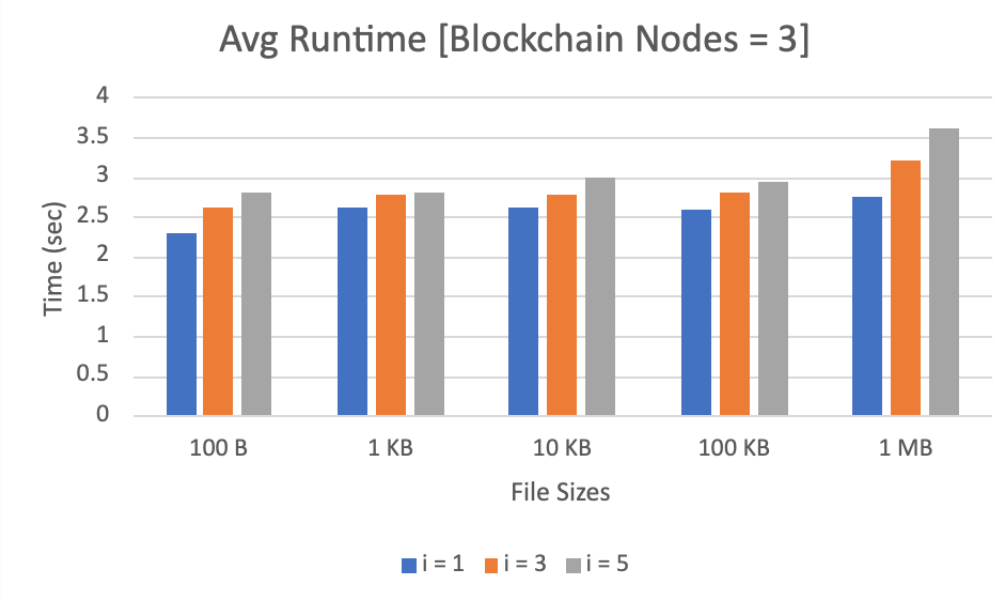


Fig. 5.11 The graph represents the Avg Runtime with file size variation when the number of blockchain peer nodes = 3

5.9 Security Analysis

Security is a prominent feature of any framework and our proposed framework should be no exception to this. In this section, the proposed architecture is evaluated for the following security threats: Man-in-the-Middle attacks, DDoS attacks, Sybil attacks, Information Disclosure and Forking Attacks. These attacks have been specifically chosen as they are critical from a decentralized architecture point of view.

5.9.1 Man-in-the-Middle (MITM) Attack

In order to prevent MITM attacks, several defence mechanisms have been provided in the architecture. Firstly, all the communications taking place between architecture components happen over a secure channel over TLS/SSL. Secondly, all the transactional messages are digitally signed with the secret key *sk* of the sender which makes it difficult for an impersonator to replicate them. Each upload and download request is completed using a unique one-time token, ω and δ respectively that provides resistance against replay attack. Besides, since the sellers and clients in the network possess unique *did*, secured by a public-private key pair, an attacker cannot inject malicious signed messages.

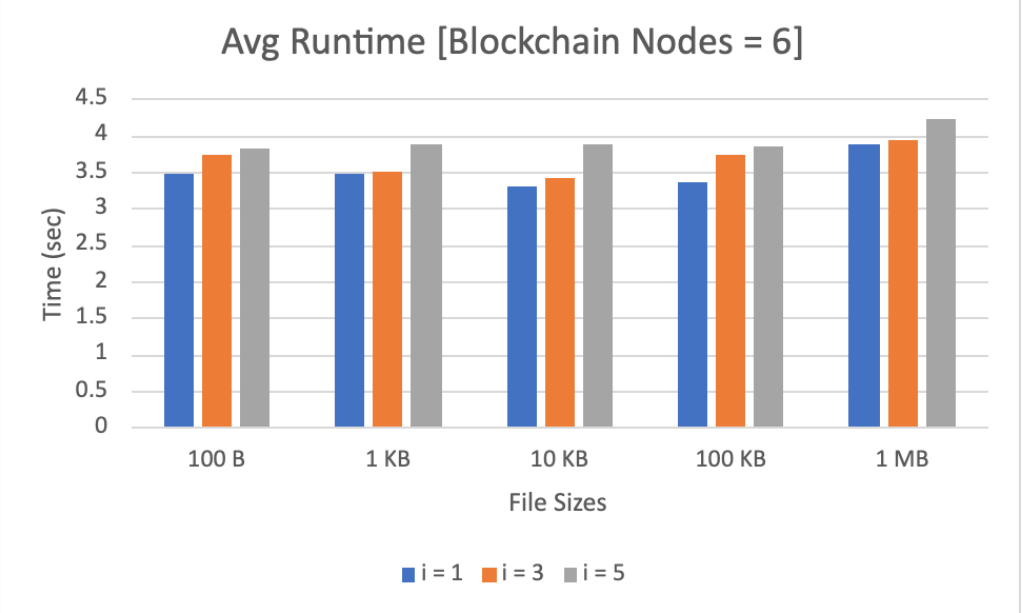


Fig. 5.12 The graph represents the Avg Runtime with file size variation when the number of blockchain peer nodes = 6

5.9.2 Distributed Denial of Service (DDoS) attack

DDoS attack can be mitigated by using unique identities in the network as proposed in our architecture. The blockchain network in general provides a certain level of security against malicious nodes. Moreover, since the blockchain in our architecture is a permissioned network, only trusted nodes are added to the network. In addition, only entities whose *did*s have been successfully verified are allowed to make accounts and later query/send transactions in the network thus, preventing malicious traffic attacks on the network.

5.9.3 Sybil Attack

Sybil attacks are a major challenge in public decentralized networks where the same node can fake multiple identities. The proposed framework has an advantage over this attack as the blockchain network is a permissioned network. In this network, the consensus over a set of transactions is achieved only after receiving endorsements from a number of verified nodes. Hence, no single node or a set of nodes holds absolute power. In the Fabric blockchain, identity certificates are issued to each node and hence, identity faking becomes difficult.

5.9.4 Information Disclosure Attack

In the proposed framework, the information stored on the blockchain can strictly be queried only by verified actors of the system. For e.g., a client can only request key release once they have made full payment for the data. In order to query *DO* which includes price, metadata and terms of data exchange, an entity needs to verify their *did*. Similarly, no suspicious entity can gain access to the data stored on broker nodes until they authenticate themselves at *Security Manager DApp* using an authentication token.

5.9.5 Forking Attack

Hyperledger Fabric blockchain differs from well-known public blockchains like Bitcoin and Ethereum in the sense that in Fabric the consensus among nodes is a three-step process which includes execute-order-validate unlike order-execute. First, the transactions are checked for correctness and endorsed by endorsing peers. Then, they are ordered via a consensus protocol and lastly, they are validated according to application-specific endorsement policy before committing transactions to the ledger. Hence, the proposed architecture can prevent forking attacks.

5.10 Conclusion

In this chapter, a decentralized and fair marketplace for IoT data trading is proposed. Such a marketplace supports two different type of trading models: a pay-per-purchase and subscription model for compensating the data owners/producers is presented. By leveraging an innovative combination of blockchain, peer-to-peer storage and smart contracts, the system guarantees fairness to all participants, the privacy of content from peers involved in delivering data, and a configurable level of fault tolerance and availability, all without reliance on a central facilitator.

The design, implementation and analysis of such a decentralized data marketplace is performed. The design implementation of the proposed solution was thoroughly evaluated to gain insight into the scalability and performance of such a system. The evaluation shows that the system is scalable with low overheads.

Chapter 6

A Privacy-Aware Authentication and Usage-Controlled Access Protocol for IIoT Decentralized Data Marketplace

This chapter is the fourth and final chapter describing the research contribution of this dissertation. The content of this chapter, which covers aspects widely described in Publication IV, focuses on the related challenges.

Data is ubiquitous, powerful and valuable today. With vast instalments of Industrial Internet-of-Things (IIoT) infrastructure, data is in abundance albeit sitting in organizational silos. Data Marketplaces have emerged to allow monetization of data by trading it with interested buyers. While centralized marketplaces are common, they are controlled by few and are non-transparent. Decentralized data marketplaces allow the democratization of rates, trading terms and fine control to participants. However, in such a marketplace, ensuring privacy and security is crucial. Existing data exchange schemes depend on a trusted third party for key management during authentication and rely on a 'one-time-off' approach to authorization. This work proposes a user-empowered, privacy-aware, authentication and usage-controlled access protocol for the IIoT data marketplace. The proposed protocol leverages the concept of Self-Sovereign Identity (SSI) and is based on the standards of Decentralized Identifier (DID) and Verifiable Credential (VC). DIDs empower buyers and give them complete control over their identities. The buyers authenticate and prove claims to access data securely using VC. The proposed protocol also implements a dynamic user revocation policy. Usage-controlled-based access provides secure ongoing authorization during data exchange. An open-source library is developed to test the proposed protocol for a decentralized IIoT data marketplace.

6.1 Background and Motivation

Sensing-as-a-service is a concept that allows data owners to sell and/or exchange sensor data with consumers interested in large open markets. Data owners with the capability to generate vast amounts of sensor/IoT data engage in such a data exchange primarily for monetization purposes. A data marketplace is an open platform that allows buyers to look up, browse and purchase the most suitable data according to their needs from a vast array of heterogeneous data hosted on it by various sellers in domains like health, automotive, manufacturing, agriculture [9] etc. Unlike centralized data marketplaces [142] which are governed and managed by a handful of central authorities, a decentralized data marketplace leveraging distributed ledger technology (DLT) like blockchains is more transparent, enabling sellers to set their rates and terms of data exchange. Aiming to explore these benefits, multiple studies such as [159] and [160] have attempted to design such marketplaces for data sharing. A decentralized data marketplace based on DLT is essentially a trading portal implemented as a DApp with the help of blockchain smart contracts and a decentralized data streaming network to host IoT data in a reliable and fault-tolerant manner. DApp is a decentralized software application that interacts with the blockchain back-end using smart contracts. The smart contracts perform functions such as listing, onboarding actors, enabling data upload and other business logic related to the trading of data. The data streaming network is hosted by a cluster of decentralized peer-to-peer (p2p) nodes i.e. *storage operator* nodes that host IoT data on behalf of *sellers* as they may not have the infrastructure or will to do so. *Storage operator* nodes host the data in the form of data streams and are incentivised for their service. It is worth noting that while the work in [159] focuses on aspects of IoT data marketplace like defining architecture and processes that assure fair settlement of compensation between all the parties, enabling unbiased content listing using calculated trust metric, converting business logic into an automated set of transaction rules for trading and fault tolerant and assured data delivery; it does not focus on other aspects of a data marketplace like security and privacy of users in such a marketplace, secure authentication and authorization related flows that provide granular monitoring of resources being accessed on the marketplace. The latter is the focus of the work presented in this work.

While decentralized IoT data marketplaces have enabled sellers and buyers to interact and engage in data trading in a fair and transparent manner, security and privacy concerns exist. To highlight the first concern, the conventional state-of-the-art marketplaces deploy a centralized identity infrastructure which gives them unrestricted view to user's activities on their platform. While centralized identity systems are easy to maintain and deploy, they have serious security challenges like a single point of failure, lack of user control, data breaches, linkability and eventually privacy threats. Additionally, a malicious

storage operator can easily trace the activity of *buyer* and gather personal details while transacting on the marketplace. Tracking the activities of a *buyer* allows the attacker to obtain their digital footprints and understand their purchase patterns. Marketing and advertising companies can target the *buyer* to send unsolicited advertisements or even influencing their choice when browsing the marketplace. This is a familiar trend in centralized data marketplace where the marketplace operator has full knowledge of the activities of every seller and buyer on their platform.

The second security concern lies in the fact that the existing data marketplace authentication protocols do not consider the importance of *buyer* empowerment; the user is not at the centre of such designs. Traditional designs deploy a trusted party in a centralized system to manage the identities and associated keys of *sellers* and *buyers* which leads to the risk of information leakage. In such systems, the exposure of hundreds of user data is only a hack away.

In a data marketplace *buyer* i.e. the user needs to be authenticated and authorized in a secure manner by the *storage operator* to establish their claims and rights to access data. The third security concern in a data marketplace is that a malicious *buyer* can eavesdrop on the communication and use the obtained credential to get authorization that works in a 'one-time-off'. 'One-time-off' format is witnessed in traditional authorization schemes such as Role-Based Access Control (RBAC) or Attribute-Based Access Control (ABAC). Such access control models are tightly coupled to the user's identity to perform authorization only once before the access is allowed and do not continuously monitor the usage of its hosted resource or evaluate change in the user's attributes during resource usage. While traditional access schemes work well for static data like video content, files, images, etc., IoT generates real-time data that draws its value from its *freshness*. Thus, in a real-time data access scheme for a data marketplace, continuous usage-based monitoring is required. In addition, a dishonest *buyer* who failed to complete previous payments should be barred from accessing any more data streams. Therefore, ensuring the security and privacy of all the parties in the marketplace is essential.

The proposed work focuses on the above-discussed issues of an IoT data marketplace. The discussed challenges in a data marketplace need a new approach to how identities are established and claims are presented and verified between the user and verifier. The concept of decentralized identity is employed in this work to solve these challenges. While the overall security of the system is crucial, maintaining the privacy of the users is paramount too. DIDs and derived pseudo identifiers in the proposed framework ensure the credibility of the user without disclosing any PII related to them. When these DIDs are bound with the issued VCs from an authorised issuer, they can convey the user's claims in a privacy-preserving way to the verifier. Hence, data minimization is ensured in all the communications between the issuer, holder and verifier. ZKP-enabled VC will allow

the holder to share only the proof of their claims with the verifier. Hence, the privacy of the user is maintained and the digital footprint is reduced. Dynamic revocation in a marketplace is crucial so that malicious actors cannot continue to operate and cause any harm to the rest of the users/services. Finally, continuous monitoring of the data streams being accessed is crucial for two main reasons, (i) to ensure fair compensation to the seller and storage operator for the data and (ii) to lock out malicious actors from further accessing the data streams.

This work proposes a user-empowered, privacy-aware, authentication and usage-controlled access protocol for a decentralized IoT data marketplace. The proposed protocol leverages the concept of SSI to enable user empowerment. To tackle the challenge of privacy preservation, remove reliance on a centralized key-managing system, and establish trust among involved parties, the concept of DID [161] and VC [162], two main standards of SSI are employed. Though DID and VC have emerged as promising technologies to enable decentralization and user empowerment, they have not addressed user revocation. Dishonest *buyers* should be removed from the system, thus the proposed protocol implements a dynamic *buyer* revocation. An efficient Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (zk-SNARK) proof is also provided, which is a privacy-preserving way to present and verify VC. After secure authentication, the $Buyer's$ claims to access the data stream is evaluated against its associated policies and on successful evaluation, access is allotted. Thereafter, usage of resources is monitored on an ongoing basis and policies are continuously evaluated until the data consumption is completed. In the proposed work, after successfully establishing the legitimacy of the *Buyer*, the *storage operator* runs a usage-controlled access protocol until the completion of data consumption by the *buyer*. The proposed work is an extension of our earlier work [159] that presented a fair, secure and trusted IIoT data marketplace. The key contributions of this work are the following:

- **A novel buyer-empowerment based privacy-aware authentication protocol based on DID and VC:** The proposed protocol enables buyers' autonomy and eliminates reliance on a centralized identity infrastructure. The buyers present Zero-Knowledge proof (ZKP) of their VCs to *storage operator* to access data.
- **A usage-controlled access protocol to monitor usage and buyer attributes on an ongoing evaluation basis:** The proposed protocol evaluates access control decisions based on continuous monitoring of IoT data resource usage to provide efficient and secure control. *Storage operator* nodes run this protocol on their end to control access to hosted IoT data resources to *buyers*.
- **A dynamic buyer revocation policy**: Buyers on the platform who are dishonest, have non-payment dues or violated terms of data exchange are put on the revocation

list. A *Buyer* must prove to the *storage operator* that they are not on the revocation list to request data stream access. This is achieved in a privacy-aware manner.

- **Proof-of-concept (PoC) implementation and Evaluation**: A PoC is implemented with wallets using Hyperledger Indy Blockchain and a decentralized real-time usage-controlled access policy is implemented using Linux utility: Cronjob. The work includes a discussion of performance and security analysis.

6.2 Related Work

Blockchain-enabled data sharing: Data exchange facilitated by blockchain has garnered significant attention due to inherent benefits like non-repudiation, and providing a trusted sharing service, where information is reliable and can be traced [163]. In [164], authors present a blockchain-based cloud storage protocol for sensors in IIoT using a group signature scheme. Smart contracts and proxy re-encryption is used to realize data sharing in an industrial facility. A blockchain-enabled dynamic and traceable data-sharing scheme for the smart factory is proposed in [165]. Blockchain performs the authentication and stores the ciphertext index and public keys to avoid tampering. The tracking algorithm tracks malicious users and adds them to a revocation list. The authors in [166] propose a private data-sharing framework using blockchain for applications such as tracking products for counterfeit detection, product recall and compliance with regulations. A blockchain-based hierarchical data-sharing framework to provide fine-grained access control and retrieval over encrypted personal health records between multiple stakeholders is presented in [10]. The authors in [167] propose a monitor-based usage control model to enforce data usage policies on the user side using blockchain and SGX (Software Guard Extensions). Though the above-discussed schemes ensure the secure exchange of IoT data and while some of them also focus on the privacy of the sensors, most of them use a centralized model for data exchange and do not consider a multi-stakeholder environment. Some of them store the data on blockchain and most of them use complex key-distribution mechanisms. In some of these schemes, the system master secret key was generated in a centralized manner by a trusted party. If a trusted party is compromised by an adversary, there is the risk of leakage of the system master key. Further, most of them have considered a 'one-time-off' access control technique that does not protect against insider attacks during resource usage.

IoT Data Trading: On the other hand, literature on IoT data *trading* or Data-as-a-Service model for monetization has also seen progress [168],[169]. Such systems mostly involve p2p data sharing model where data owner can decide the *rates, terms* and *policies* for data trading [170], [159]. Figueredo et al. in [171] discuss one such practical IoT data

marketplace framework 'oneTRANSPORT' implemented between various public bodies in the UK. A blockchain-based solution for resource-constrained IoT streaming devices that allows data chunks to be transferred in a decentralized, traceable and secure manner has been proposed in [172]. [160] present a 'Databox-based' delivery service via blockchain to provide data consumers with secure and controlled access to the requested data source of interest. Trading of IoT data in other domains like health [173] and energy data [174] has also witnessed interest. Though the above-mentioned works try to address different issues related to an IoT data trading marketplace, they lack details on users' privacy, use empowerment and most importantly monitoring IoT data usage when it is traded on such decentralized platforms. User authentication, identity verification and usage-control layer are missing from the above-mentioned proposals. Proving the access rights and user's claims in a privacy-preserving manner remains unaddressed in the state-of-the-art decentralized data marketplaces.

Use of DID in IoT: The concept of DID and VC has recently gained popularity as a viable privacy-preserving method to enable identity empowerment in IoT [175], [176]. The use of DID for IoT applications was discussed in [117]. The work indicates the feasibility of DID even on devices with limited resources. IoT-as-a-Service (IoTaaS) concept to offer IoT device services using DID and VC on a demand basis has been proposed by [118]. A solution based on DID and VC that enables a distributed V2X (Vehicle to everything) access authorization mechanism to allow a vehicle owner to prove their identity without compromising privacy was proposed in [119]. While DID enables authentication and user identification, VC can verify a set of claims that can be verified against access policy to allow secure and granular authorization [120]. [121] present a privacy-sensitive information protection and management scheme to allow automated distribution of information. Though the above-discussed schemes employed DID and VC to allow users to control their identity, they have not considered key issues like dynamic user revocation. The schemes in [120] and [121] have not considered the critical issue of continuous resource usage monitoring in an access control scheme.

The above-discussed works clearly outline the importance of IoT data trading and the use of blockchain technology for such data exchange/trading applications. Blockchain offers several advantages in a multi-stakeholder environment where parties inherently do not trust each other and wish to achieve transparency, and non-repudiation and avoid siloed views of transactions. A few of these data trading applications also propose the introduction of DID and VC as identity pillars to remove the centralization of identity management. To fulfil the above-identified research gaps, a privacy-aware, authentication and usage-controlled access protocol is proposed for IIoT data marketplaces. The proposed work not only leverages the use of DIDs and VCs for providing mutual authentication between the parties coming together to trade on such a platform but it's also done in a privacy-

Table 6.1 Comparison of Blockchain Marketplace Solutions based on design features [*P* = *Proposed work*]

Design Features	[174]	[172]	[168]	[121]	[160]	[166]	[169]	[177]	*P*
Privacy of User	-	-	+	+	-	+	+	+	+
User empowerment	-	-	-	+	-	-	-	+	+
Mutual Auth	-	-	-	-	-	-	-	-	+
Decentralized ID	-	-	-	+	-	-	-	+	+
Dynamic Revocation	-	-	-	-	-	-	-	-	+
Non-Repudiation	+	+	-	+	+	+	+	+	+
Unlinkability	-	-	-	+	-	-	-	+	+
Multi-Stakeholders	+	+	+	-	+	+	+	+	+
Access-Policy	-	-	-	+	+	+	-	-	+
Usage-Control	-	-	-	-	-	-	-	-	+

enhancing way so that tracking of engaging parties is not feasible for the marketplace service provider. This inhibits behaviour such as exhibiting biased search results, targeted ads or correlation of a user's activities on the marketplace. The traceability is sufficiently difficult to achieve with the help of DIDs. Our solution also provides anonymity for trading parties in case they wish to do so using ZKP-enabled VC. Our protocol makes dynamic revocation of dishonest parties from the marketplace so that they can no longer engage in trading activities. Once mutual authentication between trading parties is achieved, our protocol continuously monitors user and resource attributes to achieve a more secure transaction using usage control. A feature-wise comparison of the most closely related works discussed above is presented in Table 6.1.

6.3 Proposed Protocol

6.3.1 Overview

A privacy-aware, authentication and usage-controlled access protocol for IIoT data marketplace is presented in this section. The primary actors in this protocol are the $Seller(SL)$, the *Buyer* and the $StorageOperator(SO)$ nodes as shown in Figure 6.1. These system actors in our proposed protocol can be mapped to the VC ecosystem actors presented in the previous section. In the proposed work, the *SL* acts as an *issuer* as it issues VC to both *SO* and the *buyer* signed by its private key K_{SL}^{pr}. *SO* and *Buyer* both are *holder* as they collect VC from *SL*. While it's certain that since an *SO* can verify any verifiable presentations from any holder, containing proofs of claims from any issuer, consequently it is a $verifier$ [57]. However, each *buyer* may or may not have that capability and it will depend if it can act as a *verifier* for each type of verifiable presentation from every issuer. The proposed

protocol, assumes that both *SO* and *buyer* will act as verifiers and will perform mutual authentication based on issued VCs.

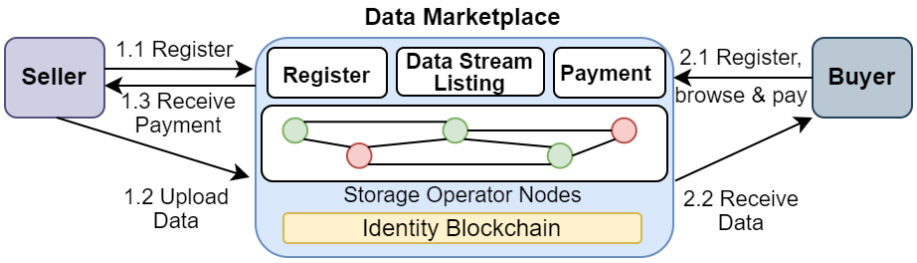


Fig. 6.1 System Actors in the Proposed Protocol

The data marketplace provides a range of services including data listing, registration, payment for data etc. Identity Blockchain provides a decentralized identity infrastructure to log registration of actors and issuance of VCs. The phases in the proposed protocol are *system setup and registration*, *authentication* and *UCON phase*. The high-level workflow is illustrated in Figure 6.2. The system setup and registration phase is only performed once. In the setup phase, all the participants register themselves by creating DID and associated keys on the blockchain. After this step, the *Buyer* can browse the listed data streams on the marketplace advertised by several *SL* with their *rates*, purchase mode (*batchMode* or *streamMode*) [159] and *terms* etc. The purchase mode defines the frequency of payment made by the *Buyer*. In registration phase, the *Buyer* interested in purchasing a data stream from *SL* presents a signed digital identity to the *SL* for verification. If the digital identity verification along with requested payment is successful, the *SL* generates a VC with claims including the $payment_{ID}$. *SL* signs the VC with its private key and issues it to the *Buyer*. Similarly, SO_j also receives VC from *SL* after identity verification. In the proposed protocol, EdDSA (Edwards-curve Digital Signature Algorithm) [77] scheme based on elliptic curve cryptography (ECC) is used for VC signing and verification.

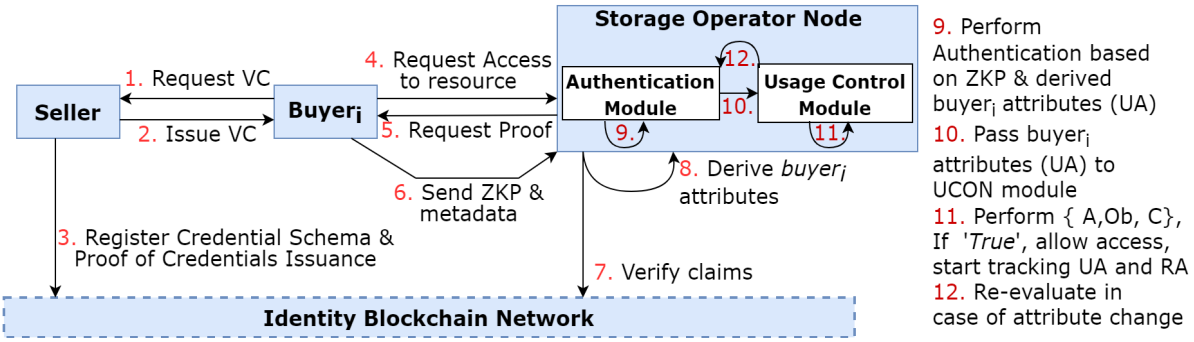


Fig. 6.2 Overview of proposed protocol with detailed flow of interactions between actors

Table 6.2 Notations

Symbol	Description
G	Generator point
DID_x	DID of x
$PDID_i$	Pseudo-identifier of $Buyer_i$
K_x^{pr}	Private key of x
K_x^{pu}	Public key of x
K_i	Secret key between SL and $Buyer_i$
K_{DBj}	Secret key between SL and DOj
$payment_{ID}$	Payment ID given to *Buyer* after payment
$[VC_i]_{K_{SL}^{pr}}$	VC of $Buyer_i$ issued by SL
$[VC_{SO}]_{K_{SL}^{pr}}$	VC of SOj issued by SL
$[VC_{DigIDi}]_{K_{MC}^{pr}}$	VC of $Buyer_i$ issued by market consortium
$\oplus$	XOR Operation
$h(X)$	Hash of X
ϕ_{id}	$stream_{ID}$ of IoT data
w_i	Non-membership witness of revocation List L

After registration phase, the *Buyer* goes through authentication followed by usage-controlled access phase with the *SO* hosting the purchased data stream on behalf of *SL*. To authenticate, *Buyer* sends the data access request to the *SO*. It then generates the ZKP of the VC and sends it to the *SO*. The SO_j verifies the received proof to confirm the $Buyer's$ right to access a data stream during the authentication part of the protocol. Thereafter, the usage-controlled access protocol evaluates the $Buyer's$ attributes presented in VC to authorize it. From thereon, the buyer and resource attributes are monitored on an ongoing bases until the data consumption is completed/terminated. The notations used in the proposed protocol are presented in Table 6.2.

6.3.2 System Setup and Registration Phase

System Setup

Step 1: The DID document corresponding to the *SL*'s DID, DID_{SL}, is stored on the blockchain. *SL* generates a private key K_{SL}^{pr}. The public key is generated through EdDSA key generation function. Let G be the generator point. The public key K_{SL}^{pu} is a point on the elliptic curve, calculated by EC point multiplication as $K_{SL}^{pu} = K_{SL}^{pr} * G$. Thereafter, K_{SL}^{pu} is stored in the *SL*'s DID document.

Step 2: $Buyer_i$ generates a DID, DID_i and stores the corresponding DID document on the blockchain. Then, $Buyer_i$ generates a pair of private (K_i^{pr}) and public (K_i^{pu}) keys using

the EdDSA key generation function similar as discussed in Step 1. The *Buyer* stores K_i^{pr} in its digital wallet. K_i^{pu} is stored in *Buyer*'s DID document.

After this step, the *Buyer* is onboarded on the data marketplace platform and can start browsing the listed data offers from various $sellers$. Once the *Buyer* decides to purchase a data stream with ID, ϕ_{id}, it enters into an agreement with its *SL* using the registration phase.

Step 3: The Storage Operator, SO_j also generates its DID (DID_{SOj}) and stores its public key immutably in its DID document on the blockchain.

Step 4: *SL* generates an accumulator, a revocation list L and a secret key k_{Acc}. Initially, L does not have any elements. As discussed in Section **??**, *SL* generates an accumulator U from k_{Acc} and L using G_{Acc} function.

Step 5: The SO_j hosts data streams with associated usage-controlled access policies as defined by *SL* in order to provide access to legitimate *Buyer*s only on providing valid set of attributes.

Registration Phase:

In this phase, the registration of the $Buyer_i$ with a *SL* on the marketplace takes place to initiate data trading. The SO_j also register on the marketplace as it agrees to host data for *SL*.

Buyer Registration: consists of the following steps:

Step 1: The *Buyer* holds $[VC_{DigIDi}]_{K_{MC}^{pr}}$, the VC of their digital identity issued by a trusted party (e.g., marketplace consortium) after KYC verification. The process is very similar to an e-Know Your Customer (eKYC) process done by an organization to onboard customers. Next, $Buyer_i$ generates a pseudo-identity $PDID_i$. The *buyer* can generate a new pseudo-identity for trading with each new seller. In this way the activity of *buyer* on the marketplace cannot be tracked and linked. The *Buyer* decides to purchase the data stream with ID ϕ_{id} using mode $md(batch/stream)$ from *SL*. If the mode selected is $batch$, the *Buyer* will need to pay only once for a burst mode of data, whereas in $stream$ mode, payment needs to be made periodically to receive real-time IoT data [159]. Then, the buyer generates a message M_{REG1} with a registration request, streamID ϕ_{id}, mode md, DID_i, $PDID_i$ and $[VC_{DigIDi}]_{K_{MC}^{pr}}$ and sends it to *SL*.

Step 2: The *SL* verifies the $Buyer's$ digital identity using the EdDSA's signature verification. After that, *SL* generates a nonce n and a key K_i for future communication with $Buyer_i$. A nonce is a random number used only once to provide various types of security guarantees. At this stage the nonce n prevents against reflection attack as explained later. Then, *SL* generates a payment request Pay_{req} for requested stream ID ϕ_{id} to be consumed in the mode md. Then, the payment request, K_i and n are concatenated and its hash value

Algorithm 6: *Buyer* REGISTRATION PHASE

1: *Buyer* :
Generate: DID_i, $PDID_i$
Generate: ${K_i}^{pr}$, ${K_i}^{pu}$ pair
Select: data stream ID (ϕ_{id}) & $md(stream/batch)$
M_{REG1} :
$\{Req_{id}, \phi_{id}, md, [VC_{DigIDi}]_{K_{MC}^{pr}}, DID_i, PDID_i\}$
$Buyer \rightarrow SL$: M_{REG1}

2: *SL*:
$Verify_{EdDSA}$: $[VC_{DigIDi}]_{K_{MC}^{pr}}$
Generate: K_i, nonce n
Generate: Payment Req for (ϕ_{id}) in mode md
hV = h ($Pay_{req} \| K_i \| n$)
M_{REG2} : $\{hV, Pay_{req}, K_i, n\}$
$SL \rightarrow Buyer$: M_{REG2}

3: *Buyer* makes payment and receives $payment_{ID}$
Generate: N_i
$hV2$ = h ($payment_{ID} \| K_i \| n \| N_i$)
M_{REG3} : $\{hV2, payment_{ID}, N_i\}$
$Buyer \rightarrow SL$: M_{REG3}

4: *SL*:
Generate: n^{new}, credential $cred$
hV_{cred} = h ($cred \| K_i^{pu} \| n^{new}$)
Sign $hV3$ with K_{SL}^{pr}
i.e., $[VC_i]_{K_{SL}^{pr}} = Sign_{EdDSA}(hV3)$
$n^{*new} = K_i \oplus n^{new}$
$w_i = G_w(U, k_{Acc}, L, PDID_i)$
M_{REG4} : $\{[VC_i]_{K_{SL}^{pr}}, cred, n^{*new}, w_i\}$
Store: $DID_i, PDID_i, n^{*new}, w_i, K_i$
$SL \rightarrow Buyer$: M_{REG4}

5: *Buyer*: **Store:** $[VC_i]_{K_{SL}^{pr}}, cred, K_i, w_i n^{*new}$

hV is computed. After that, the SL generates a message M_{REG2} with hV, Pay_{req}, K_i and n and sends it to the buyer.

Step 3: $Buyer_i$ receives M_{REG2} from SL. $Buyer_i$ makes payment on the marketplace to receive $payment_{ID}$ for this transaction. Then $Buyer_i$ generates a nonce N_i. The nonce N_i will prevent the replay attack by a malicious entity as explained later. Then, $payment_{ID}, K_i$, n and N_i are concatenated and its hash value $hV2$ is computed. Then, $Buyer_i$ generates a message M_{REG3} with $hV2, payment_{ID}$, and N_i and sends it to the SL.

Step 4: After SL has successfully verified the payment details, it generates a nonce n^{new} and a credential $cred$. The newly generated nonce n^{new} will act as a shared secret between buyer and seller for next round of interaction to enhance security as explained later. The generated credential is a set of claims that the $Buyer_i$ can assert by presenting it to the right verifier. After that, $cred, K_i^{pu}$ and n^{new} are concatenated and its hash value hV_{cred} is computed. Then the SL signs the credential with its private key using EdDSA's signature generation algorithm.

For that the SL generates a secret integer r and a random point is generated as:

$$r = hash(hash(K_{pr}^{SL}) + m) mod\, q; R = r * G \tag{6.1}$$

Then the SL calculates the signature as:

$$h = hash(R + K_{pu}^{SL} + m) mod\, q \tag{6.2}$$

$$s = (r + h * K_{pr}^{SL}) mod\, q \tag{6.3}$$

The VC, $[VC_i]_{K_{SL}^{pr}}$ of $Buyer_i$ indicates that the $Buyer_i$ is a valid registered user who has made payment for requested data stream. It can be verified using the corresponding public key K_{pu}^{SL} of the SL. The SL also generates a non-membership witness w_i for $Buyer_i$ by calling $G_w()$ function with inputs U, k_{Acc}, L, and $PDID_i$ as mentioned in Section **??**. The new nonce n^{new} is XORed with K_i to produce n^{*new}. This operation ensures that the shared secret n^{new} is only decipherable by the intended recipient with the shared secret key K_i i.e. $buyer_i$. The SL stores DID_i, $PDID_i$, n^{new} and K_i for future communications with the $Buyer_i$. After that, SL generates a message M_{REG4} with $[VC_i]_{K_{SL}^{pr}}, cred$, n^{*new} and w_i and sends it to the buyer.

Step 5: $Buyer_i$ receives M_{REG4} from the SL. It stores $[VC_i]_{K_{SL}^{pr}}, cred, K_i, w_i$ and n^{new} in its digital wallet.

SO Registration: The steps are as follows:

Step 1: Similar to $Buyer$ registration, SOj registers with SL and receives a VC, VC_{SOj} signed by SL with K_{SP}^{pr}. However, it skips the steps for payment as done by the buyer. It

Algorithm 7: AUTHENTICATION PHASE

1: $Buyer \rightarrow SO : MA1 : \{PDID_i, Usage_{req}, VC_{req}\}$
2: $SO \rightarrow Buyer : MA2$
$MA2 : \{DID_{SOj}, [VC_{SOj}]_{K_{SP}^{pr}}, Proof_{req}\}$
3: $Buyer :$
$Verify_{EdDSA}$: $[VC_{SOj}]_{K_{SP}^{pr}}$
Calculate: $hV_{cred} = \mathrm{h}\ (cred \| K_i^{pu} \| n^{new})$
Store: hV_{cred}
$\pi = Prove([VC_i]_{K_{SP}^{pr}})$
$MA3 : \{\pi, PDID_i, w_i\}$
$Buyer \rightarrow SO$: $MA3$
4: SO : **Verify**: π
$V_w(U, w_i, PDID_i) = ?1$

sends a request for VC to the *SL* to attest to its legitimacy of hosting data streams for that *SL*. It stores received VC_{SOj}.

Step 2: *SL* produces key K_{SOj} and sends it to SO_j. *SO* stores K_{SOj} for future communication with *SL*.

6.3.3 Authentication and UCON Phase

Authentication Phase

$Buyer_i$ must be authenticated by SO_j to access the data stream hosted by SO_j on its storage network on behalf of *SL*. $Buyer_i$ must also authenticate SO_j.

Step 1: After successful registration with *SL* and receiving valid VC that will help it to access the data stream it purchased on the marketplace, $Buyer_i$ authenticates with SO_j. $Buyer_i$ generates M_{A1} with $PDID_i$, requests to access stream ϕ_i and requests for the *SO*'s VC. Then, M_{A1} is sent to SO_j.

Step 2: SO_j composes and sends a message M_{A2} to $Buyer_i$ with $DID_{SOj}, [VC_{SOj}]_{K_{SP}^{pr}}$ and a request of proof of *Buyer*'s VC to validate its claim to access the data stream.

Step 3: $Buyer_i$ receives M_{A2} from SO_j. Then, $Buyer_i$ verifies SO_j's VC by using the EdDSA signature verification algorithm using K_{SP}^{pu}. After that, credential *cred*, K_i and nonce n^{new} are concatenated and the resultant hash value is computed and stored. Next, the $Buyer_i$ produces a ZKP of the VC using the *Prove* algorithm with an output π as mentioned in Table 2.3. This is ZKP of the VC and $Buyer_i$ presents it to SO_j to prove in Zero Knowledge that it holds a valid VC. Then, $Buyer_i$ generates M_{A3} using π and w_i and sends it to the SO_j.

Step 4: When SO_j receives M_{A3}, it verifies π using the $Verify$ algorithm as mentioned in Table 2.3. The *SO* can only know that $Buyer_i$ holds a valid credential and some metadata

about the data stream. It cannot learn anything about its identity. Then the SO_j verifies the non-membership witness w_i of the *Buyer* by calling V_w function as explained in Section **??** to confirm that the buyer is not revoked during this session. Once the $Buyer_i$ and SO_j are mutually authenticated and the *SO* is assured of the buyer's claims and revocation status, it passes the $Buyer's$ metadata to the policy evaluation engine that manages access of data streams on the network.

It is worth noting that during the complete process of registration and authentication, all the communications are P2P between the interacting parties i.e., there is no centralized party involved to acquire keys etc to authenticate or authorize users. It enables the decoupling of issuers and verifiers and they can function independently. For e.g. if for any reason the issuer goes offline, the verifier can still verify the authenticity and claims of the buyer.

Usage-Controlled Access Policy Evaluation

This evaluation takes place at SO_j node. After authentication is completed, $buyer's$ metadata is passed to the policy evaluation engine in the UCON module. Policy Engine parses the metadata ($PDID_i$, mode of access, ϕ_{id} etc.) received from $buyer's$ VC to evaluate the policy against its attributes. The usage-controlled access components are presented in Table 6.3. As observed the metadata along with verified VC make up the $User\ Attributes$ (UA) set. The resource in this case is the data stream that would be accessed by the buyer hence; *Resource Attributes* (RA) can be represented as a set of attributes like the seller ID who owns this data stream(SL_{ID}), name of the stream, time of upload/creation etc. *UA* and *RA* are sets of attributes that uniquely identify a user or resource. *Rights* defines how the resources can be used by the user and in this case they are represented by the mode of access and whether they can be read, downloaded or computed upon. Pre-obligation(*preOB*) checks are defined and they define that a user must provide a non-membership proof and must have completed the payment and provide a paymentID as proof. Similarly, Ongoing-Obligations (*onOB*) define that during ongoing access, the non-membership proof must hold true along with allotted time and if pre-paid balance and stream mode are opted it should be available to charge payments. In the proposed work, *preOB* would be pre-payment and proving non-membership of the revocation list. *onOB* would be evaluated on an ongoing basis until the completion of data access and it includes rechecking revocation status on an ongoing basis and/or decreasing $buyer's$ prepaid balance. Finally, *Condition* are system-oriented decision factors, hence these will be based on factors such as system load and IoT data stream real-time generation.

Since pre-payment is completed, $Buyer_i$ is verified and since it has provided a witness of non-membership (w_i) of the revocation list, it passes the authorization and pre-obligation phase. The data usage begins after this point and hence the ongoing obligation check

begins. The UCON module continuously monitors the *Buyer's* (UA) and Data stream's (RA) attributes to record any change against defined ongoing obligations. Access is terminated based on the termination of the allotted access period or in case of any of the ongoing obligation violations, whichever happens first. In short, a Usage Policy is defined as $Policy = \{UA \wedge RA \wedge Right\}_{A,OB,C}$ i.e., evaluate policy using elements of set *UA*, *RA* and *R* under defined *A*, *OB* (both pre and on) and *C*. The *allow(UA, RA,R)* predicate indicates that a user with attributes *UA* is allowed to access a resource with attributes *RA* with rights *R* only if the indicated *preOB* conditions are fulfilled. The *Fulfilled(preOB)* predicate tells us if each of the required obligations is true to start the access. It is worth noting that the UCON policy evaluation takes place before the start of every new access request by a *buyer* to the *storageOperator* node and each session is continuously monitored until it is terminated or completed. The usage-controlled access components defined in Table 6.3 stand true for the proposed protocol and it is a business-logic dependent implementation, hence it can vary with various implementations.

Table 6.3 Usage-Controlled Access Components

Elements	**Explanation**
User/Buyer Attribute(UA)	$PDID_i$, $payment_{ID}$ $prepaidBalance$, duration=X
Resource/Data Stream Attribute(RA)	SL_{ID}, ϕ_{ID}, $streamName$, Time($DDMMYY$)
Rights(R)	$mode$(batch/stream) (Read, Download, Compute)
Pre-Obligation(preOB)	w_i, $payment_{ID} \; ! = 0$
Ongoing-Obligation(onOB)	w_i, $allotedTime > 0$ & $prepaidBalance > 0$
UCON Policy (λ)	$allow\ (UA, RA, R) \Rightarrow \lambda\ (UA, RA, R)$ $\wedge\ Fulfilled\ (preOB)$

$$UserAttributes = \{PDID, DID\}$$

$$ResourceAttribute = \{Seller_{ID}, \phi_{ID}, name, time\}$$

$$Right = \{mode(batch/stream), read, download\}$$

$$Policy = \{UA \wedge RA \wedge Right\}_{A,OB,C}$$

Usage Control defines two types of attributes for both *User* and *Resource*: *mutable* and *immutable* attributes. Mutable attributes are user/resource attributes that change as

a result of access and immutable attributes do not change during access. E.g., a *buyer's* prepaid balance has to be decreased as it consumes or accesses the data stream hosted on *storage operator* node and hence it is a mutable attribute. *PDID* is an immutable attribute in this context. UCON monitors the *mutable* attributes during access to evaluate policies on an ongoing basis.

6.3.4 Design Analysis: Dynamics of SSI in a data marketplace

In a traditional marketplace setting, the identity infrastructure deployed by the marketplace provider would usually be a centralized system. In such an identity infrastructure, the identity database is instantiated and maintained by the marketplace provider. This gives a lot of visibility, power and tracking capability to the marketplace provider. Other concern with such a centralized infrastructure are challenges like a single point of failure, lack of user control, data breaches and likability. While marketplace providers would like to ensure fair, trusted and secure way of enabling data exchange they must do so in a privacy-enhancing manner. In the proposed framework, SSI is leveraged to achieve a new approach in the way identities are established and claims supported by those identities are presented and verified.

The proposed framework establishes the identity of both buyer and seller with the help of decentralized ID (DID) after satisfying the regulatory obligation such as eKYC in a given jurisdiction. This ensures onboarding of genuine users and corporations. Later the interacting seller issues the VCs to the buyer to define their rights and access permissions for a given IoT data stream that they paid for. These VCs can be *time-bound* or *count-bound* to define the access limit of a certain data-stream. It is also possible for a seller to revoke a given VC at any time if an actor is behaving maliciously or lapses on payment.

While SSI can bring several advantages to a decentralized data marketplace, there may be certain limitations to it. SSI or decentralization in broader sense is a spectrum rather than an absolute value. It means that there can be several implementations along that decentralization spectrum. Hence, every aspect of implementation will require careful consideration. For example, in the proposed framework, if the buyer can create several pseudo-identities $PDID_i$ against its DID_x and default on payments to seller using a new pseudo-identity each time, it can lead to losses to the sellers. Similarly, if the seller acts maliciously and tries to revoke VC of a buyer without valid reason, it can impact the reputation of the platform. Therefore, monitoring the behaviour of both seller and buyer is crucial for the marketplace provider. With verified onboarding, maintaining and monitoring a TRUST SCORE and maintaining a competitive incentive model as explained in 5.4.2 can help to maintain trust in the marketplace. Therefore, right design patters and supporting incentive model is crucial along with security and privacy.

6.4 Implementation and Evaluation

6.4.1 System Components

1. *Hyperledger Indy:* The proposed framework is blockchain platform agnostic as long as it supports the digital identities rooted in blockchain. Hyperledger Indy (henceforth called Indy), an open-source project of the Linus Hyperledger foundation provides a decentralized identity implementation including support for a ledger, DID's implementation and all features related to VCs. It supports the building of applications that interact with the ledger for decentralized identity management. It is an open source, identity blockchain platform that permits pluggable components for the wallet (also referred to as an agent), DID resolver and Indy nodes among other cryptographic modules.

2. *System Actors and Roles:* The system includes the agents (henceforth called '*wallets*') of each subject i.e., *SL*, *SO* and *Buyer*. The agents are represented as separate docker containers for each actor. In the proposed case, the seller acts as the issuer of VCs to both the buyer and storage operator, while buyer and storage operators both perform mutual verification of credentials before data exchange. The *authentication module* and *usage control module* within the Storage operator node as shown in Figure 6.2 are emulated as services within the same docker container.

3. *Interaction Details:* During system setup, each actor *SL*, $Buyer_i$ and SO_j create a DID and associated key pair using their Indy wallets and register themselves on the blockchain. This only requires one call for each wallet to the ledger. Later in order for the *SL* to issue VC to the $Buyer_i$ and SO_j, it has to perform some setup. Before *SL* can issue VC, it requires completing two VC setup steps after registration on the blockchain: VC Schema generation and VC Definition generation. VC Schema generation involves the creation of the schema by *SL* and sending it to the ledger. VC definition generation is completed after two interactions with the ledger, first to get the schema of the VC and later to store the generated VC definition on the ledger. Now VC creation for issuing can take place and this comprises two calls to the ledger from the issuer's (*SL*) agent.

6.4.2 Evaluation

The experiment was set up on Google Cloud Computing Platform (GCP) running an instance of Linux Ubuntu configured with 16vCPUs, 64 GB RAM and 100 GB HDD storage. The experiments were performed by varying the number of parameters to understand the dependency on each of them. The parameters used are the number of blockchain nodes i.e.

(N), the number of claims (C) in a VC issued to a buyer and the number of *obligations*(both pre and ongoing) imposed in access policy check. The setup is tested by varying the number of claims in each VC and later by varying the number of blockchain nodes in the Indy ledger. The wallets are written in Python 3.2. To perform usage-based access control, a cronJob scheduler [178] was written which interacts with the underlying Unix operating system. For VC signing and verification EdDSA, particularly Ed25519 [77] is used which is based on a performance-optimized elliptic curve, Curve25519. Cryptosystems based on ECC have the advantage of using small key and signature sizes. Hence, they have low memory and computational resource requirements. For the ZKP system, the API implementation given in [179] is used. The implementation of this protocol is available online at Github [157]. To understand the performance of the protocol, tests on each stage are executed to see how the system performs with variations in the above-discussed parameters.

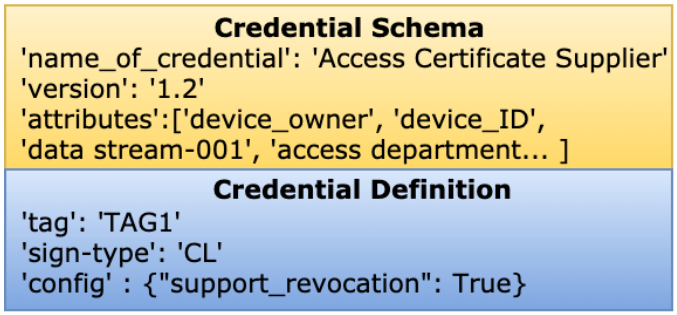


Fig. 6.3 Credential Schema details of the fields of an issued credential and Credential Definition informs the identity and public keys of the issuer

System Setup and Registration: Figure 6.4 presents the system setup time which can be divided into three steps: issuer (*SL*) registration, VC schema generation and VC definition generation as shown in Figure 6.3. These individual setup time and combined setup time for each claim and node are shown in this figure.

It can be observed that the time taken for *SL* (issuer) registration remains approximately the same with an increase in the number of claims (C) for a specific number of nodes; however, with an increase in the number of nodes (N), the registration time increases. This is due to the fact that with an increase in the number of blockchain nodes, the transaction synchronization time among nodes increases. This effect where the real bottleneck is transaction processing, and not the consensus protocol as elaborately explained in the work [158]. The transaction synchronization time can be reduced by vertical scaling of the system i.e. increasing the processing power of each node. A similar pattern is observed with time taken for VC Schema generation and VC definition generation. It is worth noting that since the system setup is a one-time process and therefore, it is a feasible overhead

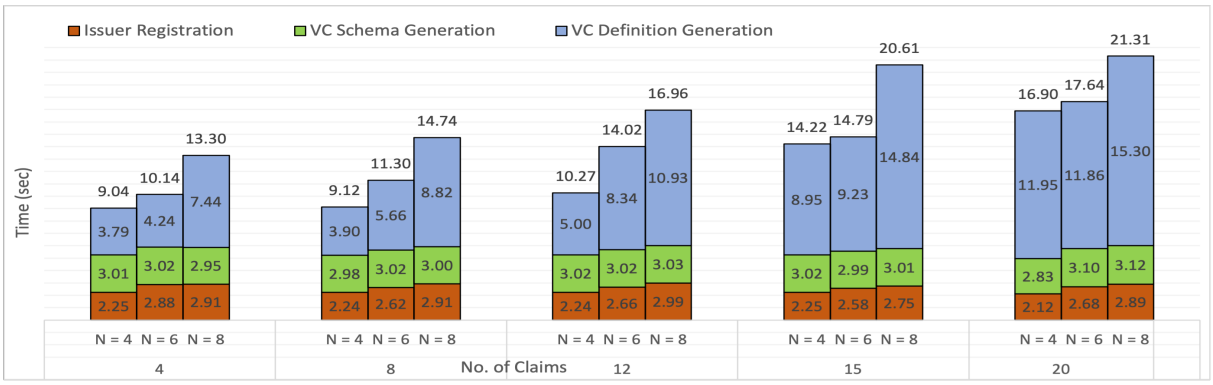


Fig. 6.4 System Setup Time with varying number of Indy Nodes(N) and number of claims(C)

and with vertical scaling, performance can be improved. Here, it is important to note that, N represents the number of Indy nodes which is part of the decentralized identity infrastructure and it does not represent other blockchain nodes that may be employed to perform business logic in a data marketplace [159].

Once the system is set up and the issuer(*SL*) is ready to issue credentials(VC), the time taken for VC creation and henceforth verification is observed. As shown in Figure 6.5, the time for VC creation(C) and verification(V) increases with no. of claims. And it also rises with the number of nodes for the same reason as described above. However, even with the number of claims being as high as 20, the value remains considerably low, particularly for VC verification. It is to be noted that while VC issuance happens once for that VC, verification will happen multiple times so it is an important parameter to understand the system speed.

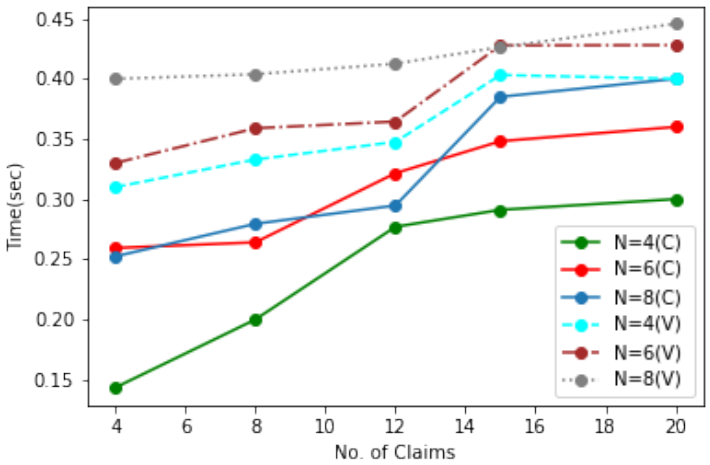


Fig. 6.5 Time taken for VC Creation/Verification

Table 6.4 Storage Cost: *SL*/*Buyer* Wallet [C = No. of claims]

For *SL* After	**C=4**	**C=8**	**C=12**	**C=15**	**C=20**
Wallet Creation(B)	258	258	258	258	258
VC Schema Gen.(B)	580	589	633	633	640
VC Definition Gen.(KB)	6.3	8.2	10.7	12.5	15.6
VC creation(KB)	15.9	20.2	26.2	30.4	34.4
For *Buyer* After					
Wallet Creation(B)	261	230	230	230	230
VC Receive(KB)	22.4	28.4	36.7	42.8	51.7

To understand the storage cost of the *SL* and $Buyer_i$ wallets as the VCs are issued, the values in Table 6.4 are observed. The table represents the values of the wallet storage cost of the seller and buyer at each stage of VC computation. It is observed that the storage cost for *SL* after VC Schema generation and VC definition increases steadily with an increase in the number of claims however it remains in the order of bytes and KB respectively. Similarly, the wallet storage cost for *SL*, increases with an increase in the number of claims and remains in the order of KB. The storage cost for *buyer* on receiving VC grows very reasonably with the number of atomic claims in VC. Even with 20 claims the storage cost remains in the order of KB. A small variation in the $buyer's$ wallet creation cost can be seen in the first two values as it was averaged over a run of multiple trails and does not indicate any significant deviation in the result.

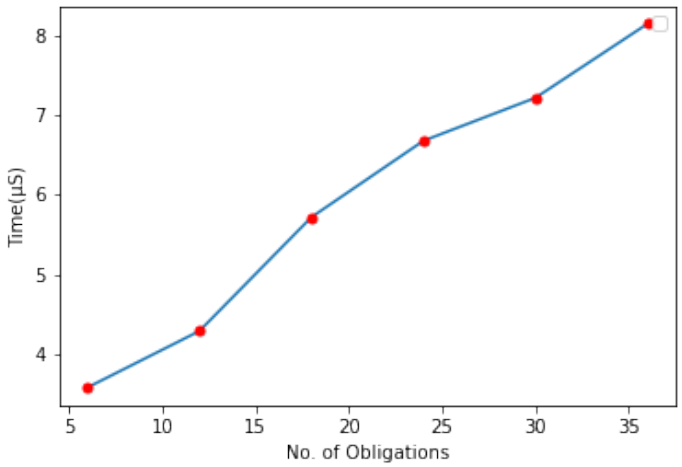


Fig. 6.6 Time taken for usage-controlled policy evaluation

Authentication and Usage-Controlled access phase: While the setup and registration process are executed only once, the authentication and usage-controlled phase will be

executed each time the $Buyer_i$ establishes a new session with SO_j. The total Computation Cost for the authentication phase as calculated during the implementation is shown in Table 6.5. While the time taken for VC computation during registration are dependent on number of claims in VC generation, others are constant and depend on the type of operations. Since the time taken by XOR and concatenation operation is negligible, they are not considered in evaluating computation cost during authentication. The $Buyer_i$ incurs negligible cost during the registration phase and hence not shown in this table. During the authentication, the $Buyer_i$'s device executes $Verify_{EdDSA}$ + $Prove_{ZKP}$ operations which equals to 18.27 ms. While the *SO* executes operation $Verify_{EdDSA}$+$Sign_{EdDSA}$ during the registration phase which equals to 24.96 ms. For verification of presented VC, the verifying party (*SL*/*SO*) performs the operation $Verify_{ZKP}$ and it only incurs 1.1 ms of time.

Figure 6.6 illustrates the computation cost incurred during the usage-controlled access phase as a function of a number of obligations. Figure 6.6 was plotted by observing the time taken for resource consumption of a constant file size with varying numbers of ongoing obligations to check during the access phase. The computation cost observed during authentication in Table 6.5 and time taken for usage-policy evaluation in Figure 6.6 are in the order of *ms* and µs and hence feasible in a real-world use case with further operational optimization.

6.5 Security Analysis

In this section, the proposed protocol is thoroughly analysed in terms of security. A formal security verification is performed using the broadly accepted automated software validation tool, known as Scyther tool [180]. A security analysis is also carried out by intuitive reasoning through the non-mathematical (heuristic) approaches to discuss the most common security threats and vulnerabilities in authentication and access management protocols and how the proposed work overcomes them in a decentralized setting such as a data marketplace.

Table 6.5 Computation Cost Independent of VC setup

Phase	**Buyer**	**SL/SO**
Register	-	$Verify_{EdDSA}$+ $Sign_{EdDSA}$ = 24.98 ms
Authenticate	$Verify_{EdDSA}$+ $Prove_{ZKP}$ = 18.27 ms	$Verify_{ZKP}$ = 1.1 ms
$Sign_{EdDSA}$ = 10.28 ms; $Verify_{EdDSA}$ = 14.7 ms		
$Prove_{ZKP}$ = 3.57 ms; $Verify_{ZKP}$ = 1.1 ms		

```
role Sx{
fresh  n, ki, payreq, nnew, cred, VCi, wi : Nonce ;
var DIDb, PDIDb, md, phiid, reqid,  Ni, payid  : Nonce;

recv_1(Bi,Sx, {Bi, DIDb, PDIDb, md, phiid, reqid}
pk(Sx) );
fresh hV: Response;
send_2(Sx, Bi, {{DIDb, n, ki} sk(Sx), payreq, hV} pk(Bi),
hash(hV) );
var hV2: Response;
recv_3(Bi, Sx, {{ki, Ni} sk(Bi), hV2, payid } pk(Sx),
hash(hV2)  );
send_4(Sx, Bi, { {VCi} sk(Sx), cred, xor(ki, nnew), wi }
pk(Bi) );

claim(Sx, Running, Bi, nnew, ki);
claim_S1(Sx, Alive);
claim_S2(Sx, Weakagree);
claim_S3(Sx, Secret, Ni);
claim_S4(Sx, Niagree);
claim_S5(Sx, Nisynch);}
```

Fig. 6.7 Role for Seller *Sx* in SPDL

6.5.1 Formal Security Verification: A simulation study using Scyther tool

In this section, the proposed protocol's resilience to different attack vectors in authentication protocol is assessed using the widely accepted automated software validation tool, Scyther tool. Through the simulation study using this tool, it is demonstrated that the proposed scheme is safe against attacks such as impersonation, passive secret disclosure, man-in-the-middle and replay attacks and hence provides strong authentication.

Scyther [180] is a security tool which can be used for verification, falsification and analysis of the security protocols. It uses a pattern refinement algorithm to produce an infinite set of traces. The Scyther framework allows to model adversaries in security protocol analysis, varying between Dolev-Yao style adversaries to other more powerful adversaries. Also, it supports security notions such as key compromise, impersonation, weak perfect forward secrecy and adversaries capable of state-reveal queries [180]. The protocol to be verified is provided to the scyther tool in the form of protocol description written using the "Security Protocol Description Language" (SPDL). The protocol description comprises a set of roles. Each role consists of a sequence of events. The events can be send or receive of terms (security parameters).

```
role Bi
{
fresh DIDb,PDIDb,md,phiid,reqid,Ni,payid,PDIDi,usagereq,
vcreq,wi: Nonce;
var payreq,n,ki,cred,nnew,VCi,DIDso,VCso,proofreq: Nonce;

send_1(Bi,Sx,{Bi, DIDb, PDIDb, md, phiid,reqid} pk(Sx));

var hV: Response;
recv_2(Sx,Bi,{{DIDb, n, ki}sk(Sx),payreq, hV}pk(Bi),
hash(hV));

fresh hV2: Response;
send_3(Bi, Sx, {{ki, Ni} sk(Bi), hV2, payid} pk(Sx),
hash(hV2) );
recv_4(Sx,Bi,{{VCi}sk(Sx),cred, xor(ki, nnew),wi}pk(Bi));
send_5(Bi,So,{Bi, DIDb, usagereq, vcreq}pk(So));
recv_6(So,Bi,{{DIDb, DIDso, VCso}sk(So),proofreq}
pk(Bi));
fresh pi, hVcred: Response;
send_7(Bi,So,{{pi, DIDso,PDIDi,hVcred }sk(Bi),wi}pk(So),
hVcred);

claim(Bi, Running, Sx, nnew, ki);
claim_B1(Bi, Alive);
claim_B2(Bi, Weakagree);
claim_B3(Bi, Secret, ki);
claim_B4(Bi, Secret, n);
claim_B5(Bi, Niagree);
claim_B6(Bi, Nisynch);}
```

Fig. 6.8 Role for Buyer *Bi* in SPDL

```
role So{
fresh DIDso,proofreq,VCso: Nonce ;
var DIDb,PDIDi,usagereq,vcreq,wi: Nonce;

recv_5(Bi, So, {Bi, DIDb, usagereq, vcreq} pk(So) );
send_6(So, Bi, {{DIDb, DIDso, VCso} sk(So), proofreq}
pk(Bi) );
var pi, hVcred: Response;
recv_7(Bi, So,{{pi, DIDso, PDIDi,hVcred}sk(Bi),wi}
pk(So),hVcred) ;

claim_So1(So, Alive);
claim_So2(So, Weakagree);
claim_So3(So, Secret, pi);
claim_So4(So, Niagree);
claim_So5(So, Nisynch);}
```

Fig. 6.9 Role for Storage Operator *So* in SPDL

The actors *Seller* (Sx), *Buyer* (Bi) and *StorageOperator* (So) are communicating with one another in the proposed protocol. They are modeled as roles (Sx) , (Bi) and (So) as shown in Figures 6.7, 6.8 and 6.9 respectively. A role begins with the declaration of the sending and receiving terms, then the exchange of such terms followed by the security claims. The security claims are used to model the protocol's security properties. These claims are crucial part of the protocol design without with Scyther tool would not know what is to be verified. The output Table 6.6 confirms that the roles *Sx*, *Bi* and *So* are reachable. This ensures that non-reachability does not lead to any obvious weakness in the protocol description.

Our claims on role *Sx* include (i) aliveness, (ii) weak agreement, (iii) secrecy of nonce *Ni*, (iv) non-injective agreement and (v) non-injective synchronisation. Secondly, our claims on role *Bi* comprise (i) aliveness, (ii) weak agreement, (iii) secrecy of key *ki*, (iv) secrecy of nonce *n*, (v) non-injective agreement and (vi) non-injective synchronisation. Finally, our claims on role *So* are (i) aliveness, (ii) weak agreement, (iii) secrecy of proof *pi*, (iv) non-injective agreement and (v) non-injective synchronisation.

From the scyther verification results in Table 6.7, useful insights can be derived. Firstly, the protocol guarantees the secrecy of nonce *n* and secret key *ki* generated by the seller to establish secure communication with the buyer to complete its registration. Similarly, the secrecy of nonce *Ni* is assured hence verifying the transmission of payment details from buyer to seller is guaranteed. Since similar steps are used for the storage operator registration, hence, secrecy of similar protocol elements between *SL* and *SO* is assured.

Table 6.6 Scyther Results: Verification of reachability of the roles for *Sx*, *Bi* and *So*

Scyther results: characterize				
Claim			**Status**	
Proposed	Sx	Reachable	Ok	Verified
	Bi	Reachable	Ok	Verified
	So	Reachable	Ok	Verified

With these guarantees impersonation and passive secret disclosure attacks are prevented. The claims (Sx, Running, Bi, nnew, ki) and (Bi, Running, nnew, ki) defined under the roles of *Sx* and *Bi* are used in order to specify data agreement between *Sx* and *Bi*. The claim *Weak agreement* is essential to check if the authentication is successful. The other two claims *non-injective agreement* and *non-injective synchronisation* claims are used to verify if our protocol is protected against replay and man-in-the-middle attacks.

6.5.2 Informal Security Analysis

In this section, the proposed protocol is analysed against various attack vectors by intuitive reasoning. It is worth noticing that the informal (non-mathematical heuristic) security analysis is performed to show the proposed protocol is secure against other attacks that are not all covered in the previous subsection.

Credential Theft/Stuffing: Attackers often steal credentials to gain unauthorized access in traditional centralized systems. They gain access to authentication servers to steal user credentials [17]. This is possible as identifiers and credentials of users are stored in a single verifying repository. In the proposed protocol, the concept of DIDs and VCs is leveraged that enable decentralized identity and credential management. The lifecycle of DID is pinned in the underlying identity blockchain which is essentially a decentralized database of these records. It is fault-tolerant, enables efficient revocation and is secure to credential stuffing. The *Buyer* create and manage their own IDs and do not depend on the marketplace to maintain them. The users are empowered by the fact they have complete control of their IDs. While in the proposed protocol, the credentials are still susceptible to be stolen/leaked, by not having a centralized repository it becomes less attractive to the attacker and at times not worthy of amount of the effort required.

Inference Attack: Inference attacks are possible by mining information available about patterns of a user on any platform. By drawing information from their digital footprint, user activity can be linked and tracked in a marketplace. This leads to loss of privacy and unsolicited marketing communications. In the proposed protocol, there are two safeguards against the possibility of an inference attack. Firstly, the *buyer* always interacts with other actors in the marketplace using a pseudo-ID $PDID_i$. For two successive sessions x and

Scyther results: verify					
Claim				**Status**	**Comments**
datamarketplace	*Sx*	Proposed,S1	Alive	Ok	No attacks.
		Proposed,S2	Weakagree	Ok	No attacks.
		Proposed,S3	Secret *Ni*	Ok	No attacks.
		Proposed,S4	Niagree	Ok	No attacks.
		Proposed,S5	Nisynch	Ok	No attacks.

(a) Verifying *Sx*'s claims

Scyther results: verify					
Claim				**Status**	**Comments**
datamarketplace	*Bi*	Proposed,B1	Alive	Ok	No attacks.
		Proposed,B2	Weakagree	Ok	No attacks.
		Proposed,B3	Secret *ki*	Ok	No attacks.
		Proposed,B4	Secret *n*	Ok	No attacks.
		Proposed,B5	Niagree	Ok	No attacks.
		Proposed,B6	Nisynch	Ok	No attacks.

(b) Verifying *Bi*'s claims

Scyther results: verify					
Claim				**Status**	**Comments**
datamarketplace	*So*	Proposed,So1	Alive	Ok	No attacks.
		Proposed,So2	Weakagree	Ok	No attacks.
		Proposed,So3	Secret *pi*	Ok	No attacks.
		Proposed,So4	Niagree	Ok	No attacks.
		Proposed,So5	Nisynch	Ok	No attacks.

(c) Verifying *So*'s claims

Table 6.7 Scyther results: verification of claims

$x+1$, $PDID_i^x \neq PDID_i^{x+1}$. thus the identities of the users are unlinkable. Secondly, the *Buyer* provides a ZKP of the VC to the *SO*. The *SO* cannot learn anything about the $Buyer_i$ from the ZKP, and only know the claims against which policy evaluation has to be done. Thus, an adversary will be unable to link $Buyer's$ real identities with their pseudo-IDs or VC.

Reflection Attack: A reflection attack occurs when an attacker tricks a legitimate user into performing an action by reflecting the user's own request back to them. To prevent this attack *SL* sends a nonce *n* in $M_{REG2} : \{hV, Pay_{req}, Ki, n\}$ to the *buyer* when requesting to make payment. Later, after completing the payment, *buyer* uses the unique nonce *n* shared by *SL* to compute $hV2 = h(payment_{ID} \| K_i \| n \| N_i)$ in M_{REG2} .

Reply Attack: An adversary may intercept and retransmit messages exchanged between sender and receiver to get information(payment/data). To prevent it, during *buyer* registration when the buyer shares the completed payment details with *SL*, in M_{REG3}: $\{hV2, payment_{ID}, N_i\}$, a nonce N_i is included in the message to ensure that the message is unique. It will allow the *SL* to verify whether the nonce has been used before. If the nonce has already been used, the message is rejected, preventing a malicious actor from registering with the *SL*.

Man-in-the-middle Attack: In order to prevent MITM attacks, several defence mechanisms have been provided in the protocol. Firstly, strong authentication guarantees are provided by the usage of VCs. The *SO* presents its VC to $Buyer_i$ which $Buyer_i$ verifies. Then, the *Buyer* presents the ZKP of its VC signed and issued by *SL* to the *SO* ad the *SO* verifies it. Only legitimate buyers and legitimate SOs hold the valid VCs, signed with the private key of the *SL*, K_{SL}^{pr}. Therefore, mutual authentication between actors makes it difficult for an impersonator to replicate the messages/requests. Secondly, since the sellers and buyers in the network possess unique *DID*, secured by a public-private key pair, an attacker cannot inject malicious signed messages.

Sybil Attack: In Sybil attack, a malicious user creates multiple fake identities to gain control over a significant portion of the network. To prevent this attack there are two security guarantees provided in the protocol. Firstly, on the *SL* level, before a buyer can register with a seller, it needs to present $[VC_{DigIDi}]_{K_{MC}^{pr}}$, the VC of their digital identity issued by the marketplace consortium after eKYC verification. This provides the first safeguard against the possibility of creating multiple fake buyer profiles on the marketplace. Secondly, on the *SO* level, a dynamic buyer revocation is proposed using a cryptographic accumulator. The marketplace maintains a revocation list and buyers must prove to *SO* before starting access that they are not members of this list by providing a non-membership witness issued by SL.

Privilege Abuse and Escalation: Excessive privilege in access control systems can lead to its abuse and escalated threats. Authorizing least-privilege access helps in putting a check

on malicious actors who can harm the system if they get in maliciously. It ensures that users have only the required level of permissions, for the required duration and under required conditions in order to perform access. Usage-based access control performs ongoing along with pre-evaluation to determine the applicability of the right to access a certain resource. By limiting access under specific *authorization* policy governed by *obligations* and *conditions*, the attack surface (potential entry points for attackers) is reduced. In the proposed marketplace protocol, if any malicious actor gains access to a data stream hosted on a *storage operator* node, the checks in place on an ongoing basis will ensure to stop access effectively and efficiently.

6.6 Conclusion

In this work, a user-empowered, privacy-aware, authentication and usage-controlled access protocol for the IIoT data marketplace is proposed. The proposed protocol addresses the issue of secure and privacy-aware authentication in real-time decentralized trading of IoT data which is a future reality owing to the massive progress in advanced AI algorithms. Along with presenting a secure and privacy-aware mutual authentication using DIDs and VCs, the protocol also implements usage-controlled access of data streams being accessed by the *buyer* from *storage operator*. The protocol is implemented to test its feasibility in terms of computation and storage performance. It can be observed that although the setup of such a protocol can be a bit expensive for the first time, its successive use for authentication and access control is computationally inexpensive. In addition, the security guarantees of using decentralized identifiers and credentials that can support privacy-preserving data trading support such a protocol.

Chapter 7

Conclusion and Future Work

This is the final chapter of this dissertation. It presents the conclusion of the work presented in this dissertation. The novel contribution of this dissertation is summarized with details on how they were achieved. Finally, this chapter explores the future work directions that arise as improvements over the work done as part of this dissertation.

In spite of the extensive use of the Internet-of-Things (IoT) and its indisputable advantages, multiple challenges still remain open. Clearly, some aspects of the IoT require further research to reach their full potential. One of those aspects is to overcome the barriers of boundaries, platforms and silos in terms of communication and data exchange. IoT services are undergoing a massive overhaul in terms of their operational pattern and real-world use cases. With this appears unique forms of threats that require a new approach to problem-solving. Without it, the applicability of the IoT for particular scenarios would remain incomplete.

This dissertation aims to explore the challenges in IoT networks like scalability, interoperability, security and ultimately centralized nature of operations in identity and access management. Due to the large scope of this research area, this dissertation concentrates on achieving this in four specific categories. The first category relates to understanding and exploring state-of-the-art solutions in terms of how identities in IoT networks are assigned, managed, utilised and maintained over their life cycle. Despite the fact that solutions for these operations already exist in the Internet domain, the limitations of the IoT technology raise new challenges, making most of the existing solutions not fully compatible with the IoT scenarios. Having new solutions that consider unique IoT limitations and operation patterns would increase the connectivity of IoT technologies to the extent that devices belonging to different infrastructures could communicate indistinguishably with each other. In accordance with these limitations, we implemented a platform-agnostic mechanism that enables the communication and connectivity of IoT devices and machines beyond identity silos.

The second category consists of achieving a mechanism to uniquely identify IoT devices and assign digitally verifiable attributes linked to these identifiers for secure communication. With current IoT device identifiers, several challenges exist such as reduplication, repeated IDs, weak passwords etc which result in weak authentication and authorization mechanisms. Honeypot databases retaining identity data and common root of trust to derive identifiers impede scalability and are potential attack vectors. Hence, we design and implement a decentralized identity-managing framework that is capable of managing a large scale of IoT devices. The framework implements a unique way of managing identity and attributes to prevent the formation of a single large collection of data resources.

The third category addresses the issue of fragmented IoT data silos held by individual organizations barring them from re-using and sharing the data for the fear of security, privacy, lack of transparency and unfair payment settlement. The majority of data-selling platforms are under the control of large multinational corporations with no transparency between buyer and seller in terms of payment details, listing, data discovery and storage. Siloed identity verticals, unauthorized malicious entities posing as data producers, centralized control of the platform and fear around equitable payment settlement are challenges that have been addressed in this category. To this end, we designed and implemented a decentralized data marketplace for trading IoT data that relies on blockchain technology to ensure fairness in trading activities, trust in the network and secure exchange of IoT data across different silos.

Finally, the fourth category describes the challenges associated with static authentication techniques and the lack of granularity in access control mechanisms in a multi-stakeholder environment. Traditional authentication and authorization techniques are designed for a homogeneous single-owner environment in which subjects from the same domain trust each other and share resources. Additionally, the traditional authentication and access control mechanisms are one-time off and only provide security at the point of entry. Extended resource access monitoring is crucial to provide secure access to shared IoT resources.

In summary, we provide solutions to all four categories in terms of enhancing the identity and access management techniques in modern-day IoT use cases that are more decentralized, and distributed in terms of shared control of resources and demand scalable solutions. Moreover, all our proposed solutions have been thoroughly evaluated and, in some cases, compared with the traditional state-of-the-art solutions, thereby showing their feasibility.

7.1 Contributions

In Section 1.5, an overview of the contributions made in this dissertation is presented. In this Section, we further elaborate on the contributions made and the novel work achieved through each of them. Therefore, the following contributions have been made to this dissertation:

- **A novel smart-contract enabled framework for identity and access management of distributed IoT networks** [25]: This work describes and implements a novel decentralized framework for managing identity and access management policies for IoT. The proposed framework is a new blockchain smart contract-enabled and decentralized life-cycle and access management system. It leverages the peer-to-peer, content-addressable distributed file system known as InterPlanatery FileSystem (IPFS) as a distributed registry of device resources. These concepts enhance the scalability and throughput of the system. A framework is implemented and experiments are performed. This concept performed significantly well in comparison to the centralized state-of-the-art solutions, achieving a throughput of 930 requests served per second by a single PTG with a latency of 3 msec at MQTT Q=1. The effect of This work is the first to consider a parent-child smart-contract-based cascading call design for identity and access management in IoT.

- **A decentralized digital identity framework extended using self-sovereign identity model** [26]: This work improves the previously proposed scheme. The learning from previous work is mapped to the recent standard of W3C's Self-Sovereign Identity (SSI). In this work, the distributed ledger technology is used as an enabler of portability, interoperability and secure verification between the subjects. The standards of decentralized identifiers (DIDs) and verifiable credentials (VCs) in the context of IoT are introduced in this work as they significantly improve the management of device identity and its associated credentials. The novel technique of creating and verifying cryptographically verifiable device identities and credentials is achieved in the order of 0.2 sec and 0.3 sec respectively. The solution is shown to overcome state-of-the-art challenges associated with scalability, interoperability and portability of device credentials.

- **A fair, secure and trusted decentralized data marketplace enabled by blockchain** [22]: A novel platform is designed and developed integrating the concepts from the previous contributions. The proposed scheme designs a decentralized digital IoT data marketplace that overcomes the challenge of obtaining high-quality data for novel use cases like research and training AI models etc. The platform leverages a decentralized data streaming network to host IoT data in a reliable and fault-tolerant

manner. The platform ensures fair trading, data storage and delivery in a privacy-preserving manner and trust-metric calculation for actors in the network. To study the feasibility of the proposed platform, an open-source library is developed using Hyperledger Fabric and a data network layer built on VerneMQ. The concept of trust in a decentralized marketplace was studied and how it can be quantified for such a marketplace was explored. The library is tested and results are analysed for throughput, overheads and scalability.

- **A Privacy-Aware Authentication and Usage-Controlled Access Protocol for IIoT Decentralized Data Marketplace**: While working on the previous contribution, areas of improvement were noted in collaborative IoT operations such as the IoT data marketplace. This scheme integrates learning from previous schemes to propose a novel architecture for a context-aware continuous authentication and usage control system for a collaborative IoT ecosystem. This scheme takes into account the challenges of legacy security models that are not designed to account for granular access control levels and fail to detect real-time threats in such collaborative spaces. The platform ensures real-time access monitoring and instant revocation in case of malicious or unauthorized behaviour. The platform leverages a decentralized identity infrastructure for issuing credentials to devices in a decentralized and fault-tolerant manner. To study the feasibility of the proposed platform, an open-source library is developed using Hyperledger Indy and Linux Utility-Cron Job to design a UCON system. The library is tested and results are analysed for time and storage cost, overheads and scalability. This is the first use of VCs and usage control for the purpose of continuous authentication.

7.2 Key Insights

At the beginning of this dissertation, we outlined the key problem statements that the research contribution should target in order to facilitate its easy adoption. All these features have been implemented by all the solutions defined in this dissertation.
Fragmented Identity Silos plays a crucial motivation in all the proposed solutions. Almost all IoT enterprises follow a set nomenclature strategy based on their characteristic requirement however, this acts as a constraint for cross-domain communication. Since device identifiers are only valid in their unique environment, it renders them non-useful in other premises. These fragmented identity silos hold back the overall potential of the Internet of Things. This challenge is recognized in this dissertation and addressed specifically in Publication I and Publication IV. Proposed solutions in these publications address this issue and design solutions that relinquish vertical fragmented identity silos

between communicating parties. As a result, IoT devices have a decentralized root of trust to exchange and validate communication requests. To the extent possible, the specific operations related to the constrained nature of such devices have remained unchanged in our systems.

Along the same lines, the proposed solutions also needed to focus on *issues with device identifiers*. This key requirement is linked to the fact that default username-password combinations, weak passwords and non-standard identifiers for IoT devices are the root cause of a large proportion of attacks on IoT networks. Globally unique identifiers are favourable for both scalability issues and security challenges that ultimately lead to stronger authentication guarantees. Similar to the previous case, the unique identifier characteristic in our designed systems has been achieved by not modifying IoT devices. Consequently, the proposed solutions in this dissertation place the minimum possible load on the IoT devices which has been reflected in the presented results. Publications II and IV specifically present solutions to this particular challenge in detail.

Honeypot Databases in IDAMS are a result of siloed identity verticals that exist in IoT industry today. Various IoT solutions hosted on clouds today maintain a centralized directory of identifiers and other sensitive information related to IoT devices onboarded in the network. These databases of sensitive device information are lucrative attack targets for malicious entities. If hacked, these directories can potentially harm the security and privacy of businesses. With the projected number of IoT devices estimated to be in the range of billions, there is no doubt that this challenge will only augment. Therefore, this dissertation pays high importance to design decentralized IoT device identity management solution that forgo centralized ID databases. With that goal in mind, particular attention was paid to the design of our solutions to provide an adequate level of decentralization. We leverage distributed ledger technology and model it for specific IoT scenarios. Along with that experimental evaluations conducted in this dissertation confirm that our solutions can support large numbers of IoT devices. The majority of our solutions have been designed to support decentralization, horizontal along with vertical scaling and scalability to avoid the formation of honeypot repositories.

Interoperability is the core concept of this dissertation. Right from the first contribution to the last one, interoperability is demonstrated in each contribution, particularly in Publication II, Publication III and Publication IV. IoT by nature is a distributed technology where each element in the system plays a distinct but specific role. Interoperability can fundamentally support fault tolerance and availability for this technology. Interoperability can address some of the new challenges that the evolving IoT landscape presents, some of which have been addressed in this dissertation; communication/coordination methods for globally managing resources related to identity and access information (addressed in Publication II), authorization methods performed on a global scale for data trading (Publication

III), and methods to share digital device attributes globally and unambiguously in case of a shared and continuous monitoring environment (Publication IV). This dissertation provides solutions to the interoperability-related challenges that we have considered to be relevant to build up the future of IoT.

Scalability is an advantageous characteristic of IoT networks, particularly in IIoT, where the scale of instalments is massive as compared to home IoT instalments. With further growth predicted, there is no doubt about the high importance of creating identity management solutions that scale well with number of IoT devices. With that as the motivation, IAM scalability is of particular interest in this work. In a technology like IoT where ad-hock communications are frequently established, ensuring throughput with rising connection requests is extremely critical. The most notable case is the solution presented in Publications I, III and IV. Publication I addresses this in terms of managing simultaneous access requests of shared resources and studying the behaviour of the framework with an increase in the load. Publication III extrapolates the previous work on a wider scale with a platform acting as an intermediary to divert these data access requests. Publication IV addresses and explores the scalability in a distributed system where granular monitoring of access requests is carried out in a continuous mode.

A new set of *Peer-to-Peer Services* in the IoT domain are emerging such as point-to-point vehicular communication for payments, charging, data exchange and authentication-based services. New technologies such as IoT relay beacons, robotic machines, and connected vehicles, which are developed under the umbrella of IoT engage in minimal but frequent data exchange. Therefore, such services exchange in frequent and often unbounded (in terms of security boundary) exchange of authentication credentials, thus portability and interoperability of authentication mechanisms are essential. In Publication III and Publication IV, such peer-to-peer applications are discussed when IoT devices interact with other devices or online services to exchange data.

Lastly, *Access Control* in IoT networks is a critical part of device management and ensures the safety and security of the network. A heterogeneous IoT system that can handle a large number of simultaneous requests for resource access is essential. The traditional access control methodologies originally designed for the Internet made their way to IoT, hence they were either too verbose or too heavy for IoT devices. Also, one-time access control techniques like Access Control Lists (ACLs) are susceptible to insider attacks. Granular access control with continuous monitoring of access rights as a function of usage performed can vastly enhance the security of the systems by stopping malicious actors from causing large-scale damage at the first hint of suspicious activity. On this basis, the solution described in Publication IV takes into account the nature of continuous and attribute-based access control of resources.

In conclusion, this dissertation focuses on building solutions with all these characteristics in mind. In this respect, all of these characteristics have undoubtedly contributed to creating decentralized and scalable solutions in the four contributions categories described in this chapter.

7.3 Future Work Directions

After studying and researching about various drawbacks of centralized identity and access management techniques for resource constrained devices in various setup, this dissertation explored the prospect of designing a decentralized, p2p architectural framework for such devices. In the process, we explored the use of distributed ledger technology for identity and access management for IoT devices. We also expanded the use of Self-Sovereign Identity (SSI) for IoT devices by integrating them in novel contemporary use-cases. IoT data ownership has long been a challenge and with IoT data being traded, it has given new opportunity to device owners to receive economic incentives. Subsequently, this dissertation explores the access control challenges in a collaborative multi stakeholder IoT environment. Network boundaries overlap and frequent switching of networks should not bar secure authentication and access control techniques.

While this dissertation touches on aforementioned areas, there are still topics worth discussing beyond the focus of this dissertation. In particular, this dissertation mainly focuses on the following research areas: decentralized identity management, decentralized and granular access management, a decentralised data trading platform and the scalability of these proposed solutions. Altogether, these aspects are of vital importance for the modern development of IoT. Without proper solutions to the challenges represented in these categories, the IoT would inevitably face hindrance in its adoption rate.

However, there are other major challenges in the IoT landscape. The following describes the future work that would further enhance the contributions of this thesis:

- Interoperability among various distributed ledger technologies to support a truly decentralized ecosystem that incorporates various functionalities like payment, development of oracle smart-contracts, trust metric calculation and digital identity. For e.g., a DLT for identity platform can communicate with another that supports a data marketplace and thus can settle payments after verification of identities.

- Lightweight VCs for IoT devices can be another potential future work. VCs used in this dissertation are suitable for IoT devices that can support medium code footprint, however for very resource constraint IoT devices lightweight VCs will be appropriate. They can ensure that a large range of IoT devices can be included in the network that can authenticate via VCs issued to them.

- Another area that can be taken up as future work is designing equitable payment methodologies between the two parties in a decentralized data marketplace presented in this dissertation. The parties can agree to transact in either cryptocurrencies/native tokens or existing payment methods. Crypto-tokens can also act as an incentive mechanism for early adopters of the platform. Designing a payment layer over such a system architecture can be one of the future works. Asset tokenization is a fast growing field and can see adoption in fabrication and industrial spaces.

- Yet another challenge in a data marketplace is to put a check on the external sharing of platform IoT data in the long run. For this, game theory approaches can be taken to design, better pricing models and terms of data exchange. In the proposed work, we have proposed a basic incentivization mechanism, in future works, more robust and theoretically tested schemes can be designed to make unfair transactions economically unattractive.

- In the last piece of work in this dissertation we propose some future work directions. One challenge that can be taken up as a potential future work is to devise a self-learning algorithm that can be based on machine learning to better understand the functionalities of different mutable attributes and how they affect the outcome of authorization. This will help design better continuous authorization policies. Yet another direction of work can be developing as a federated continuous authentication model that takes into account the capabilities of different types of IoT devices and sensors to authenticate subjects in real-time.

In general, we believe that all these directions represent potential points of improvement for IoT's further development and will help IoT keep pace with the new emerging technological challenges. All of them are recommended for future research.

References

[1] Energy IoT. https://www.energyiot.co.th. [Online: Accessed 15-March-2021].

[2] Cisco IoT. https://www.cisco.com/c/en_in/solutions/internet-of-things/overview.html. [Online: Accessed 15-March-2021].

[3] Philips Innovation. https://www.innovationservices.philips.com/looking-expertise/electronic-systems-iot/. [Online: Accessed 15-March-2021].

[4] Thinger.IO. https://www.thinger.io. [Online: Accessed 15-March-2021].

[5] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou. Understanding the mirai botnet. In *Proceedings of the 26th USENIX Conference on Security Symposium*, page 1093–1110. USENIX Association, 2017.

[6] Eranga Bandara, Xueping Liang, Peter Foytik, Sachin Shetty, and Kasun De Zoysa. A blockchain and self-sovereign identity empowered digital identity platform. In *2021 International Conference on Computer Communications and Networks (ICCCN)*, pages 1–7, 2021.

[7] Bruno Cremonezi, Alex B. Vieira, José Nacif, Edelberto Franco Silva, and Michele Nogueira. Identity management for internet of things: Concepts, challenges and opportunities. *Computer Communications*, 224:72–94, 2024.

[8] H. Choi, J. Song, and K. Yi. Brightics-iot: Towards effective industrial iot platforms for connected smart factories. In *2018 IEEE International Conference on Industrial Internet (ICII)*, pages 146–152, 2018.

[9] Towards a data economy: An enabling framework. [Online]: https://www.weforum.org/whitepapers/towards-a-data-economy-an-enabling-framework. [Accessed: Jul-2023].

[10] J. Zhang, Y. Yang, X. Liu, and J. Ma. An efficient blockchain-based hierarchical data sharing for healthcare internet of things. *IEEE Transactions on Industrial Informatics*, 18(10):7139–7150, 2022.

[11] "to monetize and acquire iot data". [Online]. Available: https://www.dawex.com/en/monetization-data-iot/. [Accessed: 04-March-2021].

[12] Data: The New Asset Class. https://oceanprotocol.com/. [Online: Accessed Sept-2023].

[13] M. S. Bargh, M. Vink, and S. Choenni. On using obligations for usage control in joining of datasets. In *Information Systems Security and Privacy*, pages 173–196, Cham, 2018. Springer International Publishing.

[14] "Inside the infamous Mirai IoT Botnet: A retrospective analysis," Dec 2017. https://blog.cloudflare.com/inside-mirai-the-infamous-iot-botnet-a-retrospective-analysis/. [Online: Accessed 15-March-2021].

[15] D. Khosrowshahi. "2016 data security incident". [Online]. Available: https://www.uber.com/newsroom/2016-data-incident/. [Accessed: 21-Dec-2021].

[16] Xiaoyang Zhu and Youakim Badr. Identity management systems for the internet of things: A survey towards blockchain solutions. *Sensors*, 18(12), 2018.

[17] "cybersecurity incident - information for uk consumers". Available: https://www.equifax.co.uk/incident.html. [Accessed: 21-Dec-2021].

[18] Arne Bröring, Stefan Schmid, Corina-Kim Schindhelm, Abdelmajid Khelil, Sebastian Käbisch, Denis Kramer, Danh Le Phuoc, Jelena Mitic, Darko Anicic, and Ernest Teniente. Enabling iot ecosystems through platform interoperability. *IEEE Software*, 34(1):54–61, 2017.

[19] E. Bertin, D. Hussein, C. Sengul, and V. Frey. Access control in the internet of things: a survey of existing approaches and open research questions. *Annals of Telecommunications*, 74(7):375–388, Aug 2019.

[20] Using role-based access control. https://www.ibm.com/docs/en/mapms/1_cloud?topic=devices-using-role-based-access-control. [Online: Accessed Aug-2023].

[21] J. Park and R. Sandhu. The uconabc usage control model. *ACM transactions on information and system security (TISSEC)*, 2004.

[22] Akanksha Dixit, Arjun Singh, Yogachandran Rahulamathavan, and Muttukrishnan Rajarajan. Fast data: A fair, secure and trusted decentralized iiot data marketplace enabled by blockchain. *IEEE Internet of Things Journal*, pages 1–1, 2021.

[23] A. Dixit. https://github.com/akankshadixit/.

[24] Gordana Dodig-Crnkovic. Scientific methods in computer science. 2002.

[25] Akanksha Dixit, Waqar Asif, and Muttukrishnan Rajarajan. Smart-contract enabled decentralized identity management framework for industry 4.0. In *IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society*, pages 2221–2227, 2020.

[26] Akanksha Dixit, Max Smith-Creasey, and Muttukrishnan Rajarajan. A decentralized iiot identity framework based on self-sovereign identity using blockchain. In *2022 IEEE 47th Conference on Local Computer Networks (LCN)*, pages 335–338, 2022.

[27] Akanksha Dixit, Bruno Bogaz Zarpelao, Max Smith-Creasey, and Muttukrishnan Rajarajan. A privacy-aware authentication and usage-controlled access protocol for iiot decentralized data marketplace. *Computers Security*, 146:104050, 2024.

[28] The "■Only" Coke Machine on the Internet. https://www.cs.cmu.edu/~coke/history_long.txt. [Online: Accessed 15-March-2021].

[29] Jamali M., B. Bahrami, A. Heidari, P. Allahverdizadeh, and F. Norouzi. *IoT Architecture*, pages 9–31. Springer International Publishing, Cham, 2020.

[30] N. M. Kumar and P. K. Mallick. The internet of things: Insights into the building blocks, component interactions, and architecture layers. *Procedia Computer Science*. International Conference on Computational Intelligence and Data Science.

[31] Z. Shelby, K. Hartke, and C. Bormann. The Constrained Application Protocol (CoAP)." RFC 7252, June 2014. https://tools.ietf.org/html/rfc7252. [Online: Accessed 15-March-2021].

[32] U. Hunkeler, H. L. Truong, and A. Stanford-Clark. Mqtt-s — a publish/subscribe protocol for wireless sensor networks. In *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE '08)*, pages 791–798, 2008.

[33] Lightweight machine to machine technical specification: Core v1.1.1," tech. rep., Open Mobile Alliance OMA, June 2019. . [Online: Accessed 15-March-2021].

[34] A. Bhargav-Spantzel, A. C. Squicciarini, and E. Bertino. Establishing and protecting digital identity in federation systems. In *Proceedings of the 2005 Workshop on Digital Identity Management*, DIM '05, page 11–19, New York, NY, USA, 2005. Association for Computing Machinery.

[35] M. D. Guel. A framework for choosing your next generation authentication/authorization system. In *Information Security Technical Report*, volume 7, pages 63–78, 2001.

[36] D. Recordon and D. Reed. Openid 2.0: A platform for user-centric identity management. In *Proceedings of the Second ACM Workshop on Digital Identity Management*, DIM '06, page 11–16, New York, NY, USA, 2006. Association for Computing Machinery.

[37] H.Aftab, K. Gilani, J. Lee, L. Nkenyereye, S. Jeong, and J. Song. Analysis of identifiers in iot platforms. In *Digital Communications and Networks*. Elsevier BV, 2019.

[38] M.Abu-Elkheir, M. Hayajneh, and N. Ali. Data management for the internet of things: Design primitives and solution. page 15582–15612. Sensors, 2013.

[39] Identifiers in Internet of Things (IoT) Version 1.0, AIOTI WG03–IoT Standardisation". Available:https://euagenda.eu/upload/publications/identifiers-in-internet-of-things-iot.pdf. [Online: Accessed 15-March-2021].

[40] ETSI, GS LTN 002 V1.1.1. „ Low Throughput Networks (LTN); Functional Architecture, September 2014. http://www.etsi.org/deliver/etsi_gs/LTN/001_099/002/01.01.01_60/gs_LTN002v010101p.pdf. [Online: Accessed 15-March-2021].

[41] IEEE, IEEE 802.3, Standard for Ethernet", 2015. http://ieeexplore.ieee.org/document/7428776/. [Online: Accessed 15-March-2021].

[42] IETF, RFC 791, Internet Protocol - DARPA Internet Program Protocol Specification“, 1981. https://tools.ietf.org/html/rfc791. [Online: Accessed 15-March-2021].

[43] IETF, RFC 4291, IP Version 6 Addressing Architecture”. Available:https://tools.ietf.org/html/rfc4291. [Online: Accessed 15-March-2021].

[44] D. Hardt. “The OAuth 2.0 Authorization Framework.” RFC 6749, Oct. 2012. Available:https://rfc-editor.org/rfc/rfc6749.txt. [Online: Accessed 15-March-2021].

[45] V. Mladenov, C. Mainka, J. Krautwald, F. Feldmann, and J. Schwenk. “On the security of modern single sign-on protocols: Openid connect 1.0,” CoRR abs/1508.04324, 2015.

[46] L. Seitz, S. Gerdes, G. Selander, M. Mani, and S. Kumar. “Use Cases for Authentication and Authorization in Constrained Environments.” RFC 7744, Jan. 2016. Available:https://rfc-editor.org/rfc/rfc7744.txt. [Online: Accessed 15-March-2021].

[47] S. R. Moosavi, T. N. Gia, E. Nigussie A. M. Rahmani, S. Virtanen, J. Isoaho, and H. Tenhunen. Sea : A secure and efficient authentication and authorization architecture for iot-based healthcare using smart gateways. In *Procedia Computer Science*, volume 52, page 452–459, 2015.

[48] C.-Y. Chang, J.-C. Chen C.-H. Kuo, and T.-C. Wang. Design and implementation of an iot access point for smart home. In *Applied Sciencese*, volume 5, page 1882–1903, 2015.

[49] Mobasshir M. Progressive researches on iot security: An exhaustive analysis from the perspective of protocols, vulnerabilities, and preemptive architectonics. *Journal of Network and Computer Applications*, 168:102761, 2020.

[50] P2P Weakness Exposes Millions of IoT Devices. https://krebsonsecurity.com/2019/04/p2p-weakness-exposes-millions-of-iot-devices/. [Online: Accessed Aug-2023].

[51] K. Cameron. The laws of identity. [Available]: https://www.identityblog.com/stories/2005/05/13/TheLawsOfIdentity.pdf. [Accessed: 21-Dec-2021].

[52] Frederico Schardong and Ricardo Felipe Custódio. Self-sovereign identity: A systematic map and review. *CoRR*, abs/2108.08338, 2021.

[53] Did specification registries. [Available]: https://w3c.github.io/did-spec-registries/#did-methods. [Accessed: 24-Sept-2020].

[54] BTCR DID Method. https://w3c-ccg.github.io/didm-btcr/#continuation. [Online: Accessed Aug-2023].

[55] Sovrin DID Method Specification. https://sovrin-foundation.github.io/sovrin/spec/did-method-spec-template.html. [Online: Accessed Aug-2023].

[56] Decentralized Identity Document (DID Document). https://developer.uport.me/pki/diddocument. [Online: Accessed Aug-2023].

[57] Verifiable credentials use cases. [Online]. Available: https://www.w3.org/TR/vc-use-cases/. [Accessed: 21-Dec-2021].

[58] K. Christidis and M. Devetsikiotis. Blockchains and smart contracts for the internet of things. *IEEE Access*, 4:2292–2303, 2016.

[59] ”ethereum“. [Available]: https://ethereum.org/en/. [Accessed: 21-Dec-2021].

[60] Hyperledger Fabric. https://www.hyperledger.org/projects/fabric. [Online: Accessed Aug-2023].

[61] Satoshi Nakamoto. “Bitcoin: A Peer-to-Peer Electronic Cash System”. Available: http://www.bitcoin.org/bitcoin.pdf,2009. [Online: Accessed 10-Nov-2021].

[62] M. Conti, G. Kumar, P. Nerurkar, R. Saha, and L. Vigneri. A survey on security challenges and solutions in the iota. *Journal of Network and Computer Applications*, 203:103383, 2022.

[63] Hedera. [Available]: https://hedera.com. [Accessed: 21-Dec-2021].

[64] B. K. Mohanta, S. S. Panda, and D. Jena. An overview of smart contract and use cases in blockchain technology. In *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pages 1–4, 2018.

[65] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2009.

[66] T. M. Fernández-Caramés and P. Fraga-Lamas. A review on the use of blockchain for the internet of things. *IEEE Access*, 6:32979–33001, 2018.

[67] V. Buterin. “ethereum: A next-generation smart contract and decentralized application platform,” tech. rep. ethereum foundation, jan 2014. [Accessed: 04-March-2021].

[68] X. Fu, H. Wang, and P. Shi. A survey of blockchain consensus algorithms: mechanism, design and applications. In *Sci. China Inf. Sci.*, volume 64, pages 1–4, 2021.

[69] M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *J. ACM*, 27(2):228–234, apr 1980.

[70] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, jul 1982.

[71] PROOF-OF-STAKE (POS). https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/. [Online: Accessed Aug-2023].

[72] Miguel Castro, Barbara Liskov, et al. Practical byzantine fault tolerance. In *OsDI*, volume 99, pages 173–186, 1999.

[73] G. Wood. “ethereum: A secure decentralised generalised transactionledger,” tech. rep., ethereum foundation, 2014. [Accessed: 04-March-2021].

[74] ”hyperledger indy“. [Available: https://www.hyperledger.org/use/hyperledger-indy. [Accessed: 21-Jan-2022].

[75] Decentralized Storage. https://ethereum.org/en/developers/docs/storage/. [Online: Accessed 25-Nov-2022].

[76] Ipfs powers the distributed web. [Available]: https://docs.ipfs.tech/concepts/. [Accessed: 14-Oct-2020].

[77] ”eddsa and ed25519“. [Online]: https://cryptobook.nakov.com/digital-signatures/eddsa-and-ed25519. [Accessed: Jul-2023].

[78] J. Groth. On the size of pairing-based non-interactive arguments. In *Advances in Cryptology – EUROCRYPT 2016*, pages 305–326, 2016.

[79] J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *Advances in Cryptology — CRYPTO 2002*, pages 61–76, 2002.

[80] J. Li, N. Li, and R. Xue. Universal accumulators with efficient nonmembership proofs. In *Proceedings of the 5th International Conference on Applied Cryptography and Network Security*, page 253–269, 2007.

[81] L. D. Xu, W. He, and S. Li. Internet of things in industries: A survey. *IEEE Transactions on Industrial Informatics*, 10(4):2233–2243, 2014.

[82] H. Choi, J. Song, and K. Yi. Brightics-iot: Towards effective industrial iot platforms for connected smart factories. In *2018 IEEE International Conference on Industrial Internet (ICII)*, pages 146–152, 2018.

[83] B. Bencsáth, G. Pék, L. Buttyán, and M. Félegyházi. Duqu: Analysis, detection, and lessons learned. *ACM European Workshop on System Security (EuroSec)*, 2012.

[84] P. Mahalle, S. Babar, R.N. Prasad, and R. Prasad. Identity management framework towards internet of things (iot): Roadmap and key challenges. In *Re- cent Trends in Network Security and Applications. 2010*, pages 430–439, 2010.

[85] A. Reyna, C. Martín, J. Chen, E. Soler, and M. Díaz. On blockchain and its integration with iot. challenges and opportunities. *Future Generation Computer Systems*, 88:173 – 190, 2018.

[86] “block”. [Online]. Available: https://ethereum.org/en/developers/docs/blocks/. [Accessed: 30-July-2024].

[87] “algorand parameter tables”. [Online]. Available: https://developer.algorand.org/docs/get-details/parameter_tables/. [Accessed: 30-July-2024].

[88] J. Benet. IPFS - content addressed, versioned, P2P file system. *CoRR*, abs/1407.3561, 2014.

[89] S. Horrow and A. Sardana. Identity management framework for cloud based internet of things. Association for Computing Machinery, 2012.

[90] U. Habiba, R Masood, M. A. Shibli, and M. A. Niazi. Cloud identity management security issues & solutions: a taxonomy. *Complex Adaptive Systems Modeling*, 2(1):5, 2014.

[91] O. Novo. Blockchain meets iot: An architecture for scalable access management in iot. *IEEE Internet of Things Journal*, 5(2):1184–1195, April 2018.

[92] A. S. Omar and O. Basir. Identity management in iot networks using blockchain and smart contracts. In *2018 IEEE International Conference on Internet of Things (iThings)*, pages 994–1000, 2018.

[93] Bubbles of trust: A decentralized blockchain-based authentication system for iot. *Computers Security*, 78:126 – 142, 2018.

[94] G. Papadodimas, G. Palaiokrasas, A. Litke, and T. Varvarigou. Implementation of smart contracts for blockchain based iot applications. In *2018 9th International Conference on the Network of the Future (NOF)*, pages 60–67, Nov 2018.

[95] W. Li, A. Sforzin, S. Fedorov, and G. O. Karame. Towards scalable and private industrial blockchains. Association for Computing Machinery, 2017.

[96] Smart contract federated identity management without third party authentication services.

[97] M. Castillo. Honeywell Is Now Tracking $1 Billion In Boeing Parts On A Blockchain. https://www.forbes.com/sites/michaeldelcastillo/2020/03/07/honeywell-is-now-tracking-1-billion-in-boeing-parts-on-a-blockchain/. [Online; accessed 23-March-2020s].

[98] C. Gutierrez. Boeing Improves Operations with Blockchain and the Internet of Things. https://www.altoros.com/blog/boeing-improves-operations-with-blockchain-and-the-internet-of-things/. [Online; accessed 23-March-2020s].

[99] G Bertoni, J Daemen, M Peeters, and G. V. Assche. The keccak sha-3. In *"Proceedings Eurocrypt 2013-32nd Annual International Conference Theory Applications Cryptograph*, volume 7881, page 313–314.

[100] "what is mqtt quality of service (qos)". [Online]. Available: https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels/. [Accessed: 30-July-2024].

[101] "owasp threat modelling process: Stride". [Online]. Available: https://owasp.org/www-community/Threat_Modeling_Process#stride. [Accessed: 15-December-2023].

[102] E. Bertino. Data privacy for iot systems: Concepts, approaches, and research directions. In *2016 IEEE International Conference on Big Data (Big Data)*, pages 3645–3647, 2016.

[103] C. Allen. "the path to self-sovereign identity". [Online]. Available: http://www.lifewithalacrity.com/2016/04/the-path-to-self-soverereign-identity.html. [Accessed: 1-Aug-2020].

[104] "world wide web consortium". Available: https://decentralized-id.com/web-standards/w3c/. [Accessed: 21-Dec-2021].

[105] M. Hermann, T. Pentek, and B. Otto. Design principles for industrie 4.0 scenarios. In *2016 49th Hawaii International Conference on System Sciences (HICSS)*, pages 3928–3937, 2016.

[106] Hyperledger architecture, volume ii. [Available]: https://www.hyperledger.org/wp-content/uploads/2018/04/Hyperledger_Arch_WG_Paper_2_SmartContracts.pdf. [Accessed: 21-Dec-2021].

[107] M. Malik, M. Dutta, and J. Granjal. A survey of key bootstrapping protocols based on public key cryptography in internet of things. *IEEE Access*, 7:27443–27464, 2019.

[108] Types of digital certificates. [Available]: https://www.ibm.com/support/knowledgecenter/en/ssw_ibm_i_72/rzahu/rzahutypesofcerts.htm. [Accessed: 24-Sept-2020].

[109] R. Ansey, J. Kempf, O. Berzin, C. Xi, and I. Sheikh. Gnomon: Decentralized identifiers for securing 5g iot device registration and software update. In *2019 IEEE Globecom Workshops (GC Wkshps)*, pages 1–6.

[110] S. Helme. Revocation is broken. [Available]: https://scotthelme.co.uk/revocation-is-broken/. [Accessed: 23-Sept-2020].

[111] ”what is self-sovereign identity?“. [Available: https://sovrin.org/faq/what-is-self-sovereign-identity/. [Accessed: 21-Jan-2022].

[112] R. Soltani, U.T. Nguyen, and A. An. A new approach to client onboarding using self-sovereign identity and distributed ledger. In *2018 IEEE International Conference on Internet of Things (iThings)*, 2018.

[113] A. Othman and J. Callahan. The horcrux protocol: A method for decentralized biometric-based self-sovereign identity. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2018.

[114] P. C. Bartolomeu, E. Vieira, S. M. Hosseini, and J. Ferreira. Self-sovereign identity: Use-cases, technologies, and challenges for industrial iot. In *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1173–1180.

[115] D. Lagutin, Y. Kortesniemi, and N. Fotiou. Enabling decentralised identifiers and verifiable credentials for constrained internet-of-things devices using oauth-based delegation. In *Workshop on Decentralized IoT Systems and Security, San Diego, CA.*, 2019.

[116] G. Fedrecheski, J. M. Rabaey, L. Costa, P. C. Calcina Ccori, W. T. Pereira, and M. K. Zuffo. Self-sovereign identity for iot environments: A perspective. In *2020 Global Internet of Things Summit (GIoTS)*, pages 1–6, 2020.

[117] Y. Kortesniemi, D. Lagutin, T. Elo, N. Fotiou, and R. Nardone. Improving the privacy of iot with decentralised identifiers (dids). *J. Comput. Netw. Commun.*, 2019, jan 2019.

[118] S. de Diego, C. Regueiro, and G. Macia-Fernandez. Enabling identity for the iot-as-a-service business model. *IEEE Access*, 9:159965–159975, 2021.

[119] J. Lim, H. Oh, K. Sim, S. Kim, and K. Kim. A v2x access authorization mechanism based on decentralized id (did) and verifiable credentials (vc). In *2023 International Conference on Information Networking (ICOIN)*, pages 801–805, 2023.

[120] H. Saidi, N. Labraoui, A. Abba Ari, L. A. Maglaras, and J. Emati. Dsmac: Privacy-aware decentralized self-management of data access control based on blockchain for health data. *IEEE Access*, 10:101011–101028, 2022.

[121] R. Oku, K. Shiomoto, and Y. Ohba. Decentralized identifier and access control based architecture for privacy-sensitive data distribution service. In *2022 IEEE 8th World Forum on Internet of Things (WF-IoT)*.

[122] Introduction to hyperledger sovereign identity blockchain solutions: Indy, aries ursa. [Available]: https://www.edx.org/course/identity-in-hyperledger-aries-indy-and-ursa. [Accessed: 21-Dec-2021].

[123] Eip-1056: Ethereum lightweight identity. [Available]: https://eips.ethereum.org/EIPS/eip-1056. [Accessed: 28-Sept-2020].

[124] Erc: Ethereum claims registry. [Available]: https://github.com/ethereum/EIPs/issues/780. [Accessed: 28-Sept-2020].

[125] "decentralized identifiers". [Online]. Available: https://www.w3.org/TR/did-core/. [Accessed: 15-Feb-2021].

[126] A. Hertig. What Is DeFi? https://www.coindesk.com/what-is-defi. [Online: Accessed 11-Feb-2021].

[127] Medicalchain. https://medicalchain.com/en/. [Accessed: 11-Feb-2021].

[128] Electron. https://electron.net. [Accessed: 11-Feb-2021].

[129] "the data exchange for advanced mobility and infrastructure". [Online]. Available: https://terbine.com. [Accessed: 04-March-2021].

[130] A. Bröring, S. Schmid, C. Schindhelm, A. Khelil, S. Käbisch, D. Kramer, D. Le Phuoc, J. Mitic, D. Anicic, and E. Teniente. Enabling iot ecosystems through platform interoperability. *IEEE Software*, 34(1):54–61, 2017.

[131] S. J. Liebowitz. *Re-thinking the Network Economy: The True Forces that Drive the Digital Marketplace*. AMACOM, 2002.

[132] P. Banerjee, R. Friedrich, C. Bash, P. Goldsack, B. Huberman, J. Manley, C. Patel, P. Ranganathan, and A. Veitch. Everything as a service: Powering the new information economy. *Computer*, 44(3):36–43, 2011.

[133] J. Li, A. Grintsvayg, J. Kauffman, and C. Fleming. Lbry: A blockchain-based decentralized digital content marketplace. In *2020 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)*, pages 42–51, 2020.

[134] M. Cohen and A. Sundararajan. Self-regulation and innovation in the peer-to-peer sharing economy. *University of Chicago Law Review Online*, 82(1), 2015.

[135] B. Cohen. Incentives build robustness in bittorrent, 2003.

[136] J. Benet. IPFS - content addressed, versioned, P2P file system. *CoRR*, abs/1407.3561, 2014.

[137] Protocol Labs. Filecoin. https://filecoin.io. [Online: Accessed 11-Feb-2021].

[138] OrbitDB. https://orbitdb.org. [Online: Accessed 11-Feb-2021].

[139] Storj. https://storj.io. [Online: Accessed 11-Feb-2021].

[140] Sia. https://sia.tech. [Online: Accessed 11-Feb-2021].

[141] U. Habiba, R Masood, M. A. Shibli, and M. A. Niazi. Cloud identity management security issues & solutions: a taxonomy. *Complex Adaptive Systems Modeling*, 2(1):5, 2014.

[142] Oracle data marketplace. [Online]: https://docs.oracle.com/en/cloud/saas/data-cloud/data-cloud-help-center/AudienceDataMarketplace/AudienceDataMarketplace.html. [Accessed: Jul-2023].

[143] Open Bazar. https://openbazaar.org. [Online: Accessed 12-Feb-2021].

[144] LBRY. https://lbry.com. [Online: Accessed 12-Feb-2021].

[145] Origin. https://www.originprotocol.com/en. [Online: Accessed 12-Feb-2021].

[146] Y. Xu, P. Ahokangas, S. Yrjölä, and T. Koivumäki. The fifth archetype of electricity market: the blockchain marketplace. *Wireless Networks*, Jul 2019.

[147] P. Missier, S. Bajoudah, A. Capossele, A. Gaglione, and M. Nati. Mind my value: A decentralized infrastructure for fair and trusted iot data trading. In *Proceedings of the Seventh International Conference on the Internet of Things*, IoT '17, 2017.

[148] G. S. Ramachandran, R. Radhakrishnan, and B. Krishnamachari. Towards a decentralized data marketplace for smart cities. In *2018 IEEE International Smart Cities Conference (ISC2)*, pages 1–8, 2018.

[149] H. Niavis, N. Papadis, V. Reddy, H. Rao, and L. Tassiulas. A blockchain-based decentralized data sharing infrastructure for off-grid networking. In *IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 994–1000, 2020.

[150] "terbine and flash labs partner to bring distributed security to smart cities transactions". [Online]. Available: https://terbine.com/terbine-and-flash-labs-partner-to-bring-distributed-security-to-smart-cities-transactions/. [Accessed: 04-March-2021].

[151] "internet of things (iot) solutions ". [Online]. Available: https://aws.amazon.com/marketplace/solutions/IoT. [Accessed: 04-March-2021].

[152] "evernym". [Online]. Available: https://www.evernym.com. [Accessed: 15-Feb-2021].

[153] K. W. Khong, N. C. Onyemeh, and A. Y. Chong. Bsem estimation of network effect and customer orientation empowerment on trust in social media and network environment. In *Expert Systems with Applications*, volume 40, pages 4858–4870, 2013.

[154] R. Schollmeier. A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. In *Proceedings First International Conference on Peer-to-Peer Computing*, pages 101–102, 2001.

[155] L. Xiong and L. Liu. Peertrust: supporting reputation-based trust for peer-to-peer electronic communities. *IEEE Transactions on Knowledge and Data Engineering*, 16(7):843–857, 2004.

[156] M. Bender, E. Kirdan, M.O. Pahl, and G. Carle. Open-source mqtt evaluation. In *2021 IEEE 18th Annual Consumer Communications Networking Conference (CCNC)*, pages 1–4, 2021.

[157] A. Dixit and V. S. Jadon. Supply chain. https://github.com/akankshadixit/SupplyChain, 2022.

[158] P. Thakkar and S. Natarajan. Scaling hyperledger fabric using pipelined execution and sparse peers, 2021.

[159] A. Dixit, A. Singh, Y. Rahulamathavan, and M. Rajarajan. Fast data: A fair, secure, and trusted decentralized iiot data marketplace enabled by blockchain. *IEEE Internet of Things Journal*, 10(4):2934–2944, 2023.

[160] M. Qi, Z. Xu, Z. Wang, S. Chen, and Y. Xiang. Databox-based delivery service via blockchain. In *2022 IEEE International Conference on Web Services*.

[161] Decentralized identifiers (dids) v1.0. [Online]: https://www.w3.org/TR/did-core/. [Accessed: Jul-2023].

[162] Verifiable credentials data model. [Online]: https://www.w3.org/TR/vc-data-model/. [Accessed: May-22].

[163] A. Reyna, C. Martin, J. Chen, E. Soler, and M. Diaz. On blockchain and its integration with iot. challenges and opportunities. *Future Generation Computer Systems*, 88:173–190, 2018.

[164] J. Lu, J. Shen, P. Vijayakumar, and B. B. Gupta. Blockchain-based secure data storage protocol for sensors in the industrial internet of things. *IEEE Transactions on Industrial Informatics*, 18(8), 2022.

[165] R. Ma, L. Zhang, Q. Wu, Y. Mu, and F. Rezaeibagha. Be-trdss: Blockchain-enabled secure and efficient traceable-revocable data-sharing scheme in industrial internet of things. *IEEE Transactions on Industrial Informatics*, pages 1–10, 2023.

[166] S. Qi, Y. Lu, Y. Zheng, Y. Li, and X. Chen. Cpds: Enabling compressed and private data sharing for industrial internet of things over blockchain. *IEEE Transactions on Industrial Informatics*, 17(4), 2021.

[167] X. Zhang, X. Li, Y. Miao, X. Luo, Y. Wang, S. Ma, and J. Weng. A data trading scheme with efficient data usage control for industrial iot. *IEEE Transactions on Industrial Informatics*, 18(7), 2022.

[168] H. Oh, S. Park, G. M. Lee, J. K. Choi, and S. Noh. Competitive data trading model with privacy valuation for multiple stakeholders in iot data markets. *IEEE Internet of Things Journal*, 7(4), 2020.

[169] L. Tian, J. Li, W. Li, B. Ramesh, and Z. Cai. Optimal contract-based mechanisms for online data trading markets. *IEEE Internet of Things Journal*, 6(5):7800–7810, 2019.

[170] P. Banerjee, C. Govindarajan, P. Jayachandran, and S. Ruj. Reliable, fair and decentralized marketplace for content sharing using blockchain. In *2020 IEEE International Conference on Blockchain (Blockchain)*.

[171] K. Figueredo, D. Seed, and C. Wang. A scalable, standards-based approach for iot data sharing and ecosystem monetization. *IEEE Internet of Things Journal*, 9(8):5645–5652, 2022.

[172] H. R. Hasan, K. Salah, I. Yaqoob, R. Jayaraman, S. Pesic, and M. Omar. Trustworthy iot data streaming using blockchain and ipfs. *IEEE Access*, 10:17707–17721, 2022.

[173] C. Agbo, Q. Mahmoud, and J. Eklund. Blockchain technology in healthcare: A systematic review. *Healthcare*, 7(2):56, Apr 2019.

[174] J. Cui, N. Gu, and C. Wu. Blockchain enabled data transmission for energy imbalance market. *IEEE Transactions on Sustainable Energy*, 13(2), 2022.

[175] Nassar Kyriakidou, C. D., A. M. Papathanasiou, and G. C. Polyzos. (decentralized identity with applications to security and privacy for the internet of things.). *Computer Networks and Communications*, 1(2), 2023.

[176] Afjal H Sarower and Md Maruf Hassan. Necessity of reliable self-sovereign identity management framework for resource constrained iot devices. In *AIP Conference Proceedings*, volume 2579. AIP Publishing, 2023.

[177] Santiago de Diego, Cristina Regueiro, and Gabriel Maciá-Fernández. Enabling identity for the iot-as-a-service business model. *IEEE Access*, 9:159965–159975, 2021.

[178] How to monitor cron jobs. [Online]: https://www.airplane.dev/blog/how-to-monitor-cron-jobs. [Accessed: Jul-2023].

[179] Zpie: Zero-knowledge proofs in embedded systems. https://github.com/xevisalle/zpie. [Accessed: Jul-2023].

[180] Cas J. F. Cremers. The scyther tool: Verification, falsification, and analysis of security protocols. In Aarti Gupta and Sharad Malik, editors, *Computer Aided Verification*, pages 414–418, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.