Blockchain-Based Access Control Framework for Autonomous Vehicle Data Sharing

Reem Alhabib

PhD

University of York

Computer Science

September 2025

Abstract

Autonomous vehicles (AVs) generate vast amounts of data from onboard sensors, internal systems, and their surrounding environments. Sharing this data, particularly in stored form, is essential for enhancing safety, enabling post-event analysis, improving vehicle performance, and supporting regulatory oversight. However, the sensitive nature of AV data introduces significant challenges related to security, ownership, trust, and potential conflicts of interest among multiple stakeholders. Addressing these challenges requires a robust and transparent data-sharing framework that ensures traceability, accountability, and controlled access.

This research investigates blockchain technology as a foundation for secure and trustworthy AV data sharing. Blockchain offers decentralisation, immutability, and fine-grained access control, making it a strong candidate for managing sensitive automotive data. Among available platforms, Hyperledger Fabric (HLF) is selected due to its permissioned architecture, modular design, and support for customisable endorsement policies, which align with the governance requirements of the automotive ecosystem.

The research presents the design, implementation, and evaluation of a multi-party data-sharing framework for AVs using HLF and smart contracts. An additional contribution is the reconfiguration of endorsement policies (EPs), which define the participants required to validate transactions, to reflect real-world trust hierarchies. In particular, greater endorsement authority is assigned to vehicle manufacturers, reflecting their central role in the data lifecycle. To assess the implications of this design, three EP configurations are implemented and tested under varying workloads, measuring throughput, latency, and transaction success rate.

The evaluation demonstrates that system throughput is constrained by architectural bottlenecks, including peer processing capacity, single-orderer contention, and state database overhead, rather than endorsement logic alone. User concurrency and chaincode complexity are identified as key factors limiting scalability, with throughput reaching a stable ceiling under increasing transaction loads. Among the evaluated configurations, Approach 2 shows greater resilience under high workloads, while Approach 1 achieves lower latency under lighter conditions. Furthermore, stricter endorsement policies improve security and accountability but introduce measurable performance overheads, whereas relaxed policies enhance efficiency at the cost of reduced resilience and potential centralisation.

These findings indicate that practical AV data-sharing systems need to consider the alignment of endorsement policy design, chaincode complexity, and infrastructure provisioning with expected workload characteristics. They also highlight that scalability limitations in Hyperledger Fabric-based systems are driven primarily by system-level constraints rather than policy configuration alone, emphasising the need for broader architectural optimisation. Overall, this thesis presents a practical prototype for AV data sharing that integrates blockchain's capabilities with configurable governance mechanisms, offering valuable insights for the design of data-sharing systems.

Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Full Term
AV	Autonomous Vehicle
ADS	Automated Driving System
RSU	Roadside Units
NHTSA	National Highway Traffic Safety Administration
SAE	Society of Automotive Engineers
V2I	Vehicle-to-Infrastructure
V2N	Vehicle-to-Network
V2P	Vehicle-to-Pedestrian
V2V	Vehicle-to-Vehicle
V2X	Vehicle-to-Everything
OEDR	Object and Event Detection and Response
ODD	Operational Design Domain
ML	Machine Learning
GPS	Global Positioning System
Lidar	Light Detection and Ranging
Radar	Radio Detection and Ranging
5G	Fifth Generation
6G	Sixth Generation
OEM	Original Equipment Manufacturers
EDR	Event Data Recorder
DSSAD	Data Storage System for Automated Driving
ABS	Anti-lock Braking Systems
TCS	Traction Control Systems
CAVs	Connected and Automated Vehicles
VANETs	Vehicular Ad-hoc Networks
VTD	Virtual Test Drive
HLF	Hyperledger Fabric
IPFS	InterPlanetary File System
DApp	Decentralised Application
DLT	Distributed Ledger Technology
PKI	Public Key Infrastructure
CA	Certification Authority
EOV	Execute-Order-Validate
ABAC	Attribute-Based Access Control
MSP	Membership Service Provider

Abbreviation	**Full Term**
EP	Endorsement Policy
SUS	System Usability Scale
TPS	Transactions Per Second
WV	Weighted Voting

This thesis is dedicated to my parents, whose love, guidance, and sacrifices have been the foundation of everything I have achieved.

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Dr. Poonam Yadav, whose guidance, support, and encouragement have been invaluable throughout this research journey. Her presence at every step, along with her generous sharing of time and knowledge, has made this work possible.

I would also like to sincerely thank Dr. Siamak Shahandashti, a member of my Thesis Advisory Panel, for his insightful feedback, constructive suggestions, and support, which have greatly contributed to this research.

I am also very grateful to all my colleagues and SYSTRON Lab members for their collaboration, encouragement, and for creating a supportive academic environment that enriched both my research and personal growth.

I am also profoundly thankful to my little family, who left home, country, friends, and familiar surroundings to accompany me on this journey, providing me with unwavering support and love. Their sacrifices and encouragement have been a constant source of strength.

Finally, I am deeply grateful to Shaqra University for their generous funding, and to the Saudi Cultural Bureau for their invaluable support throughout my research journey.

Declaration

I declare that this thesis is a presentation of original work and I am the sole author. This work has not previously been presented for an award at this, or any other, University. All sources are acknowledged as References.

Parts of the research described in this thesis have been previously published in:

1. R. Alhabib and P. Yadav, "Data authorisation and validation in autonomous vehicles: A critical review," *Discover Applied Sciences*, vol. 7, no. 7, p. 735, 2025.
2. R. Alhabib and P. Yadav, "Hyperledger Fabric Platform for Secure and Efficient Data Sharing in Autonomous Vehicles," in *Proceedings of the 2024 11th International Conference on Wireless Networks and Mobile Communications (WINCOM)*, IEEE, 2024, pp. 1–9.
3. R. Alhabib and P. Yadav, "Impact of Custom Endorsement Policies on Hyperledger Fabric Performance in Autonomous Vehicle Data Sharing Platforms," in *Proceedings of the 2024 6th International Conference on Blockchain Computing and Applications (BCCA)*, IEEE, 2024, pp. 635–641.
4. R. Alhabib and P. Yadav, "Evaluating the impact of endorsement policies on Hyperledger Fabric performance for autonomous vehicle data sharing," *Cluster Computing*, vol. 29, no. 1, p. 8, 2026.
5. R. Alhabib and P. Yadav, "Data Sharing in Autonomous Vehicles: Hyperledger Fabric Platform," Poster presentation at the *Network and Distributed System Security (NDSS) Symposium*, 2024. Available online: `https://www.ndss-symposium.org/wp-content/uploads/vehiclesec2024-9-poster.pdf`

Chapter 1

Introduction

The National Highway Traffic Safety Administration (NHTSA) [1] defines autonomous vehicles as "those vehicles in which at least aspects of a safety-critical control function (e.g., steering, throttle, or braking) occur without direct driver input". Autonomous vehicles (AVs) represent a significant advancement in the automotive industry, offering safer, more efficient and environmentally sustainable transportation solutions [2, 3], while contributing to economic growth [4]. In recent years, leading automotive companies have invested heavily in various AV technologies, with total expenditures exceeding billions. Estimates suggest that self-driving cars may take 15 to 20 years to achieve a 25% global market share [5]. In parallel, governments around the world are establishing regulatory frameworks to facilitate the testing of AV on public roads. For instance, the UK introduced a Code of Practice in 2019 outlining guidelines for on-road trials [6]. Later, in early 2022, the UK government amended the Highway Code to ensure the safe deployment of the first self-driving vehicles [7].

An automated driving system (ADS) is a sophisticated vehicle system that integrates various onboard technologies and sensors to enable autonomous navigation from an initial location to a destination without human interaction. Cameras, GPS and other sensors with multiple control units are interconnected to facilitate data exchange, allowing the vehicle to make independent driving decisions. In addition, they can interact with any compatible system, including other vehicles, infrastructure and pedestrians. This interaction, known as vehicle-to-everything (V2X) technology, enables seamless communication between the vehicle and its surroundings. As a result, AVs operate in highly dynamic environments, requiring real-time data from multiple sources to make timely and accurate decisions.

Given the multiple sources of data and the fact that the reliability of ADS depends greatly on the accuracy, security, consistency and integrity of the data collected and generated , it is essential to maintain a trustworthy and well-structured data ecosystem. Optimising data utilisation is crucial to enhancing the overall performance of AVs. By effectively processing, managing and securely exchanging data among vehicles and other entities, these systems can improve decision-making, adaptability and safety in a highly dynamic environment.

Many studies have been conducted to explore various aspects of data in AVs, mainly focused on secure real-time data exchange [8, 9, 10, 11, 12, 13], ensuring privacy [14, 15, 16, 17, 18, 19] and preserving data integrity [20, 21, 22]. Moreover, some works have proposed innovative

methods for storing event data and protecting integrity and confidentiality using various distributed and cryptographic approaches [23, 24, 25]. This focus on accident data arises because it is important for determining liability, insurance claims, legal investigations, and improving road safety. However, accident data represent only one part of the larger data ecosystem in AVs. Modern AVs and their infrastructures continuously generate a wide variety of data, including detailed sensor logs, driving behaviour records, and environmental interaction data. These datasets are highly valuable not only for real-time operational purposes but also for long-term research, system development, policy-making, legal accountability, and commercial applications. Despite their significance, the reviewed literature has largely focused on real-time data exchange, accident data recording, and isolated integrity or privacy mechanisms, and has given limited attention to how AV data, once generated and stored, can be accessed and shared securely and in a controlled manner among multiple stakeholders. Based on the surveyed studies and existing frameworks reviewed in this thesis, no solution was identified that provides a comprehensive, decentralised framework specifically designed to support post-storage access to AV data across multiple stakeholders with fine-grained access control and integrity guarantees.

In addition to NHTSA [1], several other organisations and authors [26] have sought to identify relevant stakeholders in AV data and have provided various classifications. Those identified include the following:

1. Government and legal authorities: Accident data collected from AVs serve as crucial evidence in legal proceedings to establish liability in criminal cases. Moreover, government agencies can leverage these data to analyse accident patterns, enhance road safety measures and implement strategies to minimise future incidents and associated losses.

2. Original Equipment Manufacturers (OEMs): Manufacturers can utilise the data gathered to monitor system performance, refine their products and enhance safety features. Another potential use case is reconstructing accident scenarios to identify underlying causes and implement improvements that reduce future accident risks.

3. Suppliers: Suppliers can analyse AV data to assess the performance of their components and services, allowing them to refine their products and enhance compatibility with OEM-manufactured vehicles.

4. Vehicle owners: AV owners may use recorded data as supporting evidence to dispute liability in legal cases. These data can also be instrumental in insurance claims, ensuring fair compensation and access to necessary services.

5. Insurance companies: Stored AV data offer verifiable evidence to resolve insurance disputes fairly, preventing fraudulent claims and ensuring that all parties receive unbiased treatment.

6. Testing and certification bodies: Testing organisations require AV data to reconstruct accident scenarios, assess safety performance and update regulatory and technical standards to align with evolving industry needs.

7. Road authorities: Insights provided by data can help road authorities evaluate infrastructure conditions, improve traffic management systems and enhance overall roadside safety.

8. Researchers: Academics and industry experts can analyse AV crash data to improve vehicle design, refine autonomous driving algorithms and contribute to advancements in infrastructure and the broader mobility ecosystem.

9. Other manufacturers: Companies involved in vehicle production can use AV-generated data to improve their models, assess compatibility with autonomous driving technologies and enhance AV training in unfamiliar environments. This ultimately strengthens adaptability and operational efficiency, fostering the development of safer and more reliable autonomous systems.

Despite the clear demand for solutions that could provide stakeholders with secure and fair access to AV-generated data while ensuring integrity and protecting against misuse, existing approaches discussed in the literature only partially address these requirements, particularly when considering AV post-storage data access across a large and diverse set of stakeholders. As a result, mechanisms that support coordinated and controlled data sharing across multiple organisations remain limited in current implementations, which may affect opportunities for innovation, regulatory transparency and system improvement, especially considering the large number of stakeholders who can not only benefit from the data but also fundamentally need it. Developing data-sharing approaches that address data access control and ownership, as well as ensuring integrity, presents significant challenges in terms of continual assessment and analysis, supporting research, improving AV systems and enabling regulatory processes. Several conventional and centralised solutions [27, 28, 29] have attempted to facilitate data sharing in various fields. While these traditional centralised databases or federated access-control systems can support controlled data access, they rely on a trusted central authority responsible for managing permissions and maintaining audit logs. In environments such as AV ecosystems, where multiple independent organisations interact, reliance on a single controlling entity may introduce concerns related to transparency, single points of failure, and limited trust between stakeholders. These limitations indicate that conventional and centralised approaches are insufficient to address the requirements of secure, multi-stakeholder data sharing in autonomous vehicle ecosystems. In particular, the need for decentralised trust, tamper-resistant data integrity, and auditable data access across independent organisations cannot be effectively achieved through traditional architectures alone. Therefore, a blockchain-based approach is not only suitable but necessary to fulfil these requirements.

In recent years, blockchain has emerged as a potential secure decentralised data management system. Its core features, such as immutability, distributed consensus and transparency, make it well-suited to address integrity and trust issues in data sharing. Smart contracts further enhance its capability by enabling programmable, fine-grained access control without relying on a central authority. Within the blockchain domain, several platforms have been proposed for secure data sharing, including public blockchain networks such as Ethereum [30] and enterprise-oriented frameworks such as Quorum [31] and Hyperledger Sawtooth [32]. Public blockchains provide strong decentralisation and transparency; however, they often face challenges related to scalability, privacy and transaction costs when applied to enterprise or regulated environments. Permissioned blockchain frameworks have therefore been developed to address these limitations by enabling controlled participation and improved performance. Among these frameworks, Hyperledger Fabric (HLF) [33] provides several capabilities that meet the requirements of multi-stakeholder AV ecosystems, including permissioned identity

management, modular consensus mechanisms, and flexible governance mechanisms such as configurable endorsement policies (EP) that regulate how transactions are validated within the network. These characteristics make Hyperledger Fabric a suitable platform for supporting secure and controlled AV data sharing in the framework proposed in this research.

To address the identified gap in the literature, this work proposes a secure, decentralised data-sharing blockchain-based framework specifically designed for AV ecosystems. The proposed approach focuses on enabling post-storage access to AV data while ensuring integrity and enforcing stakeholder-specific access policies. This is achieved through the architecture shown in Figure 1.1, which combines HLF as a permissioned blockchain technology and decentralised InterPlanetary File System (IPFS) storage. The system leverages smart contracts to define and enforce fine-grained access policies and endorsement rules. This design ensures that data sharing occurs in a secure, controlled, transparent and tamper-proof manner, without the need for a central authority. The proposed system is implemented and evaluated to demonstrate its feasibility and scalability in realistic use case scenarios.

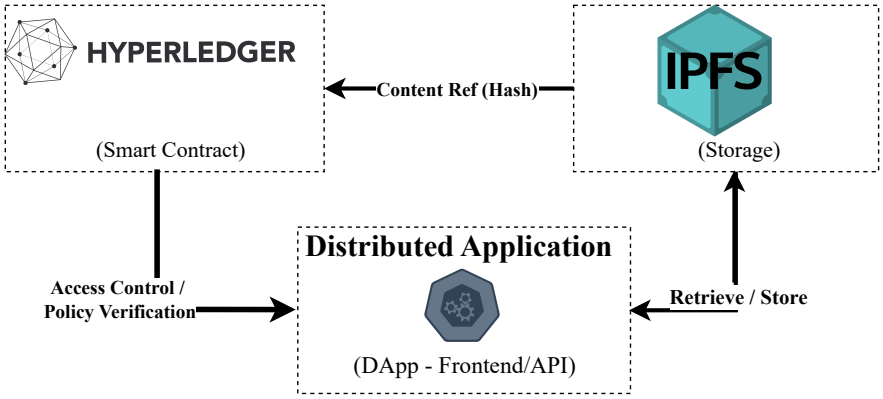


Figure 1.1: **System Overview**. This figure shows the key components of the proposed system: Hyperledger Fabric (HLF) network, InterPlanetary File System (IPFS) storage, and application layer.

The rest of this chapter is structured as follows: the Motivation and Problem Statement (1.1) outlines the research problem, the Research Aim and Research Questions (1.2) states the research goals, and the Proposed Solution (1.3) provides an overview of the proposed solution. The main contributions are then summarised in 1.4, followed by the List of Publications (1.5) and the Thesis Structure 1.6, which present related publications and thesis structure, respectively.

1.1 Motivation and Problem Statement

AVs generate vast amounts of data, including raw sensor outputs, fused data from cameras and other onboard sensors, and communication exchanges with other vehicles and Roadside Units (RSUs). This extensive dataset is crucial for reconstructing events and understanding the driving environment, particularly during critical incidents.

However, two significant challenges in AV data management and accessibility remain unresolved. First, existing storage systems, such as the Event Data Recorder (EDR) and Data Storage System for Automated Driving (DSSAD), have limited scope and capability. These systems are unable to ensure adequate data integrity and completeness, making them insufficient even for accident-related investigation [34, 35, 36].

While EDRs and DSSAD effectively capture specific event-related information, such as collisions or control status for regulatory purposes, they were not designed to support comprehensive AV data management or secure multi-stakeholder sharing. Their scope is intentionally narrow, focusing on legally mandated logging rather than broader post-operation analysis. This limitation highlights a critical gap: current solutions ensure compliance but do not provide mechanisms for authorised stakeholders to access, verify, or share the rich datasets generated by autonomous vehicles.

In this context, various approaches have been proposed to enhance data integrity (see, e.g., [37, 24, 23, 38, 20]). However, these approaches primarily focus on integrity verification mechanisms and therefore do not fully address broader requirements such as scalable data management, controlled accessibility, or secure multi-party sharing of stored AV datasets. Addressing these limitations motivates the development of the proposed blockchain-based framework for scalable, secure, and policy-driven sharing of such datasets.

Second, the demand for AV-generated data is not limited to investigation and liability purposes; it extends across multiple sectors, involving a wide range of stakeholders, including government agencies, legal authorities, OEMs, suppliers, vehicle owners, insurance companies, testing organisations, road authorities and researchers [39]. For example, sensor data analysis is essential for diagnostics in AVs, providing insights into both internal system performance and external environmental conditions [40], which in turn supports the ongoing development and improvement of autonomous technologies. Another example is the use of AV data in training and validating AI models or enhancing traffic management systems. However, existing solutions addressing AV data management and sharing remain limited in their ability to provide a secure and efficient framework that supports controlled multi-stakeholder access to these datasets.

These limitations highlight a research gap: while integrity has been a key focus of previous works, the equally important aspects of scalability, decentralised storage and policy-based access control for sharing such datasets among authorised stakeholders remain relatively underexplored. This motivates the development of the proposed framework, which incorporates distributed storage and fine-grained, attribute-based access mechanisms to support secure data sharing among multiple stakeholders.

This research studies the effectiveness of existing data-sharing solutions [41, 42, 43, 44] and introduces a framework that aims to support the decentralised, scalable, and policy-driven sharing of stored AV datasets among authorised parties.

1.2 Research Aim and Research Questions

1.2.1 Aim

This research aims to develop a multi-party data-sharing framework for stored autonomous vehicle datasets using blockchain and smart contracts. The proposed framework seeks to enhance data security and governance by enabling decentralised access control and support for scalable, user-friendly interactions among stakeholders.

1.2.2 Objectives

The aim will be achieved by addressing the following objectives:

1. To develop a permissioned blockchain framework to facilitate secure multi-stakeholder sharing of stored AV datasets.
2. To design and implement a smart contract-based system for fine-grained access control.
3. To integrate on-chain and off-chain storage approaches to optimise efficiency and scalability.
4. To develop a user-friendly interface that simplifies data sharing and policy management for stakeholders.
5. To implement and deploy a prototype of the proposed framework that integrates the aforementioned components.
6. To evaluate the performance and scalability of the framework under various conditions.
7. To assess user experience and perceived usability of the framework to ensure its accessibility and practicality.

1.2.3 Research Questions

RQ1: What are the key components and design principles required to develop a secure and scalable blockchain-based data-sharing framework for stored, non-real-time autonomous vehicles data?

To answer this question, the following sub-questions will be addressed:

a) *What are the components, tools, mechanisms, and policies required for the proposed solution?*
This involves identifying the main building blocks of the framework (e.g., blockchain network, consensus mechanism, smart contracts, data storage, and access control) and examining supporting mechanisms such as endorsement policies to ensure secure and trusted data sharing in the AV use case.

b) *Which blockchain platform is most suitable for the proposed solution? And why?*

c) *What are the types, sources and flow of the data within the system?*
This involves classifying both input and output data, as well as understanding their interactions within the framework.

d) *How can the proposed framework be implemented to integrate the identified components*

RQ2: How does the proposed blockchain-based multi-party framework perform in terms of scalability and efficiency?

This question will be answered through the following:

a) *How does the proposed framework perform in terms of throughput and latency, and how is this affected by modifying the endorsement policy to match use-case requirements?*

b) *What are the scalability limitations of the proposed framework when the number of participants, transaction rate, and data volume are progressively increased?*

RQ3: How usable is the proposed blockchain-based data-sharing system from a user perspective?

This question explores how stakeholders interact with the proposed system and evaluates its usability from the user perspective. It will focus on the intuitiveness of the interface to ensure the system meets user expectations and practical requirements. A formal evaluation method will be used to assess the system's ease of use.

As illustrated in Figure 1.3, each research question is addressed through specific objectives. RQ1 is primarily linked to Objectives 1–3 and 5, which focus on designing the framework, access control, and storage integration. RQ2 is supported by Objective 6, which covers performance evaluation. RQ3 is linked to Objectives 4 and 7, which involve designing and evaluating usability.

1.2.4 Scope of AV Data Considered in This Research

While autonomous vehicles generate and process substantial volumes of real-time sensor data during operation, it is important to clarify that this research focuses specifically on the post-storage phase of the AV data lifecycle. The proposed framework targets non-real-time, file-based datasets that have already been generated and stored, rather than live sensor streams or latency-critical control data. Consequently, continuous raw sensor streams such as full LiDAR scans, camera video feeds, or other high-frequency telemetry generated during vehicle operation are outside the scope of this framework, as these require specialised real-time processing infrastructures.

The datasets considered within this scope include driving logs, event data records, forensic evidence packages, simulation outputs, diagnostic records, and other structured

or unstructured files produced by AV systems. These datasets are typically generated during vehicle testing, operation, or incident investigation and retained for purposes such as regulatory compliance, safety analysis, system validation, research, and long-term system improvement.

In practice, such datasets are commonly distributed and shared as structured folders or compressed archives containing sensor recordings, metadata, annotations, and other contextual information. This file-based structure allows datasets to be transferred, stored, and analysed across different platforms and organisations after they have been generated.

Accordingly, the emphasis of this research is placed on secure multi-stakeholder access, integrity verification, and scalable storage integration for stored AV datasets, enabling authorised stakeholders to share and verify autonomous vehicle data throughout its post-storage lifecycle.

1.3 Proposed Solution

The proposed solution is a permissioned blockchain-based framework using HLF, that aims to provide secure, efficient and user-friendly AV data sharing among multiple stakeholders. The distributed architecture consists of three primary modules:

1. **Web Module**: Represented by the Decentralised Application (DApp), this module enables users, encompassing various stakeholders and organisations with distinct attributes, to register and interact with the system.
2. **Storage Module**: Integrating both on-chain and off-chain storage to balance integrity and scalability using InterPlanetary File System (IPFS), this is a distributed cloud storage solution [45]. It provides users with a secure and scalable platform for uploading large datasets while ensuring data integrity.
3. **Network Module**: This module utilises a permissions blockchain, which offers decentralisation, tractability, and immutability and is responsible for enforcing secure access control mechanisms. This solution integrates HLF [33], which is a permissioned blockchain platform providing security, scalability and modularity for AV data sharing.

Figure 1.2 illustrates the proposed solution, highlighting the dynamic interplay among the Web, Storage, and Network modules, and their interactions with end-users.

Together, these modules form a cohesive, secure, and scalable framework that enables trusted multi-party data sharing for autonomous vehicles, balancing trust, performance, and usability across stakeholders. The integration of HLF with on-chain and off-chain storage ensures that data remains verifiable, accessible, and protected within a controlled blockchain environment.

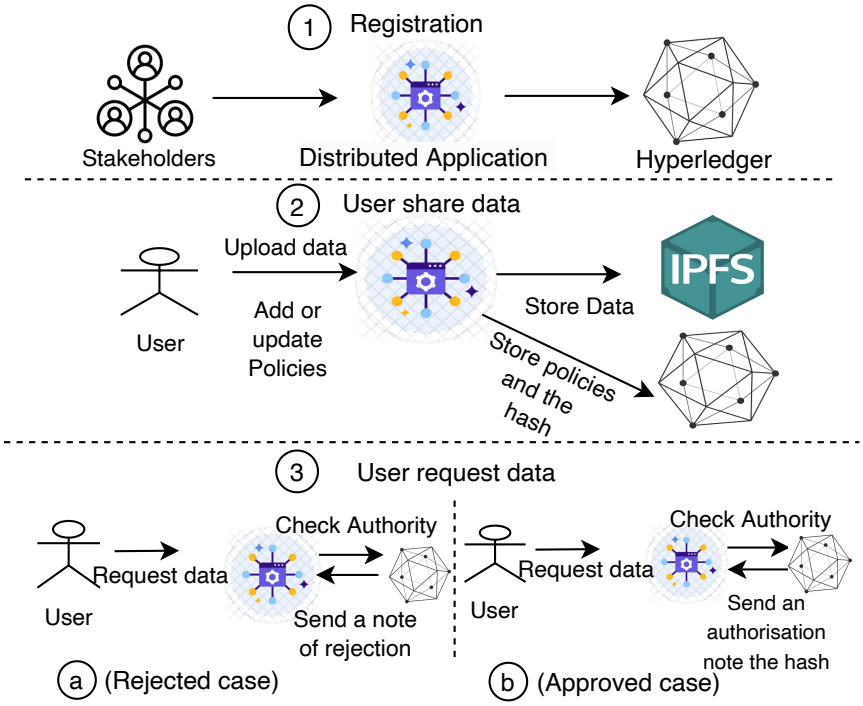


Figure 1.2: **Proposed Solution**. In this illustrative representation, the dynamic interplay among the components portrays the intricate relationships between the core components and the end-users. 1) **User Interaction (Web Module)**: managed by the DApp. 2) **Secure Data Management (Storage Module)**: utilising IPFS. 3) **Decentralised Network (Network Module)**: governed by HLF.

1.4 Contributions

This research makes several key contributions to the development of secure, scalable, and user-friendly multi-stakeholder data sharing frameworks for autonomous vehicles as follows:

C1. A Decentralised Framework for Multi-Stakeholder AV Data Sharing. This work introduces the design and implementation of a blockchain-based framework, built on Hyperledger Fabric, that enables secure, auditable, and controlled data sharing among multiple stakeholders in autonomous vehicle (AV) ecosystems. The framework integrates core blockchain features with domain-specific requirements for AVs, bridging the gap between theoretical proposals and practical deployment. In addition, smart contract modules are developed to enforce fine-grained access control over AV data, ensuring that stakeholders are granted differentiated permissions according to their roles. This contributes to ongoing efforts in the field by demonstrating how decentralised architectures can strengthen trust and accountability in collaborative data environments.

C2. Endorsement Policy Customisation. This research contributes a systematic exploration of endorsement policy configurations in Hyperledger Fabric, demonstrating how customisation can balance trade-offs between requirement and performance in AV data-sharing scenarios. By evaluating different endorsement strategies, the study

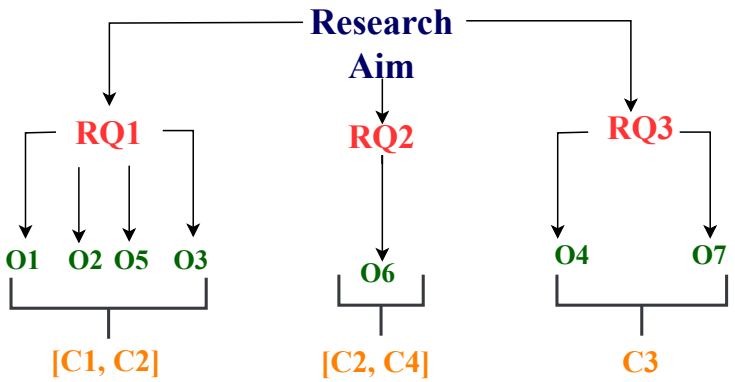


Figure 1.3: Mapping of Research Questions (RQ) to Objectives (O) and Contributions (C).

provides concrete guidance on how policy choices affect latency, throughput, and security guarantees. Through this, the work highlights the role of consensus design choices in shaping security and performance outcomes in specialised application domains.

C3. User-Friendly Interface for Blockchain-Based Data Sharing. A practical and user-oriented interface is designed and tested, enabling seamless interaction with the decentralised framework and improving accessibility for non-expert users. The usability assessment demonstrates that the proposed interface improves user experience and lowers the entry barrier for blockchain adoption in autonomous vehicle data-sharing scenarios.

C4. Comprehensive Performance Evaluation. An extensive evaluation of the system is conducted under varying network conditions, user loads, and transaction rates, providing insights into the scalability and robustness of the proposed framework. The evaluation includes comparative analyses across multiple system configurations, highlighting both strengths and limitations of the framework. These findings not only validate the practicality of the proposed design but also provide empirical evidence for future optimisation of blockchain-based AV data-sharing solutions.

Collectively, these contributions advance the state-of-the-art in blockchain-based AV data sharing by addressing security, scalability, usability, and performance challenges, delivering both conceptual advancements and practical implementations that can inform future research.

1.5 List of Publications

The research presented in this thesis has led to the publication of the following peer-reviewed outputs:

1. R. Alhabib and P. Yadav, "Data authorisation and validation in autonomous vehicles: A critical review," *Discover Applied Sciences*, vol. 7, no. 7, p. 735, 2025.

2. R. Alhabib and P. Yadav, "Hyperledger Fabric Platform for Secure and Efficient Data Sharing in Autonomous Vehicles," in *Proceedings of the 2024 11th International Conference on Wireless Networks and Mobile Communications (WINCOM)*, IEEE, 2024, pp. 1–9.

3. R. Alhabib and P. Yadav, "Impact of Custom Endorsement Policies on Hyperledger Fabric Performance in Autonomous Vehicle Data Sharing Platforms," in *Proceedings of the 2024 6th International Conference on Blockchain Computing and Applications (BCCA)*, IEEE, 2024, pp. 635–641. **This paper was selected for an invitation to submit an extended version to the journal *Cluster Computing*.**

4. R. Alhabib and P. Yadav, "Evaluating the impact of endorsement policies on Hyperledger Fabric performance for autonomous vehicle data sharing," *Cluster Computing*, vol. 29, no. 1, p. 8, 2026.

5. R. Alhabib and P. Yadav, "Data Sharing in Autonomous Vehicles: Hyperledger Fabric Platform," Poster presentation at the *Network and Distributed System Security (NDSS) Symposium*, 2024. Available online: `https://www.ndss-symposium.org/wp-content/uploads/vehiclesec2024-9-poster.pdf`

1.6 Thesis Structure

The remainder of the thesis is structured as follows:

- **Chapter 2 Background and Literature Review**. This chapter provides background information concerning blockchain technology, smart contracts, and their applications. It also reviews related work on data sharing, privacy, and trust in autonomous vehicle ecosystems. **(Addresses RQ1a and b)** by identifying the key components, mechanisms, and data flows relevant to the proposed framework.

- **Chapter 3 Research Methodology and the Proposed Solution**. This chapter outlines the research methodology, including the design science research approach used in the study. It presents the design principles, architecture, and functional requirements of the proposed blockchain-based data-sharing framework. **(Addresses RQ1a, b and c)** by defining the framework's design principles, components, and policies.

- **Chapter 4 System Implementation**. This chapter describes the technical implementation of the proposed solution. It includes the development environment, technologies used, system modules, and smart contract design. **(Addresses RQ1d)** by detailing how the identified components and blockchain platform were applied in practice.

- **Chapter 5 Baseline Performance Evaluation**. This chapter presents the initial performance evaluation of the system using a default endorsement policy. It evaluates scalability, throughput, and latency under various conditions to establish a baseline. It also includes a usability evaluation using the System Usability Scale (SUS). **(Addresses RQ2a, b and RQ3)** by measuring baseline performance and usability.

- **Chapter 6 Customised Endorsement Policy**. This chapter describes a customised endorsement policy designed specifically to meet the use case requirements of the AV data-sharing scenario. It includes rationale, implementation details, and comparative performance analysis with previous approaches. **(Addresses RQ2a–b)** by analysing the performance impact of tailoring endorsement policies.

- **Chapter 7 Simulated Weighted Voting Endorsement Policy**. This chapter introduces the weighted voting endorsement policy, which tailors validation authority based on stakeholder roles. It presents the implementation and performance evaluation of this modified approach. **(Addresses RQ2a–b)** by exploring scalability and efficiency under a modified endorsement scheme based on weighted voting.

- **Chapter 8 Comparative Analysis and Discussion**. This chapter compares the results from all three approaches, discussing trade-offs between scalability, security, and system complexity. It also reflects on the implications of endorsement policy design in blockchain-based systems. **(Addresses RQ1 and RQ2)** by synthesising findings across design, performance, and usability.

- **Chapter 9 Conclusions**. This final chapter summarises the research contributions, reflects on the findings, and discusses limitations. It also outlines possible directions for future research and improvements to the system. **(Addresses RQ1–RQ3)** by providing an overall synthesis and highlighting future work.

Chapter 2

Background and Literature Review

This chapter provides an overview of the technologies, tools, and methods relevant to the research conducted in this thesis. It also includes a review of the current state-of-the-art solutions, highlighting existing limitations and gaps that motivate the proposed work. In particular, it addresses **RQ1a** and **RQ1b** by identifying the essential components, mechanisms, and policies for the proposed framework, and by reviewing the suitability of different blockchain platforms.

This chapter begins with Autonomous Vehicles (AVs) and Data Ecosystems (2.1), introducing AVs and issues with data. It then covers Distributed Ledger and Blockchain Technology (2.2), providing background on distributed ledger and blockchain technology, followed by Decentralised Content Distribution Platforms (2.3). Then section 2.4 reviews existing literature in the field. The chapter concludes with Summary and Research Gap (2.5) and a conclusion in 2.6.

2.1 Autonomous Vehicles (AVs) and Data Ecosystems

While traditional vehicles still dominate roadways, forecasts indicate a rapid increase in the adoption of connected and autonomous vehicles. By 2030, the number of AVs could surpass 90 million [46]. AVs represent a great transformative advancement in transportation, integrating artificial intelligence (AI), sensor technologies, and communications to enable self-driving capabilities. They are designed to operate with minimal or no human intervention, relying on advanced computational models and sensor fusion techniques to navigate their environment. Their development has been driven by many potential benefits, such as improved road safety, increased mobility for individuals with disabilities, reduced energy consumption and pollution, and optimised traffic flow [2, 3]. In addition, AVs contribute significantly to the reduction of harmful gas emissions, consistent with the global goal of achieving net zero by 2050 [47].

In addition, AVs have great potential to reduce crash risks, as many conventional vehicle accidents occur due to human error and distracted driving. Advanced driver assistance systems (ADAS) can significantly improve road safety and help prevent accidents [48]. For

example, in the US, it has been estimated that partially automated crash avoidance features could reduce the severity of up to 1.3 million crashes every year, including 133,000 injury crashes and 10,100 fatal crashes [49].
This section continues with an overview of such AV systems, focusing on their architecture, sensing technologies and data flow, as well as associated data management challenges.

2.1.1 Definitions

The ADAS have been instrumental in the development of AVs, offering critical functionalities for vehicle control and safety. The early generations of assistance systems relied on sensors to monitor internal vehicle conditions, with a primary focus on enhancing safety and stability. During the 1980s, systems such as Anti-lock Braking systems (ABS) and Traction Control Systems (TCS) emerged, which aimed to improve the dynamic stability of vehicles [50]. The latest generation of ADAS has been designed with the primary goal of preventing collisions. These systems are autonomous, meaning that they are able to change the behaviour of vehicles in response to unanticipated events during operation [51].

The NHTSA [1] defines an AV as a vehicle in which at least one safety-critical function (such as steering, braking, or throttle control) occurs without direct driver input. Vehicles that only provide safety warnings without executing control functions are not considered automated. The Society of Automotive Engineers (SAE) [52] classifies AVs into six levels based on their degree of automation:

- **Level 0** (no automation): The human driver performs all tasks, although the vehicle may provide warnings or momentary assistance.
- **Level 1** (driver assistance): The system can assist with steering or acceleration/deceleration, but the driver remains responsible for the vehicle's operation.
- **Level 2** (partial automation): The vehicle can simultaneously control steering and acceleration, but requires constant driver supervision.
- **Level 3** (conditional automation): The vehicle handles all driving tasks within predefined scenarios, but requires driver intervention when prompted.
- **Level 4** (high automation): The vehicle is fully autonomous in specific environments (e.g., urban or highway conditions) but may require human control in unstructured settings.
- **Level 5** (full automation): The vehicle operates independently under all conditions, eliminating the need for a human driver.

These classifications provide a structured framework for AV development, guiding regulatory policies and technological advancements.

2.1.2 AV Architecture

AV systems typically encompass three to five core functions: perception, localisation, planning, control and system management [26, 53], as depicted in Figure 2.1. These key AV operations are as follows:

1. Perception involves gathering data and interpreting environmental information, for instance, recognising road signs. This process relies on various sensors, including cameras and Radio Detection and Ranging (RADAR).
2. Localisation enables the AV to determine its precise position and orientation in relation to its surroundings.
3. Planning consists of three stages:
    - The path planner computes the most efficient geometric route using algorithms.
    - The behaviour planner determines the optimal driving actions based on the computed path.
    - The system then estimates the best route, considering vehicle dynamics and environmental constraints.
4. Control is responsible for implementing planned manoeuvres and regulating vehicle movements, such as lane changes.
5. System management encompasses tasks related to event data recording, human-machine interactions through in-vehicle interfaces and external communication interfaces.

These operations rely on the continuous collection and processing of vast amounts of data, often reaching up to 14.424 TB daily [54]. These data are critical not only for real-time decision making but also for system updates, event recording and post-operation analysis, making efficient data management and integrity crucial in AV systems.

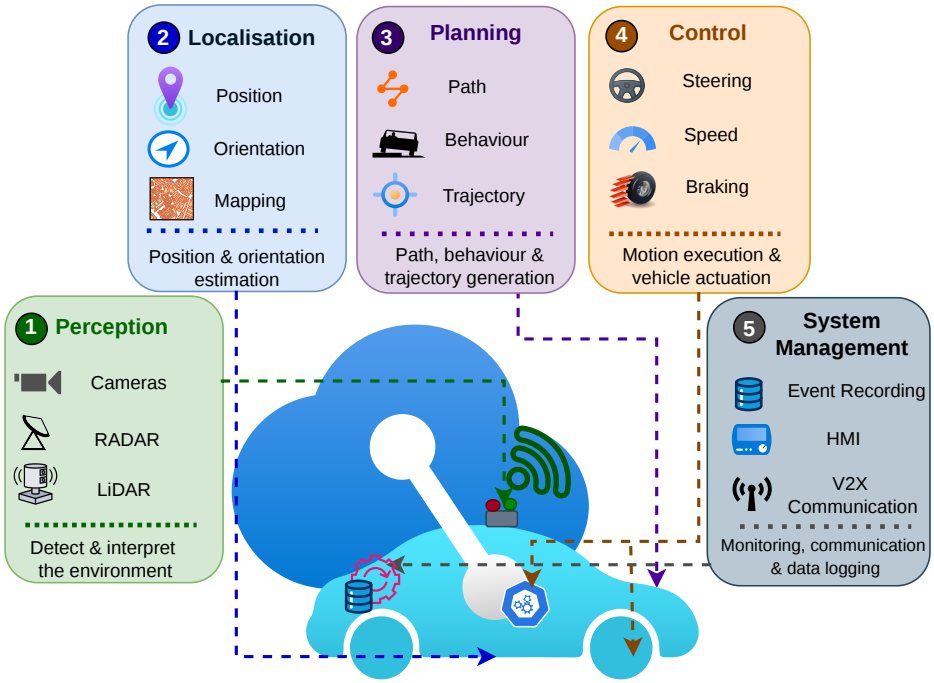


Figure 2.1: **Key Operations of an AV.** This figure presents the fundamental operations of an AV, categorised into five main areas: (1) perception, (2) localisation, (3) planning, (4) control, and (5) system management.

2.1.3 Sensor Technologies

To gain a comprehensive and real-time understanding of the surroundings, AVs must rapidly and accurately identify, interpret and track all objects in the environment. Thus, multiple sensors are integrated to offer both perceptual and positional insights, enabling the vehicle to make informed real-time decisions. Sensors are devices that convert detected objects or environmental changes into measurable quantitative data for further processing [55]. AV systems incorporate two primary types of sensors based on their operational functions: exteroceptive and proprioceptive:

- Exteroceptive sensors: These sensors perceive the external environment by detecting objects, distances and environmental factors, such as light intensity. Common examples include cameras, and light detection (LiDAR) and RADAR systems.
- Proprioceptive sensors: These sensors measure the internal state of the vehicle, capturing its dynamic condition and internal values. Examples include global positioning systems (GPS), accelerometers and encoders [56].

Since each sensor has its own advantages and limitations, the integration of multiple sensors is needed to obtain as accurate and reliable perception of the environment as possible, a process known as *sensor fusion*. *Sensor fusion* combines data from multiple sources to

enhance decision making by leveraging the strengths of different sensors while compensating for their weaknesses. A classic example is the fusion of RADAR and camera data. While RADAR is unaffected by environmental lighting conditions, it lacks the ability to provide detailed object shape information. In contrast, cameras offer rich visual details but may struggle under varying illumination conditions[57]. The first step in sensor fusion is sensor calibration, which ensures that the AV system correctly aligns the position and orientation of each sensor in both space and time [58]. Once calibrated, data from multiple sensors observing the same object can be merged to improve accuracy and provide meaningful insights.
The high-resolution capabilities of sensors, particularly LiDAR and camera systems, essential for enabling AVs to perceive and interpret their surroundings with precision.
The raw data captured by individual sensors and the insights derived from sensor fusion provide rich, multi-dimensional information, not only for real-time decision making but also for post-operation analysis. These data play a crucial role in system diagnostics, performance improvement and determining the root causes of incidents or crashes, making them indispensable for both operational integrity and continuous AV development.

2.1.4 Vehicle Communication and Connectivity

AVs have the capability to interact with various compatible systems, including other AVs, infrastructure and pedestrians. This interaction is facilitated through vehicle-to-everything (V2X) communication, which enables seamless data exchange between the vehicle and its surroundings. To assure an efficient and cooperative driving environment, the network must maintain high speed, reliability and security while minimising latency. Currently, two primary communication technologies are widely adopted: the 802.11p wireless standard and mobile networks, particularly 5G. Looking ahead, sixth-generation (6G) wireless networks are expected to play a crucial role in enhancing V2X communications for AVs [59, 60].

By combining sensing, communication and driving automation, AVs evolve into connected and Automated Vehicles (CAVs), which not only execute automated driving functions but also establish connectivity with other vehicles, infrastructure,and cloud services. Communication technologies are integral to enabling AVs to interact with their surroundings, thus improving overall road safety, traffic efficiency and user experience. The key communication technologies essential for vehicular networking include Vehicular Ad-hoc Networks (VANETs) and different types of vehicle communication, such as vehicle-to-vehicle (V2V), vehicle-to-infrastructure (V2I) and V2X.

2.1.5 Data Flow

The functionality of AVs relies on the continuous and efficient exchange of data. As depicted in Figure 2.2, these data originate from multiple sources, including onboard sensors and external inputs, and undergo various stages of processing at both local and edge computing levels. Ultimately, the processed data are either stored or analysed in backend systems.

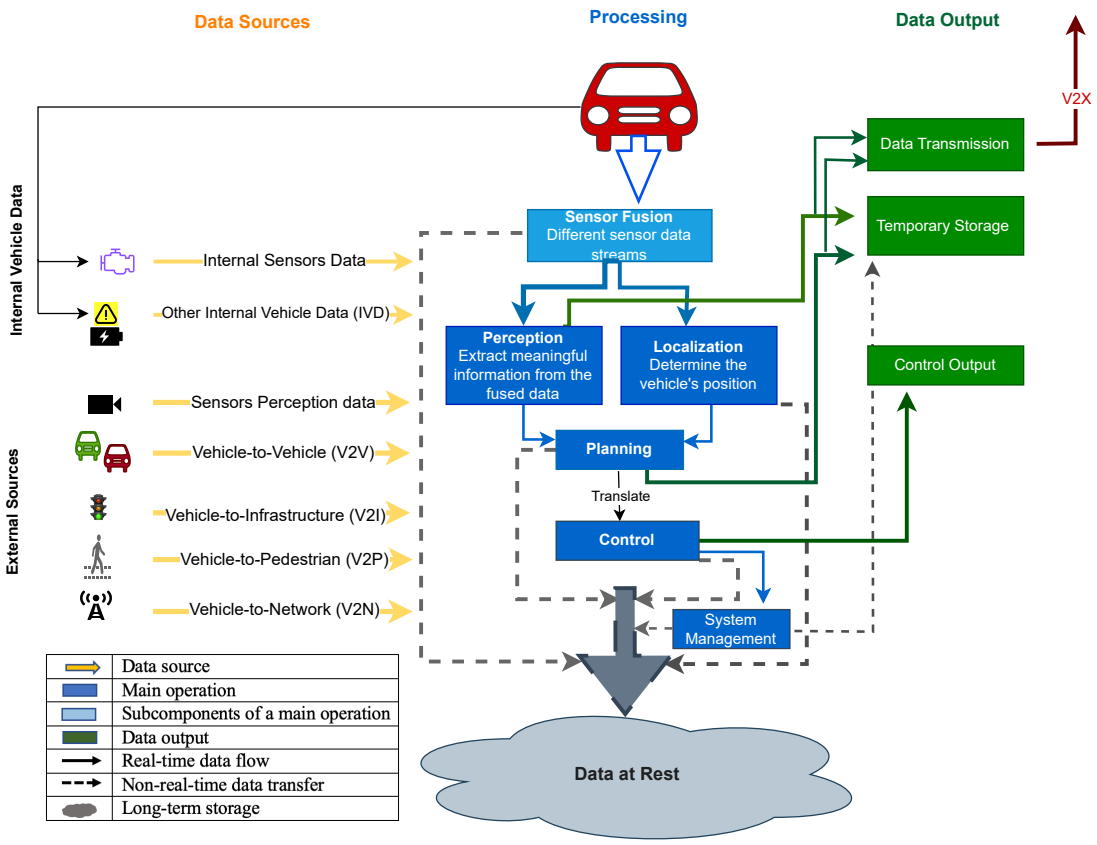


Figure 2.2: **Data Flow in an Autonomous Vehicle (AV).** This diagram illustrates the movement of data within an AV system, highlighting how sensors' inputs and external information are integrated and processed through key stages: perception, localisation, planning and control. The resulting outputs include control commands, data transmission via V2X communication and temporary storage, all of which contribute to long-term data retention in storage systems.

Beyond supporting real-time decision making, these data, both in raw and processed forms, are vital for post-operation uses, such as improving AV algorithms, identifying system failures and investigating traffic incidents.

While Figure 2.2 illustrates the broader autonomous vehicle data ecosystem, including real-time sensing, processing, communication, and storage components, this thesis concentrates specifically on the post-storage stage of the data lifecycle (data at rest) shown at the bottom of the figure. The proposed framework does not intervene in real-time data acquisition, sensor fusion, or latency-sensitive vehicle control systems. Instead, the focus is placed on datasets that have already been generated and stored, such as archived logs, event records, simulation outputs, diagnostic files, and other retained AV datasets. Within this stage, the primary challenges relate to secure multi-stakeholder access, policy enforcement, governance and scalable storage integration.

2.1.6 Current State of AV Data Storage and Accessibility

To distinguish different storage approaches, we can categorise them into two groups. First, **real-time storage solutions** include onboard storage, edge storage and in-memory databases. These tools process and store data immediately to support decision-making and vehicle control. Second, **non-real-time data storage solutions** primarily store data for long-term use, training AI models, regulatory compliance and post-drive analysis. Examples include Event Data Recorders (EDRs), Data Storage Systems for Automated Driving (DSSAD) and cloud storage.

An EDR is an in-vehicle recording device designed to capture event-related data in a readily accessible format[39]. These devices play a crucial role in crash investigations and performance analysis by recording key data moments before and after a collision. Essential information, such as vehicle speed and brake status, is stored, aiding in accident assessments and safety evaluations. DSSAD are storage devices that determine whether the system or the driver was in control of the vehicle at a specific time [61]. These systems thus provide information that can help determine responsibility and liability issues.

However, despite the critical role of these storage solutions in understanding AV behaviour, they remain limited in several respects in terms of reconstructing complex events, identifying system failures, addressing liability concerns and fully analysing the causes of crashes. It is important to note that these limitations largely stem from the regulatory scope and intended purpose of systems such as EDRs and DSSAD, which are designed to provide targeted, legally mandated recordings for accident investigation and liability determination rather than comprehensive AV data management. Consequently, they not only lack comprehensive data capture but also accessibility [62], significantly hindering transparency, accountability and the ability to improve AV technologies based on real-world performance and failures. Therefore, researchers are seeking to address these challenges by enhancing data integrity and expanding storage capacity.

2.1.7 Open Source Data and Simulators

Recently, many studies have sought to address the lack of an open platform for AVs that can be used to study and improve this technology (see, e.g., [63, 64, 65, 66]). Simulation tools play a critical role in the development and evaluation of AV systems. These tools provide virtual environments enabling researchers and developers to test perception and planning and control algorithms without the risks and costs associated with real-world testing. In terms of their design, simulation platforms vary concerning their focus, accuracy, scalability and ease of use. During the development of this prototype, we studied and summarised the available tools. Table 2.1 presents these characteristics, highlighting the strengths and limitations of the best-known tools.

Tool	Type	Access	Purpose	Accuracy	Scale
CARLA [67]	Simulator	Open	General	High	High (Parallel)
Autoware.AI [68]	Stack/Data	Open	General	High	High (Modular)
PreScan [69]	Simulator	Commercial	General	Very High	High
VTD [70]	Simulator	Commercial	General	Very High	High
AirSim [71]	Simulator	Open	General	High	Moderate
DriveSim [72]	Simulator	Commercial	General	Very High	High
Luminar Atlas [73]	Sensor Sim.	Commercial	LiDAR	Very High	High
Waymo Open Dataset [74]	Dataset	Public	Real-world	Real	High
nuScenes [75]	Dataset	Public	Real-world	Real	Very High
OpenPilot [76]	Stack/Data	Open	L2 Autonomy	Real	High
AI4CE * [77]	Dataset/Toolkit	Open	Perception	High	Moderate

Table 2.1: Comparison of Autonomous Vehicle Development Resources. * indicates the tool used in this research.

2.2 Distributed Ledger and Blockchain Technology

Unlike traditional centralised databases, distributed ledger technology (DLT) eliminates the need for a single trusted authority by enabling multiple participants to maintain a shared and tamper-resistant ledger. The integration of various cryptographic techniques, such as Merkle trees, hash functions and public-key cryptography, plays a crucial role in ensuring data integrity and security. Thus, DLT has emerged recently as a transformative approach to storing and managing digital transactions in a decentralised manner. Blockchain is the most widely adopted implementation of DLT. This section provides an overview of DLT, its key cryptographic tools and the fundamentals of blockchain technology.

2.2.1 Distributed Ledger Technology (DLT)

DLT is one of the most promising innovations reforming the field of information technologies. It has provided salient growth in various decentralised systems worldwide from inception in the 1990s through the launch of the blockchain concept in 2009.

Distributed Ledger Technology (DLT) Properties

The various properties of DLT illustrate the significance of this technology in reshaping industries and research. The following are the main features:

- **Immutability**: Immutability means that any validated data is irreversible. Once data are recorded in the ledger, they cannot be modified or deleted without consensus from the network. This ensures tamper-resistance and provides a verifiable audit trail, which is essential in domains such as finance, healthcare, and autonomous vehicles.

- **Transparency:** This means all participants have a full copy of the ledger. Such openness allows every participant to independently verify transactions, increasing accountability and reducing the need for blind trust in intermediaries.

- **Anonymous and Confidential**: The term anonymous in this context signifies that the identity of the users is anonymous.

- **Time-Stamped**: This feature refers to the logging of precise details of when an event occurred. Every transaction is recorded with a cryptographically verifiable timestamp, which is critical for maintaining chronological order, enabling traceability, and supporting legal or compliance requirements.

- **Unanimity and Consensus Mechanism**: In the context of DLT, the consensus mechanism concerns the procedure by which the nodes agree on the replication state of the maintained ledger. Different consensus models (e.g., Proof of Work, Proof of Stake, Practical Byzantine Fault Tolerance, and Raft in permissioned systems) offer trade-offs between scalability, energy efficiency, and security, making the choice of consensus vital for the performance of a given application.

- **Secure**: Each network asset is secured by reliable encryption. It can only be accessed and edited by those who are permitted to do so by their private key. Public-key cryptography ensures authentication and authorisation while hashing techniques protect the integrity of data. The distributed nature of the network also mitigates single points of failure, making it more resilient to attacks.

- **Programmable**: Distributed ledgers have developed into scalable and programmable platforms. Ethereum and IBM's HyperLedger Fabric are two examples of programmable DLT, affording opportunities for the development of solutions related to using a database or ledger.

2.2.2 Key Cryptographic Concepts in Distributed Ledger Technology (DLT)

In the context of DLT, a node refers to an individual device that is part of the decentralised network maintaining the ledger. Nodes can join and take part in the maintenance of a DLT design, which can be implemented as a public or private distributed ledger. Public DLT is designed to enable any node to join and interact in the distributed ledger. It is not necessary to register or confirm the identities of the nodes. This feature provides a high level of availability. However, it needs to be easily scalable to maintain performance when the number of nodes grows. In some situations, when the public have not able to access the assets, private DLT is preferred by businesses. Participating in private DLT requires verification of the nodes by a cryptographic key mechanism. To support these models, DLT relies heavily on cryptographic primitives that guarantee security, integrity, and trust across distributed participants. The following subsections are essential concepts in DLT.

Hash Function

In Distributed Ledger Technology (DLT), hash functions are fundamental for constructing secure and efficient data structures, such as blockchain blocks and transaction records. A hash function is a deterministic mathematical algorithm that maps an input of arbitrary length to a fixed-size output, referred to as a hash value or digest. Cryptographic hash functions, such as SHA-2 and SHA-3, are widely adopted in modern systems due to their strong security properties.

Hash functions play a critical role in ensuring data integrity and are commonly used in digital signatures, where messages are hashed prior to signing to improve both efficiency and security.

A cryptographic hash function H is expected to satisfy the following core properties:

- **Pre-image Resistance:** Given a hash value y, it is computationally infeasible to find an input x such that $H(x) = y$.
- **Second Pre-image Resistance:** Given an input x, it is computationally infeasible to find another input $x' \neq x$ such that $H(x) = H(x')$.
- **Collision Resistance:** It is computationally infeasible to find any two distinct inputs $a \neq b$ such that $H(a) = H(b)$ [78].

These properties make hash functions indispensable in DLT systems. In blockchain architectures, hash functions are used to link blocks together, ensuring that any modification to a previous block invalidates all subsequent blocks. They are also used to generate transaction identifiers and support the immutability of ledger data.

Overall, hash functions provide the cryptographic foundation for maintaining integrity, transparency, and trust in distributed environments.

Merkle Tree

A Merkle tree is a hash-based data structure that is used to encode DLT data efficiently and securely. In this context, Merkle trees are used to verify the data contained in a block. It has two types of nodes: *leaf nodes* and *non-leaf nodes*. Each *non-leaf node* is connected to two child nodes. Each *leaf node* is a hash value of a dataset. However , the *leaf nodes* are not connected to child nodes. The nodes' hash values are therefore iteratively concatenated and hashed, beginning with the *leaf nodes* and continuing to the root of the Merkle tree, which is called the root hash. Each piece of data is hashed, then each pair of hashes is concatenated and hashed together, and so on. Merkle trees are especially useful in distributed, peer-to-peer (P2P) systems in which the same data should exist in multiple places. In these use cases, data consistency and integrity are crucial and are guaranteed through data verification. It is time-consuming and computationally costly to examine the full content of each data file; however, the root hash here can even be used to verify the integrity of multiple data blocks without the need to process the entire block. In summary, Merkle trees provide

data integrity, efficiency and scalability. Thus, Merkle trees are a cornerstone of blockchain systems, supporting secure transaction validation, lightweight clients, and synchronisation in large-scale distributed ledgers.

Public Key Infrastructure

Public Key Infrastructure (PKI) was the starting point for modern security mechanisms on systems that implement digital certificates. PKI is based on asymmetric cryptography, employing a public and a private key pair to verify identities in the network using the relevant public key. This concept is used to enable the encryption and decryption of data. PKI needs certain entities to perform certain tasks. For instance, the Certification Authority (CA) is responsible for the management of digital certificates. Public key cryptography is a critical application in DLT since it is essentially used to create and digitally sign transactions. Authentication, often known as proof of ownership, is done as part of the transaction validation process. A transaction is digitally signed using the private key to confirm the legitimacy of the transaction's origin. The use of cryptographic keys guarantees the integrity of transactions and the security of user identities. By employing PKI, DLT guarantees transaction integrity and the security of user identities, preventing unauthorised modifications and ensuring that only legitimate participants can initiate transactions. This mechanism also enables trust in a decentralised environment, where nodes may not inherently trust each other but can rely on cryptographic proofs of identity and ownership.

Consensus Mechanisms

The efficient functioning of DLT depends critically on two innovations: information security protocols, such as hashing and encryption (as explained previously), and consensus mechanisms. Consensus mechanisms are fault-tolerant techniques that aim to achieve distributed agreement concerning the ledger's state. The algorithms exclude third-party intermediaries to ensure the correctness of transactions. The most popular algorithms are as follows:

- Proof-of-Work (PoW): This was first introduced in Bitcoin and later adopted by other cryptocurrencies, such as Ethereum. The DLT systems that use PoW as their consensus mechanism need each node to solve a complex computational task before new transactions can be added to the ledger. For example, in Bitcoin, nodes must find a nonce whose hash meets a target number of leading zeros. PoW ensures strong security in trustless networks but is highly energy-intensive.
- Proof-of-Stake (PoS): For low-powered communication channels, like the Internet of Things. Instead of solving computational puzzles, validators are chosen based on their stake in the network, significantly reducing energy consumption and computational requirements.
- Paxos: This is a method that supports the achievement of consensus by allowing nodes to select a single value from a range of options. Paxos can be difficult to implement due to its strict formalism and the need to handle various failure scenarios.

- RAFT is a simpler alternative to PaXoS, commonly used in permissioned blockchains like Hyperledger Fabric. Consensus is achieved through a leader node, with followers replicating the leader's log. Heartbeat messages maintain synchronisation, ensuring safety and consistency. RAFT is crash fault-tolerant, lightweight, and ideal for networks where nodes are semi-trusted.

Consensus mechanisms are essential for DLT as they ensure secure, data-integrated and participant-agreed-upon participation. The consensus mechanism chosen has a substantial effect on the decentralisation, security, efficiency and scalability of DLT.

2.2.3 Blockchain

Currently, Blockchain is one of the most widely established constructs. In 2008, Nakamoto [79] built Bitcoin, a novel electronic payment system based on cryptographic proof instead of trust. The Bitcoin system allows online payments to be made directly from one party to another without going through a banking institution. In other words, it is a peer-to-peer (P2P) electronic currency. By adding each transaction's hash to an ongoing chain of hash-based PoW, the network timestamps transactions and establishes both the observed order of events and the source of the transaction's CPU power. Nodes can quit and rejoin the network at any time and messages are broadcast. A block is a data structure that records transactions along with supplementary information, including a reference to the previous block. Figure 2.3 illustrates the core blockchain components categorised into a layered architectural framework.

A node can be any user or server computer with a membership that represents its individual identities. It contains the smart contract, which is a self-executed code that is run to apply roles and conditions between the parties. Only with this code can the private data storage be read, written and edited. A node has the ability to start a transaction, which represents the smallest unit of activity on the blockchain. These transactions are grouped into a block, which is only added to the chain after verification by miners or endorser nodes. Events, which are inherent components of smart contracts, store the arguments passed in transaction logs, while system administrators define the guidelines for blockchain operations. This layered representation describes the structural organisation of blockchain systems rather than the specific transaction processing sequence, which may vary across different blockchain platforms.

Taxonomy of Blockchain Systems

As shown in 2.2, the three main types of blockchain platforms currently in use are public blockchain, consortium blockchain and private blockchain. In a public blockchain, such as Ethereum [30] and Bitcoin [79], everyone may access all records and participate in the consensus process. In contrast, in a consortium blockchain (or federated BC), such as Quorum [31], the consensus process will involve only a select few nodes. In private blockchains, only nodes from a single organisation are permitted to participate in the consensus process. In some private blockchain implementations, such as HLF [33], the consensus process can be restricted to nodes belonging to a single organisation, although the platform also supports

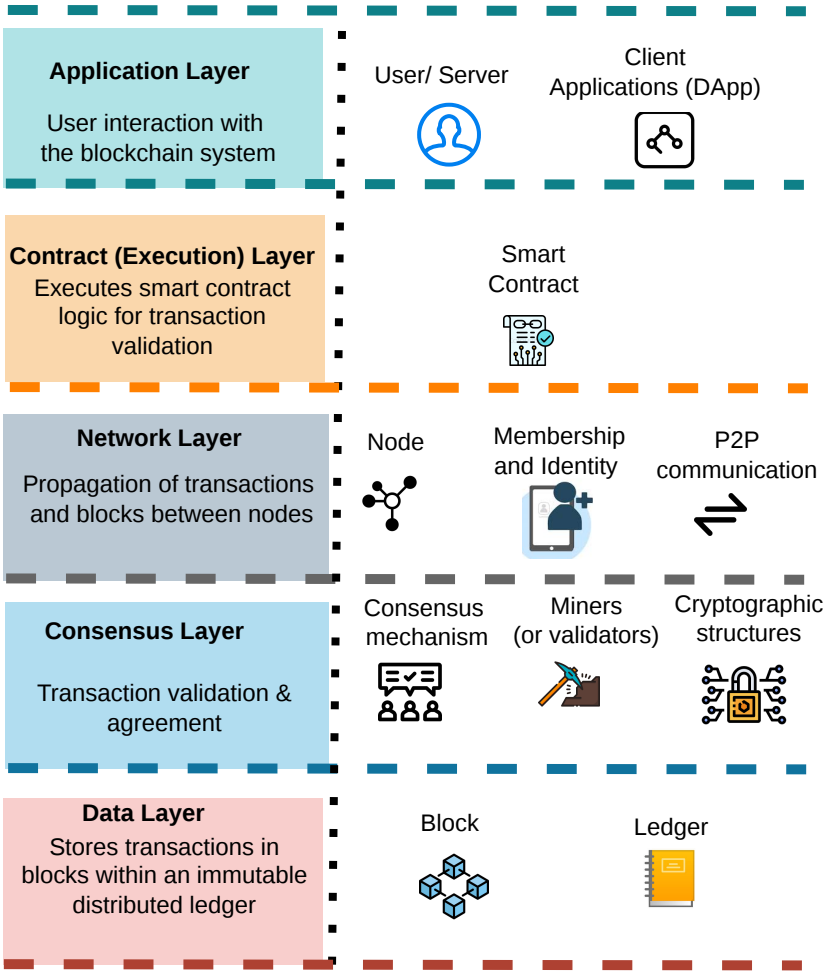


Figure 2.3: **Blockchain Components.** The diagram illustrates the structural layers of a blockchain system, ranging from user-facing applications to the underlying data storage and consensus protocols.

multi-organisation participation. Ripple [80], while not a private blockchain in the traditional sense, operates a permissioned network in which a selected set of trusted validators participate in consensus.

Table 2.2: Comparison of Blockchain Types

Network	Access	Immutable	Efficiency	Centralised	Scalability	Privacy	Cost
Public	All	Yes	Low	No	Low	Low	High
Consortium	Selected	Partial	High	Partial	Medium	High	Low
Private	Single	Partial	High	Partial	Medium	High	Low

These distinctions highlight how blockchain platforms differ in terms of access, consensus participation, scalability, privacy, and efficiency, providing a clear overview of the trade-offs

associated with each type. Considering the requirements of multi-stakeholder autonomous vehicle ecosystems, such as controlled participation, privacy preservation, and flexible governance, permissioned blockchain platforms are generally more suitable than public blockchain systems. Consequently, this research adopts Hyperledger Fabric as the underlying distributed ledger platform, as it provides enterprise-oriented features that support secure and controlled data sharing among multiple organisations.

Hyperledger Fabric (HLF)

HLF is one of the most prominent permissioned distributed ledger platforms, providing a robust environment for smart contract execution. Established in 2015 as an open-source project by the Linux Foundation, HLF offers a modular architecture with components that enable anonymity, flexibility, and scalability by defining clear roles among network participants and executing chaincode.

The way transactions are handled in permissioned blockchains generally falls into two categories:

Order-Execute: This model is a fundamental concept in how blockchains process transactions. The order phase determines the sequence in which transactions are processed, using a consensus mechanism to establish a globally agreed-upon order. The execute phase implements the actions specified in the transactions, updating the blockchain's state accordingly.

Execute-Order-Validate: Unlike traditional order-execute architectures, HLF employs the execute-order-validate (EOV) model, which progresses through three distinct phases: execution, ordering, and validation. This approach improves parallelism and system efficiency by allowing transaction simulation prior to global ordering. As depicted in Figure 2.4, the transaction flow in HLF proceeds as follows:

1. A client application initiates a transaction proposal request via the Software Development Kit (SDK), which constructs the proposal.
2. The proposal is then sent to endorsing peers, which execute a simulation using the chaincode, generate a read/write set and sign the response.
3. The signed response is returned to the SDK, which verifies the signatures. If valid, the SDK gathers the read/write sets and prepares the transaction.
4. The prepared transaction is submitted to the ordering service.
5. Ordering nodes arrange and sequence the transactions into blocks.
6. The generated blocks are sent to committing peers, which validate transactions to ensure they meet endorsement and integrity requirements.
7. Upon successful validation, transactions are committed to the ledger, updating the blockchain state.

This transaction flow relies on key components, including the client application, SDK, endorsing peers, chaincode, ordering service, committing peers and the ledger, ensuring secure and efficient transaction processing in HLF.

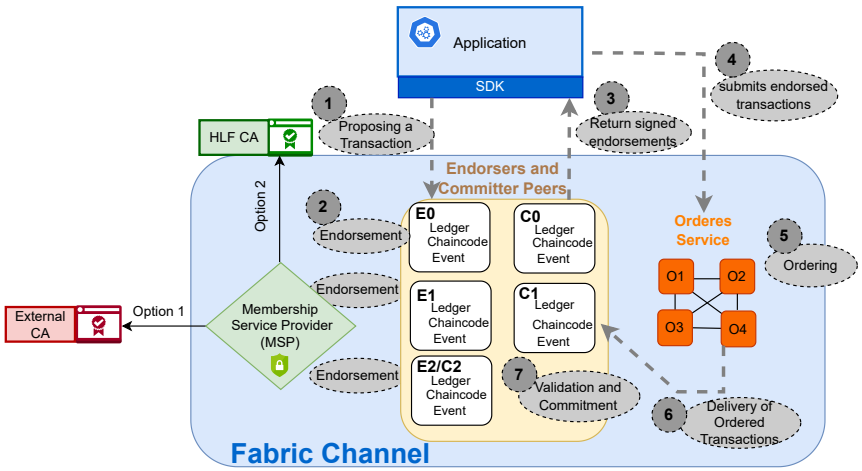


Figure 2.4: **Transaction Flow within a Channel.** The diagram illustrates the sequential steps in the execute-order-validate process, covering proposal submission, endorsement, ordering and transaction validation.

The endorsement policy (EP) is a critical part of this flow, specifying which peers must approve a transaction before it is considered valid. It enforces access control and ensures that only authorised participants can endorse and commit transactions, thereby maintaining trust, security, and integrity across the network.

HLF also introduces the concept of channels, which are private communication pathways within the network. Channels allow a subset of participants to maintain a separate ledger, ensuring the confidentiality of transactions and data while still operating under the shared network infrastructure. Private data collections can further restrict access to sensitive information within a channel, enabling fine-grained privacy control.

This transaction flow relies on key components, including the client application, SDK, endorsing peers, chaincode, ordering service, committing peers, and the ledger, working together to ensure secure, verifiable, and efficient transaction processing in HLF. Overall, these mechanisms collectively enable a flexible and robust permissioned blockchain platform suitable for enterprise and multi-stakeholder environments.

Membership and Identity

Identity and access control across the network are managed through the Membership Service Provider (MSP) mechanism. Each organisation maintains its own MSP, which defines the rules for validating the identities of its nodes and users. Local MSPs identify and validate individual peers or orderers, while channel MSPs define the organisations that are members of a channel and enforce its policies. Thus, MSP must be configured both at the level of

individual peers and orderers to sign transactions, and at the channel level, where it supports validation of identities and signatures from peers, orderers, and clients, ensuring mutual authentication among all members.

Every participant is issued a unique *X.509 digital certificate* by the Hyperledger Fabric Certificate Authority (CA). These certificates serve as the blockchain identities of users and nodes, enabling secure transaction signing and verification. Certificates can include custom attributes, such as department, role, or organisation, which can then be leveraged in chaincode to implement **Attribute-Based Access Control (ABAC)**. These certificates, along with their associated attributes, are stored in SDK wallets and used by the application to sign transaction proposals. Chaincode can inspect these attributes to enforce fine-grained access control policies, ensuring that only authorised participants with the correct attributes can access or modify sensitive data. Endorsement policies rely on MSP definitions to verify which organisations' peers are required to approve a transaction.

Access Control Based on Blockchain

Building on the secure and modular transaction mechanisms of Hyperledger Fabric, blockchain can also serve as a robust platform for enforcing access control policies. Attribute-based access control (ABAC) is a flexible logical access control (AC) model limited only by the computational language and the richness of the attributes available [81]. Integrating ABAC with blockchain enables fine-grained, cryptographically verifiable enforcement of access rights within a permissioned ledger environment. Utilising blockchain for data sharing and access control offers several benefits. First, it enhances data privacy and security by encrypting data and controlling access through cryptographic mechanisms. Second, it provides an auditable and tamper-proof record of data access and sharing activities, promoting transparency and accountability. Third, it enables efficient and trustworthy collaboration among multiple parties, eliminating the need for costly and time-consuming intermediaries.

In the realm of data sharing access control based on blockchain, the ABAC concept corresponds to the notion of utilising attributes to regulate and oversee data access. This approach guarantees that only individuals or entities with the required attributes are authorised to access the shared data. This combination of ABAC and blockchain provides a secure, flexible, and transparent framework for multi-party data sharing in complex networked environments.

2.3 Decentralised Content Distribution Platforms

As data sharing and storage demands grow, decentralised content distribution platforms have emerged as a next-generation evolution of peer-to-peer (P2P) networks. These systems were designed to overcome the limitations of first-generation P2P models, such as BitTorrent and Napster. Key platforms in this space include the InterPlanetary File System (IPFS), Swarm [82], Hypercore [83], SAFE Network [84], Storj [85], and Arweave [86]. Developed over the past decade, these decentralised architectures enhance earlier models like BitTorrent [87] by

improving long-term data availability, enabling more robust file management, supporting dynamic peer participation, and ensuring resistance to censorship.

These platforms commonly leverage key concepts such as content-based addressing, structured overlays, and cryptographic hashing, which collectively enable a decentralised, resilient, and scalable alternative to the traditional client-server paradigm. By distributing data across multiple nodes rather than relying on central servers, these systems enhance fault tolerance, reduce bottlenecks, and promote greater user control over data.

InterPlanetary File System (IPFS)

The IPFS offers advantages over other next-generation architectures due to its content-addressable storage and retrieval model, which fundamentally replaces the traditional client-server web architecture. IPFS splits files into small blocks, assigns each block a unique cryptographic hash (known as a Content Identifier or CID), and distributes these blocks across the network, allowing for efficient data lookup, deduplication, and robust file distribution.

While this distributed approach enhances scalability and content resilience, it introduces additional processing overhead, such as file chunking, hashing, and distributed lookup, which can result in higher latency compared to conventional file transfer protocols (FTPs) [88]. Nonetheless, IPFS has been widely adopted and actively studied due to its applications in areas such as web archiving, the Internet of Things (IoT), and secure data sharing [89]. Its growing popularity is attributed to its ability to reduce storage redundancy, support censorship-resistant file distribution, and seamlessly integrate with blockchain systems to strengthen data integrity, access control, and scalability.

Overall, IPFS exemplifies how decentralised content distribution platforms can overcome traditional storage limitations while providing a flexible foundation for emerging distributed applications.

2.4 Related Work

Nowadays, data plays a foundational role in many sectors, especially in safety-critical environments such as the AV ecosystem, where data generated during operation are vital for various post-event activities involving different stakeholders. This section explores current solutions that support and lead to the proposal of a tailored approach to addressing AV data and the associated stakeholders. It also reviews existing research on AV-related data and highlights the limitations.

2.4.1 Data Sharing Approaches

Ongoing research has explored secure and privacy-preserving methods for data exchange across various domains. Several studies have introduced centralised solutions, such as

the attribute-based authorisation mechanism proposed by Joshi et al., which uses cloud technology for electronic medical records [90]. While their approach and many similar ones enhance expressivity and security, their reliance on centralised storage raises concerns. Centralised solutions introduce single points of failure, scalability limitations and dependency on third-party services.

To address these issues, the literature has seen a shift toward decentralised and cryptographic solutions. Some integrate traditional access control models, such as mandatory access control (MAC), ABAC and discretionary access control (DAC), with newer strategies. For example, Bianchi et al. [91], combined identity-based encryption (IBE) with linear secret sharing (LSS) to enforce cryptographic conditions. Their framework ensures that only parties complying with predefined policies can decrypt shared datasets. Similarly, a secret-sharing-based approach, PRISM [92], has been proposed to compute private set operations over outsourced databases in a multi-party environment.

In another new shift, blockchain has been applied to data sharing in multiple data-sensitive sectors, demonstrating benefits for security immutability and auditability. For example, in healthcare, HLF-based platforms have been developed to manage electronic health records (EHR) while ensuring access control and traceability [42]. In their model, Al Shalali et al. encrypt pointers to medical records, storing them on the blockchain, while the actual data remain within healthcare databases. Similarly, Guo et al. [93] introduced a hybrid system in which edge nodes store EHR data and implement ABAC. Healthcare providers retrieve access control lists (ACLs) from the blockchain, which contain links to EHR data, ensuring that only authorised users can access records.

In industrial IoT environments, blockchain combined with attribute-based encryption has been leveraged to enable fine-grained decentralised access control for secure resource sharing such as in [44, 94, 95]. In addition, attribute-based encryption (ABE) has been integrated with blockchain to enhance access control and confidentiality, such as in the works of Alniamy and Taylor [41], Ma et al. [96], and Liu et al. [97]. Moreover, integrating blockchain with IPFS has been proposed for scalable decentralised file storage and integrity verification [98, 99, 100, 101].

In summary, existing research demonstrates a variety of approaches for secure and decentralised data sharing, including blockchain, attribute-based encryption, and hybrid storage solutions. However, these studies also reveal limitations such as scalability constraints, architectural complexity, and dependence on centralised components. These gaps highlight the need for a tailored framework that combines blockchain, access control mechanisms, and decentralised storage to provide secure, efficient, and auditable data sharing across multiple stakeholders.

2.4.2 Data Management in Autonomous Vehicles (AVs)

Effective data management is critical in AV systems due to the large volume and variety of data generated. From real-time decision-making and environmental perception to long-term

learning and system optimisation, the performance and safety of AVs are fundamentally dependent on the availability, integrity and timely processing of data. Numerous solutions have been suggested for handling the vast volumes of data used by AVs, including in-vehicle data storage, cloud-based infrastructures, edge computing technologies and real-time data sharing.

Aiming to preserve privacy in such a data-sensitive environment, several studies have proposed architectures that safeguard sensitive user information. For example, Gheisari et al. [17] presented a context-aware privacy-preserving framework leveraging software-defined networking (SDN) and differential privacy, dynamically adjusting data aggregation based on sensitivity. Parekh et al. [16] integrated blockchain and federated learning to preserve data confidentiality during distributed training, demonstrating the utility of ledger-based trust frameworks. Zhao et al. [19] proposed a lightweight affine masking approach for secure vehicle-cloud collaboration, reflecting the attention paid in this study to securing cloud-based offloading.

When addressing security challenges, researchers have proposed several architectures to ensure data integrity, confidentiality and authentication. From a security perspective, Anwar et al. [9] offered a dynamic risk assessment and mitigation approach for V2V communication, integrating game theory and security decay modelling. Moreover, various methods have recently been proposed that integrate BC technology in IoV systems, in order to enhance security. For example, Dwivedi et al. [102] used a consensus mechanism to provide a secure event-sharing protocol for smart cities using IoV and RSU. Similarly, Yi et al. [103] reported approaches aiming to protect IoV against quantum attacks. Liu et al. [104], designed a blockchain-based mechanism to provide security-related data collection and incentivise mobile nodes.

In the context of privacy-preserving data sharing, several studies have introduced privacy-preserving machine learning (PPML) techniques, such as federated learning (FL), secure multi-party computation, homomorphic encryption, and differential privacy, to facilitate collaborative model training without exposing sensitive data. These methods, when applied in the context of AVs, allow the development of shared machine learning models using large, diverse datasets, including those collected from geographically dispersed or previously unvisited locations. For instance, a framework proposed by Zeng et al. [105] leveraged federated learning, supported by large-scale wireless connectivity, to collaboratively train the control models of CAVs. To address challenges such as variable participation rates, heterogeneous data quality and dynamic network conditions, the authors introduced an algorithm that considers CAV mobility, wireless channel fading and non-independent and unbalanced data distributions.

In another study, He et al. [106] introduced a blockchain-based federated learning system for connected and autonomous vehicles (Bift) as a decentralised machine learning framework that combines federated learning with blockchain and the IPFS to enhance privacy and security during the data-sharing process among CAVs. This system ensures data integrity and confidentiality throughout the learning pipeline.

Similarly, Xiong et. al [107] presented an edge-assisted paradigm for privacy-preserving raw data sharing. The authors proposed a privacy-preserving convolutional neural network (P-CNN) architecture utilising additive secret sharing to securely encrypt raw data while

maintaining model accuracy.

These contributions highlight diverse strategies, ranging from federated learning and cryptographic techniques to blockchain-based systems, for enabling secure and collaborative data sharing in AV environments. While these works demonstrated progress in addressing specific privacy and security concerns, they primarily focused on enabling secure data sharing for collaborative machine learning and model training purposes, usually among CAVs or intra-network nodes, like vehicles, edge servers, or cloud systems. In other words, they contributed to addressing privacy and security concerns related to data sharing in distributed ML contexts but they did not fully deal with the broader and more complex issue of data sharing among diverse multi-stakeholders. Researchers have also turned their attention to data sharing for forensic purposes, where the objective is not collaboration but accountability, evidence preservation, and post-incident analysis. Forensic investigation and data integrity techniques in AVs have particularly attracted significant research attention. As summarised in Table 2.3, various solutions have been proposed for crash reconstruction solutions, for example, safety models like causal analysis based on system theory (CAST) [108] and the functional resonance analysis method (FRAM) [109]. CAST employs system theory to identify causes of failure and propose preventive measures. FRAM assesses complex interactions in socio-technical systems. However, these models lack a unified framework that covers all causal factors, and manual analysis remains costly and inefficient. The forensic event data recorder (FEDR) [36] was developed to meet investigators' data requirements independently of manufacturers. This solution aims to provide an EDR that meets all requirements but only from the investigator's perspective.

Data integrity preservation for investigation purposes in AV has been a goal of several studies that have presented frameworks based on various forms of technology. Hoque and Hasan proposed a forensic investigation framework for AVs called the AVGuard tool [38], designed for integration with the AV system. The framework assumes that the AV has local storage to store the log provenance while also communicating with a remote cloud server to publish the newly created log provenance. A robot operating system (ROS) node collects all the logs from different AV modules.

Oham et al. proposed a distributed digital forensics framework [20], which is based on the evidence reported by nearby witness vehicles if a vehicle is involved in an accident. Digital signatures, along with a corresponding certificate, are used to protect data integrity. Data exchanges between entities in the framework are stored in a blockchain and used for later decision-making.

In another approach, T-Box [23] is a trusted real-time automotive data-recording system comprising a network monitor, generator and recorder. It assumes that the gateway could be used as a network monitor. The generator reconstructs data provided by the monitor and delivers it to the recorder, which stores the data. The recorder stores an individual data entry in a block. These data blocks can then be stored locally or externally or transmitted to a remote server.

Buquerin et al. [37] offered a general concept for automotive forensics. Using Ethernet, their implementation uses the onboard diagnostics interface, the diagnostics over internet protocol, as well as the unified diagnostic services for communication. Liu et al.'s study aimed to store EDR data safely and away from manipulation. In their scheme [24], data are

sent to the manufacturer's server, as usual, but the vehicle also uploads the EDR data to a cloud server and sends the evidence of storage to the nearby vehicle through a vehicular ad hoc network.

Despite these advances, the solutions reviewed in this section exhibit limitations in terms of storage granularity and access mechanisms. In other words, while they may succeed in enhancing current data-recording tools in terms of integrity, storage or both, they do not, within the scope of the reviewed approaches, provide comprehensive availability guarantees or fine-grained access control mechanisms suitable for multi-stakeholder environments. A recent advance in forensic AV data handling is AVChain [25], which integrates blockchain with IPFS to enable scalable, secure and verifiable crash data. However, while this solution provides an access mechanism, it follows a one-directional access model, lacking the flexibility required for fine-grained, policy-based, multi-party data interactions. This means that although data can be verified and retrieved, the framework does not fully support a flexible, policy-based multi-party data-sharing model that simultaneously maintains security and scalability, particularly for post-storage access scenarios.

To conclude, large volumes of historical and forensic data require secure, auditable and scalable multi-party sharing beyond immediate transaction validation. In addition, AV data are much beyond forensic data while most of these solutions focus on EDR data. Furthermore, existing solutions discussed in the literature place limited emphasis on multi-stakeholder governance, long-term data integrity, and off-chain data storage for non-real-time AV data sharing, with most efforts focusing primarily on forensic or event-specific datasets. In addition, access control mechanisms supporting adaptive, policy-driven management across multiple organisations remain comparatively underexplored in the surveyed literature.

These identified gaps in the existing literature motivated this work, which aims to design a blockchain-based framework tailored to secure, flexible and scalable sharing of non-real-time AV data. Our approach tackles the challenges of multi-organisational collaboration, access policy enforcement, data ownership and integration with off-chain data storage, providing a secure and efficient solution for comprehensive AV data management.

2.5 Summary and Research Gap

AVs generate vast amounts of data, encompassing raw and processed inputs from cameras and sensors, along with communication exchanges with other vehicles and RSUs. This aggregated dataset is vital for understanding the context of significant events. However, there are two primary challenges related to data management and accessibility. First, existing systems, such as EDRs and DSSAD in AVs are often inadequate, even for basic accident-related scenarios [34]. To address these shortcomings, various solutions have been proposed [38, 20, 37, 23, 24], focused on enhancing data integrity by assigning different levels of trust to recorded data.

Second, the need for AV data spans several sectors, drawing interest from a wide range of stakeholders, including regulatory bodies, legal authorities, OEMs, suppliers, vehicle owners, insurance companies, testing agencies, road operators and researchers [39]. For instance, sensor data play a crucial role in diagnostics, enabling AVs to assess both internal system health and external environmental conditions [40]. This is especially important given

Table 2.3: Comparison of Solutions for Accident Data Retrieval and Integrity.

Category	Solution	Key Idea	Strengths	Limitations
Safety Analysis Models	CAST [108]	System-theoretic accident analysis	Systematic failure identification	High cost; lacks unified framework
	FRAM [109]	Socio-technical interaction analysis	Handles complex systems	High cost; lacks unified framework
Forensic Investigation Frameworks	AVGuard Tool [38]	ROS-based log collection to cloud	Modular logging support	Assumes reliable local storage
	Distributed DF Framework [20]	Blockchain-based evidence sharing	Tamper-proof evidence	Requires nearby AVs; network overhead
	AV Data Pipeline [110]	Sensor data pipeline for crash reconstruction	High-fidelity reconstruction	Privacy and security concerns
Trusted Data Recording	T-Box [23]	Real-time vehicle data recording	Reliable operation	Limited privacy support
	Generalised Automotive Forensics [37]	Ethernet-based diagnostics	Efficient communication	Limited implementation details
Forensic Data Integrity	Safe EDR Storage [24]	Cloud-based EDR with data sharing	Prevents tampering	Depends on VANETs
Integrity with Access Control	AVChain [25]	Blockchain + IPFS data sharing	Integrity and access control	Complex; limited real-time capability
	FEDR [36]	Independent forensic data collection	Supports investigations	Requires large-scale adoption

the limitations of current event recording systems and the complex issues of liability and verification in autonomous driving [111]. Moreover, AV data are indispensable for analysis [112], training deep learning models [113], and informing regulatory development [114].

Ensuring that these data are accessible to authorised parties is critical to support continuous system improvement. This necessitates fine-grained access control mechanisms that allow secure and responsible data sharing with relevant stakeholders. Accordingly, this research investigates the effectiveness of tailored blockchain-based data-sharing approaches to help bridge the gaps identified above.

2.6 Conclusion

This chapter provided a comprehensive overview of autonomous vehicle technology, its key data challenges, and a review of existing literature. It highlighted the significant limitations in current data-sharing and storage solutions, specifically in regard to data integrity, security, and ownership. The analysis of these gaps led to the identification of Distributed Ledger Technology (DLT) and blockchain as a promising approach to overcome these issues. The foundational concepts of DLT, including cryptographic principles and consensus mechanisms, were explored as a basis for developing a new, more robust system. This foundational review directly informs the system requirements and design principles that will be detailed in the subsequent chapters of this thesis.

Chapter 3

Research Methodology and the Proposed Solution

This chapter presents the methodology and design of the proposed solution developed for secure data sharing using HLF. The methodology in this study followed an experimental research approach. It included problem identification, objective definition, design and development of the proposed system, and subsequent demonstration and evaluation. The core of this work was the design and implementation of smart contracts and the manipulation of endorsement policies within a permissioned blockchain environment. Accordingly, this chapter contributes to addressing **RQ1a**, **RQ1b**, and **RQ1c** by identifying the required components, tools, and mechanisms; clarifying the types, sources, and flow of data within the system; and demonstrating how the framework can be implemented to integrate these elements in line with the design requirements. In summary, this research was conducted through a systematic, experimental, multi-phase methodology. The chapter begins by outlining the research methodology and tasks 3.1 used to guide the system design, implementation and evaluation. Following this, it details the proposed system architecture and the conceptual design 3.2, including the components involved, as well as articulating the rationale behind key design decisions. The chapter then concludes with a summary 3.4, addressing the chapter's key points.

3.1 Methodology Tasks

As illustrated in Figure 3.1, the research was conducted through a sequence of interrelated tasks, each forming a phase of the experimental methodology. These tasks collectively guided the process from problem identification to system evaluation and documentation. The tasks are described in detail below:

Task 1: Thematic Literature Review and Gap Analysis

This task formed the foundation for Chapter 2, which discussed the background to this thesis, and involved a comprehensive thematic literature review and gap analysis. **The findings of this review have been published in [62]**. The literature was systematically categorised into six key themes to capture the multifaceted landscape of data sharing and storage in the context of AVs:

1. **AV technologies and existing survey research**, encompassing studies on the current state-of-the-art in AV systems and broader survey efforts synthesising progress across domains.
2. **Non-blockchain data sharing mechanisms** covering traditional and models that rely on trusted third parties, federated systems, or other distributed architectures.
3. **Blockchain-based solutions for data sharing** focusing on decentralised approaches, consensus mechanisms, and smart contracts aimed at improving data integrity, security and traceability.
4. **Cloud and data storage management approaches** investigating the role of cloud computing, edge computing and hybrid infrastructures in managing large-scale AV data.
5. **AV-specific data-sharing frameworks** exploring tailored protocols and architectures designed specifically for autonomous driving environments, vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication.
6. **Security and privacy concerns in data sharing**, addressing critical issues such as data confidentiality, access control, identity management, and regulatory compliance.

Within each theme, studies were critically reviewed to gain a deep understanding of the current landscape of AV data sharing and storage. This critical review involved several layers of analysis:

1. Understand current methodologies and technologies used in data sharing and storage.
2. Identify trends, overlaps between approaches and shortcomings.
3. Map out unresolved challenges in AV fields related to secure data sharing, data integrity and traceability, and ownership.

These layers of analysis ensured that the review was not merely descriptive but critically evaluative, providing a robust foundation for addressing the research problem.

Subsequently, these insights were critical in defining the upcoming tasks. *This task directly contributed to RQ1 by establishing the components, mechanisms and gaps that shaped the design principles of the proposed solution.*

Task 2: Defining the Problem

Based on the thematic review of literature on existing solutions, several critical gaps and limitations in current AV data-sharing and integrity mechanisms were identified. These included limitations in how AV operational data are securely stored, managed, and shared among multiple stakeholders. **The main gap identified has been published in [43]** and discussed in the previous chapter. These findings were constructed to define the core research problem, namely, how effective a blockchain-based system is as a platform that ensures secure and traceable data sharing for AVs. *This task further aligned with RQ1 by framing the problem around secure multi-party data-sharing design.*

Task 3: Testbed of Existing Solutions

A blockchain-based solution proposed by Budel et al. [115] was deployed in a testbed environment to evaluate state-of-the-art solutions and explore the effectiveness of the solution in ensuring data integrity, availability, and access control under realistic road trial conditions. Key insights were gained into the practical challenges of incident data management by observing its limitations. These observations helped identify the technical and functional requirements for the proposed system, highlighting gaps in current approaches. *This task contributed to RQ1 by identifying the strengths and weaknesses of existing systems that informed the requirements for the proposed framework.*

Task 4: Defining the Requirements

Drawing on both empirical observations from the local testbed and insights gained through the literature review, a comprehensive set of technical and functional requirements was formulated. These requirements served as the foundation for designing the proposed blockchain-based solution, as detailed later in this chapter (see 3.2.1). The goal was to ensure the architecture would directly address the critical limitations identified in existing systems while meeting the operational needs of secure AV data sharing. *This directly supported RQ1 by formalising the requirements needed to build a secure and scalable framework.*

Task 5: Designing a Blockchain-Based Conceptual Model

A conceptual architecture was designed that incorporated blockchain (HLF) and decentralised storage (IPFS). The design was intended to mitigate the shortcomings of existing solutions by ensuring secure, auditable and tamper-resistant data exchange, while also supporting decentralised data access. This task is detailed in section 3.2. **The design methodology and architecture have been published in [116].** *This task addressed RQ1 by proposing the architecture and identifying the design principles of the framework.*

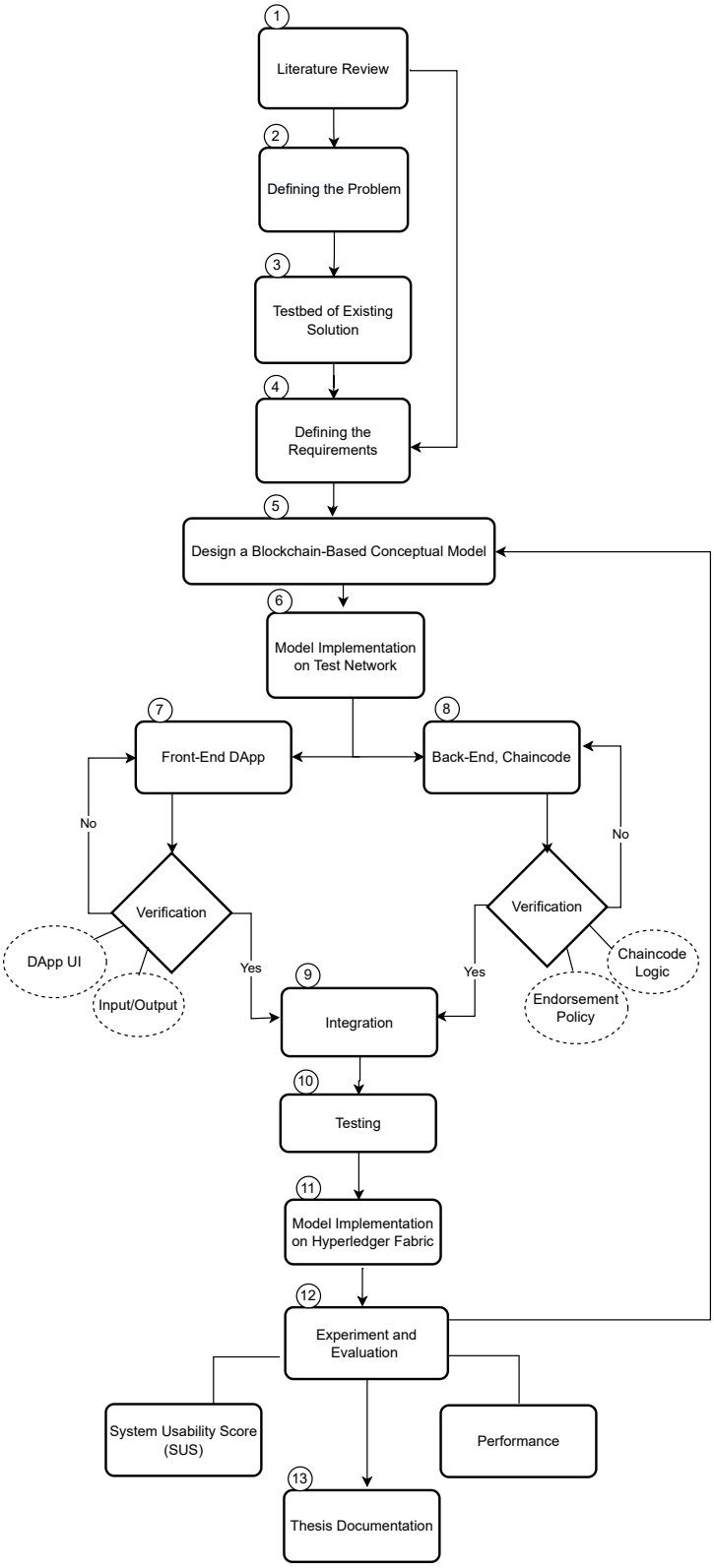


Figure 3.1: Research Methodology.

Task 6: Prototype Implementation on a Local Test Network

The conceptual model was first implemented in a controlled, local test environment to validate key aspects of the system prior to full-scale deployment. This task involved the following:

1. Verifying the functional correctness of individual system modules.
2. Testing inter-module communication and data flow.
3. Identifying and resolving architectural or logical inconsistencies at an early stage.

This task is thoroughly discussed in Chapter 4. In addition, *it contributed to RQ1 by demonstrating how the defined components could be implemented in practice.*

Task 7: Frontend DApp Development

A frontend decentralised application (DApp) was built to provide a user interface for interacting with the blockchain system. Functionality was rigorously tested through input/output validation to ensure correct handling of user interactions and blockchain responses. This task is comprehensively elaborated upon in the frontend implementation section in Chapter 4 section 4.4. *This task addressed RQ1 (framework components) and contributed towards RQ3 (usability considerations), which were later evaluated in Chapter 5.*

Task 8: Backend and Chaincode Development

Smart contracts were implemented as a chaincode to enforce access control policies, validate data hashes and integrity proofs, and interact with the ledger state. The chaincode played a central role in upholding data trust and security guarantees. The detailed implementation is set out in Chapter 4 (see section 4.6). *This task contributed to RQ1 by showing how smart contracts could be implemented to ensure secure data sharing.*

Task 9: Integration

Following the successful verification of both frontend and backend components, the system was fully integrated. Integration testing was conducted to ensure seamless interoperability between components, transactional consistency and correct execution of workflows across the blockchain network. This task is presented in Chapter 4 (see section 4.7). *This task addressed RQ1 by validating the integration of framework components into a functional system.*

Task 10: Testing

The fully integrated system underwent comprehensive testing to identify any residual bugs, validate inter-component interactions and confirm the system's ability to enforce defined

access policies and data integrity rules. This phase ensured the solution's robustness and readiness for realistic deployment. *Testing task contributed to RQ2 by establishing the baseline performance of the proposed system.*

Task 11: Model Implementation on Hyperledger Fabric (HLF)

Following successful validation in the development environment, the system was deployed on a distributed HLF test network hosted on Amazon Web Services (AWS). This deployment allowed the emulation of realistic operating conditions, enabling assessment of the system's scalability, fault tolerance and performance in a cloud-based distributed setting more representative of a real-world deployment scenario. This task is detailed in Chapter 4 (see sections 4.2 and 4.5) and *directly contributed to RQ2 by testing scalability and performance in realistic settings.*

Task 12: Experiment and Evaluation

A series of experiments was conducted to assess the system's performance under different conditions. Key performance metrics, such as transaction throughput and latency, were measured. Additionally, a System Usability Scale (SUS) survey was conducted to capture user feedback and assess the perceived usability and effectiveness of the application. This task is presented in depth in Chapter 5. In addition, it *contributed to RQ2 (performance metrics) and RQ3 (usability evaluation).*

Tasks 7 to 12 are also part of the contributions published in [116]

Task 13: Configuration Variants and Tailoring for AV Use Case

To thoroughly evaluate the proposed architecture, multiple configurations were designed and tested, each reflecting a distinct combination of system parameters tailored to the AV data-sharing context. These configurations varied in terms of endorsement policies to align with realistic use case requirements. The experimental setup and evaluations for these configurations are detailed in Chapters 6 and 7. **This work and all its sub-tasks, such as the verification tools, are part of the contributions published in [117]**. *This task directly addressed RQ2 by assessing scalability limitations and the effects of endorsement policy variations.*

Task 14: Thesis Documentation

All phases, from problem definition to evaluation and findings, were documented. The outcomes were formulated into this thesis and relevant publications, reflecting the academic and practical contributions of the work.

3.2 Conceptual Design of the System

The conceptual design aimed to address limitations identified in existing AV data-sharing approaches, particularly the lack of fine-grained access control and secure multi-stakeholder data management. Subsequently, the requirements and component mapping are identified and described in this section.

3.2.1 System Requirements

To design a secure and scalable data-sharing platform for autonomous vehicles, it is essential to first identify the system requirements. These requirements define the fundamental properties and constraints that the system must satisfy in order to ensure trust and efficient operation. They serve as the foundation for mapping system components and guiding the overall architectural design. These requirements were derived from the findings of the thematic literature review and gap analysis presented earlier in this chapter (Task 1) 3.1, together with empirical insights obtained from the evaluation of existing solutions in the testbed environment (Task 3), as discussed in (Task 4). The key requirements are as follows:

1. Immutability: Ensuring data consistency, versioning, and protection against tampering through immutable records. This is particularly important in a multi-stakeholder environment, so that participants can ensure accountability and verifiability.

2. Transparency: Transparency ensures that authorised stakeholders can verify data transactions and system operations. In a multi-stakeholder environment such as AV ecosystems, transparency improves accountability by allowing participants to observe how data are registered, accessed and managed within the system.

3. Traceability:

   Traceability enables the system to maintain an auditable history of all data-related operations, including data submission, access requests and policy updates. This capability allows stakeholders to track the origin and usage of data, supporting auditing, accountability and regulatory compliance.

4. Security: Including robust authentication, encryption and fine-grained access control to safeguard sensitive AV data. Since the data may include stored AV datasets such as sensor logs, accident reports, or other post-operation records, they must be protected against unauthorised access. A secure system ensures that only approved stakeholders can retrieve or contribute data based on defined permissions.

5. Scalability: AVs generate large volumes of data, and the system must scale to support these data, and multiple users and organisations.

6. Reliability: The goal is to design high availability and resilient distributed computing. For stakeholders to depend on the system for decision making, the platform must ensure that data are consistently available. Reliability supports continuous operations and fosters trust in the platform's robustness.

7. User-Friendliness: Prioritising intuitive user interfaces, responsive design and effective error handling to support end users and system operators is critical. Given the diversity of stakeholders, including non-technical users, a user-friendly interface is needed to ensure accessibility and smooth interaction with the system, reducing training overhead and minimising human error.

   These requirements directly inform the selection of system components and technologies, as described in the following subsection.

3.2.2 System Components and Architecture

The proposed system architecture consisted of three main modules, each selected based on its ability to meet the system requirements efficiently, as summarised in Table 3.1. The table presents a mapping between each requirement and the component(s) responsible for its implementation. The components were chosen after an in-depth study of current alternatives, focusing on the specific needs of secure, scalable and decentralised data sharing, illustrated in Figure 3.2 and described below.

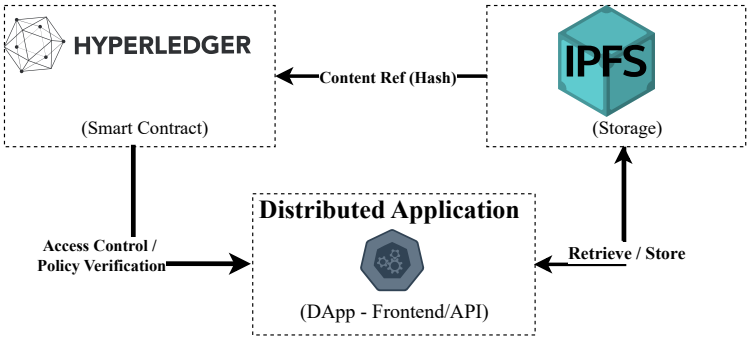


Figure 3.2: System Components.

- **Web Module:** A DApp enabling stakeholders, each with specific attributes, to register and interact with the system, performing actions such as uploading or requesting data. This module provides an intuitive interface essential for usability in AV ecosystems. A web-based DApp was preferred over a traditional platform to ensure platform independence, streamlined updates and seamless integration with blockchain networks.

- **Storage Module:** Utilising the IPFS to store large datasets securely and scalably, the data are indexed using content-addressable hashes. IPFS is adopted instead of centralised cloud storage solutions as it provides a decentralised, tamper-resistant, and scalable file-sharing mechanism, making it suitable for environments involving multiple, potentially untrusted stakeholders [118]. Compared to alternative decentralised storage systems, such as Swarm [82], and BitTorrent [87], the IPFS offers a more mature implementation, better community support and easier integration with blockchain platforms [89]. Its use of content-addressable storage enables built-in tamper detection, with any change in file content resulting in a new unique hash.

- **Network Module:** In the architecture built on HLF, this module ensures decentralisation, access control and immutable record-keeping. Smart contracts (chaincode) manage permissions and data integrity.

  HLF was selected for this work based on a layered rationale aligned with the system requirements. First, permissioned blockchains were selected over public or permissionless approaches, such as Ethereum, due to the nature of AV ecosystems. These ecosystems typically involve a consortium of trusted or semi-trusted stakeholders, including, for example, regulators, manufacturers, insurers and suppliers, who require controlled access and collaborative governance. This differs from a permissionless blockchain, which allows unrestricted participation. In addition, public blockchain suffers from higher latency and vulnerability to network attacks, whereas permissioned systems enforce strict membership rules and role definitions, significantly enhancing security and resilience [119]. This is an important quality in handling data of a sensitive nature such as in AVs. The model chosen supports scalability, privacy and efficient access control, all of which are crucial for AV data exchange scenarios [120, 121].

  Second, among permissioned platforms, HLF was chosen due to its superior performance characteristics in scalable and robust environments. Several studies have highlighted the adaptability of HLF in diverse industrial use cases, demonstrating its ability to support fine-grained access control and complex trust models [119, 122]. Unlike other permissioned platforms, HLF offers native support for pluggable consensus, channel-based privacy and scalable transaction processing, making it particularly well-suited for decentralised applications that require both performance and strong governance [121].

  Third, within the Hyperledger ecosystem, Fabric was selected over alternatives like Sawtooth and Indy due to its modular architecture, higher throughput and support for selective data visibility. In addition, Fabric offers a broader balance of enterprise-relevant features, including crash fault-tolerant ordering, fine-grained access policies and efficient transaction processing [120]. This makes Fabric the most suitable choice for complex, multi-stakeholder systems requiring both performance and security.

  In addition, Execute-Order-Validate (EOV) architecture is a core differentiating feature that sets Fabric apart from other permissioned platforms, such as Corda, Quorum and Besu, and also other Hyperledger options, such as Sawtooth. In multi-stakeholder environments like AV ecosystems, this architecture enhances system scalability and resilience by enabling parallel transaction execution and isolating faults. This separation allows faults or invalid transactions to be isolated before they can affect the ledger state. It also reduces the risk of non-deterministic transaction failures because its design forces smart contracts to be deterministic, ensuring all peers produce the same output.

Table 3.1: Mapping System Requirements to Components.

Requirement	**Mapped Component(s)**
Immutability	Network module (HLF, smart contracts); storage module (IPFS hash referencing)
Transparency	Network module (HLF distributed ledger providing auditable transaction visibility)
Traceability	Network module (HLF distributed ledger maintaining immutable transaction history); storage module (IPFS hash referencing)
Security	Network module (Attribute-Based Access Control, encryption)
Scalability	Storage module (distributed IPFS); network module (modular architecture of HLF)
Reliability	Storage module (replication in IPFS); network module (resilient peer-based infrastructure)
User-Friendliness	Web module (frontend DApp with user role awareness and responsive interaction)

3.2.3 Scenario and Use Cases

This section presents a representative scenario and detailed use cases that demonstrate how different system components and actors interact within the implemented architecture. These use cases highlight the application of ABAC, IPFS and HLF features in supporting secure, auditable access and data management.

Motivation Scenario

The motivation scenario presented in Figure 3.3 outlines the development of an integrated system that enables controlled access for stakeholders.

1. Different Data Sources: Vehicles generate diverse types of data, including V2I, V2V, V2P, V2X and V2N. Collecting these data from multiple sources provides a rich, multi-dimensional picture of vehicular environments.

2. Data Collecting and Storing: The collected data, both raw and processed by the AV system, cloud infrastructure, or edge computing, are stored in backend storage systems.

3. Potential Data Stakeholders: Various stakeholders, such as manufacturers, suppliers, regulators, insurance companies, researchers and accident investigators, require access to these data for diverse purposes, including vehicle innovation, safety analysis, regulation enforcement, liability investigation, risk assessment and academic research.

4. Data-Sharing Platform: Since multiple organisations need access to the data, it is crucial to have multi-organisational stakeholder engagement and fine-grained data access control. This ensures that each entity can access the data set required for their

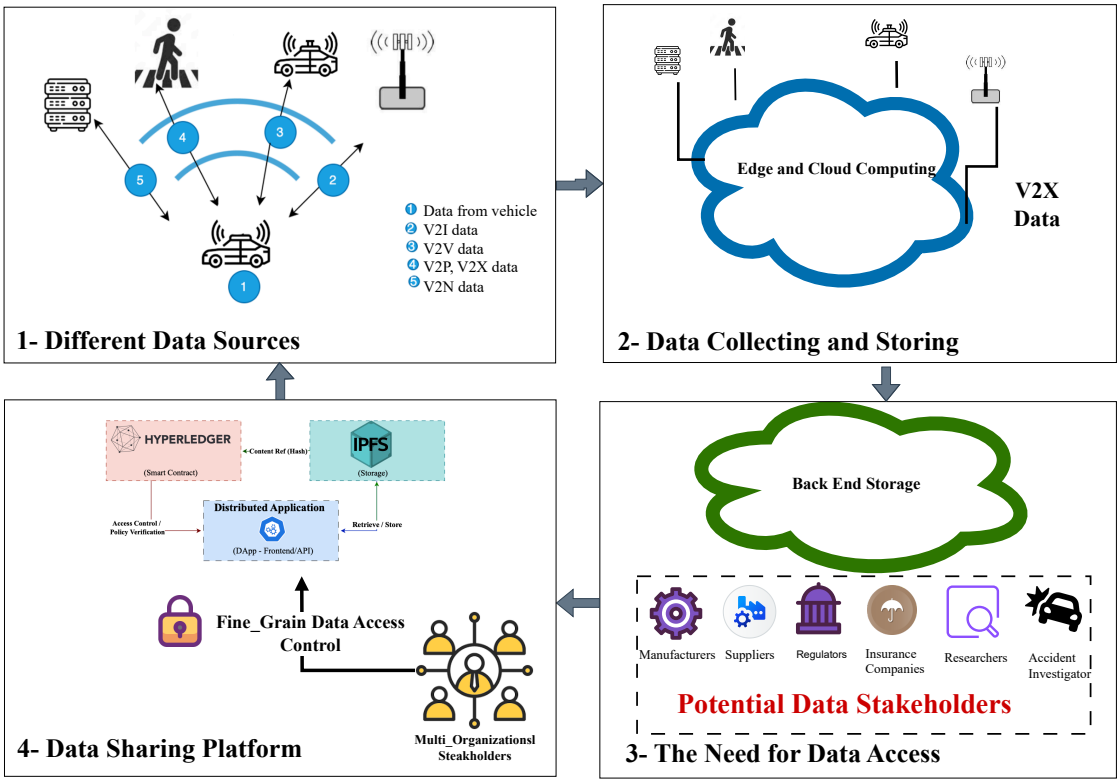


Figure 3.3: **Motivation Scenario.** The figure illustrates the need for a multi-stakeholder ecosystem where (1) diverse vehicular data, such as V2I, V2V, V2P, V2N and V2X, are collected from autonomous systems and (2) stored via cloud infrastructures. (3) Several stakeholders, such as manufacturers, insurers, regulators and researchers, need these data for different purposes. (4) A dedicated data-sharing platform enables secure, fine-grained access control for stakeholders, providing insights that drive continuous technological improvements and creating a feedback loop that enhances vehicle systems technology, safety and policy development.

role without compromising privacy or security. To facilitate secure, controlled and efficient sharing of vehicle data among multiple stakeholders, a dedicated data-sharing platform is necessary. It acts as an intermediary, ensuring appropriate access and data governance.

5. Feedback Loop: By providing controlled access to vehicular data, the platform empowers stakeholders, such as manufacturers, researchers and suppliers, to analyse real-world driving conditions, communication patterns and accident data. Insights gained from data analysis fuel the development of improved vehicle systems, enhanced safety protocols, more efficient communication technologies, and advanced driver assistance features. Subsequently, the improved technologies are deployed back into vehicles, and new and richer data are generated from updated sensors and communication mechanisms. This ongoing loop results in iterative refinement of vehicle technologies and data analytics methods, fostering innovation and improving safety, efficiency and performance standards.

As a result, each participant in the ecosystem benefits. For example, manufacturers create superior products, regulators craft better policies, insurers can more accurately price risk, researchers gain access to real data and consumers enjoy safer, smarter mobility.

Scenario Overview

In the proposed platform, stakeholders representing various organisations require a secure and decentralised mechanism to share AV data. The system enables them to register and upload datasets to the IPFS, define fine-grained access policies and retrieve datasets via approved access. All operations are performed through a web interface and recorded immutably on the HLF blockchain, ensuring traceability and compliance. To better illustrate the functionality of the proposed platform, the following subsections describe representative use cases and interactions between system actors. These use cases capture the roles of administrators, stakeholders, and the underlying modules, highlighting how access control, data sharing, and blockchain integration are achieved in practice. Each use case is presented in the form of diagrams and step-by-step flows to clearly demonstrate the operational behaviour of the system.

Actor-Centric Use Case

1. **Use Case 1:** Figure 3.4 presents the Actor-Centric Use Case Diagram for Administration, which comprises the following:

   - *Actors:* System administrator (admin).

   - *Description:* Admin approves the registration of new stakeholders and issues digital certificates using the CA.

   - *Flow:*

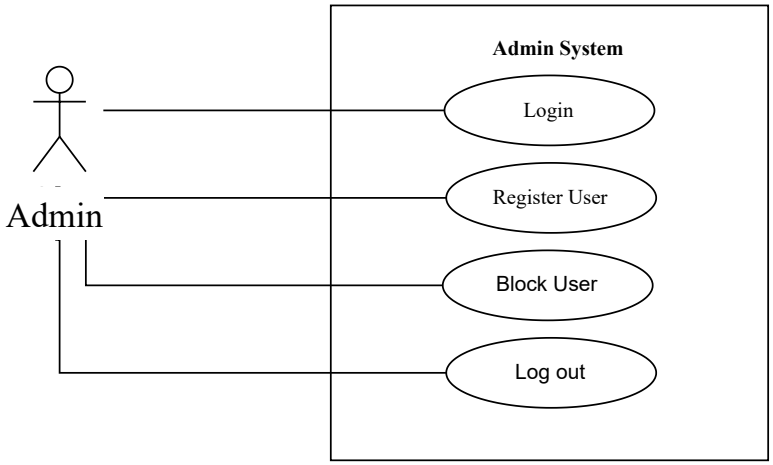


Figure 3.4: **Actor-Centric Use Case Diagram for Administration.** This diagram illustrates the interactions of the "admin" actor with core system functionalities.

(a) Admin logs in via the frontend and accesses the pending registrations list.
(b) Admin reviews the user details and selects either "Approve" or "Reject".
(c) If approved, the backend sends the registration request to the Fabric CA.
(d) If rejected, a notification is sent to the user.
(e) Admin can block or remove users if needed.
(f) Admin initiates logout, and the system invalidates the session.

2. **Use Case 2:** Figure 3.5 illustrates the Actor-Centric Use Case Diagram for the User, which is as follows:

   - *Actors:* System user.

   - *Description:* The system user interacts with the platform to perform essential actions, such as registering, logging in, uploading files to the IPFS, assigning or updating access policies, requesting access to data and logging out. These actions are governed by attribute-based access control enforced by the HLF.

   - *Flow:*

(a) The user visits the platform and submits a registration request by filling out personal and role-specific details.
(b) After approval from admin, the user receives login credentials.
(c) The user logs into the system using email and a password.
(d) Upon successful authentication, the system grants access to the dashboard.
(e) The user can:
   - Upload files to the IPFS and define access control policies.
   - View previously uploaded data and associated metadata.

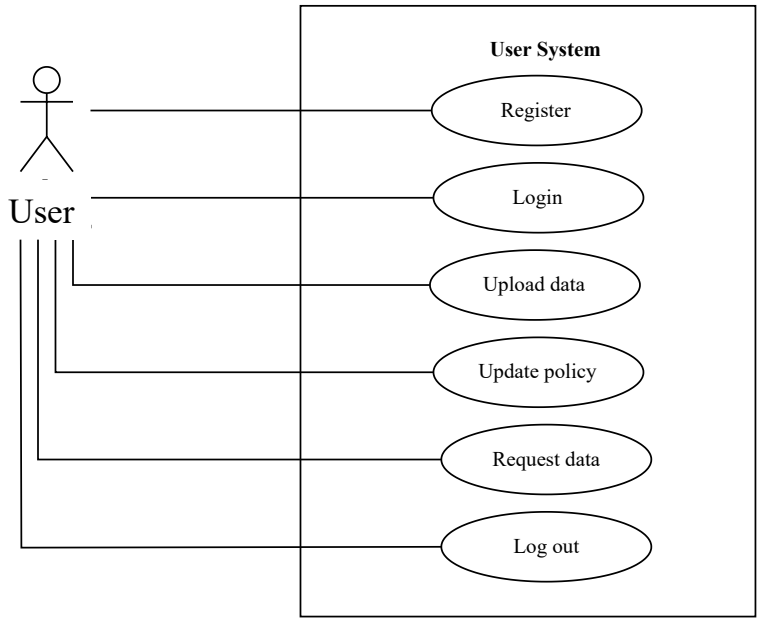


Figure 3.5: **Actor-Centric Use Case Diagram for the User**. This diagram illustrates the interactions of the "user" actor with core system functionalities.

- Request access to a specific dataset.
- Receive permission (or denial) based on ABAC policy evaluation.
- Retrieve approved files from the IPFS using the hash provided.

(f) The user logs out and the system ends the session securely.

Functional Use Case

Figure 3.6 presents the use case diagram for the proposed system, capturing the main interactions between users (new, authenticated, admin) and the system functions. The diagram outlines key functionalities, such as registration, digital certificate issuance, dataset upload, policy management and data access. Notably, the Data Retrieval use case is conditionally triggered upon a Data Access Request only if the user is authorised. In addition, the interaction with an external blockchain CA is also included to highlight the integration of digital identity management.

1. **Use Case 1: User Registration Request Submission**

   Actors: New user
   Description: A new user submits their details through a registration form. The request is stored as pending until reviewed by admin.
   Preconditions: The system is live and accessible to the public.
   Flow:

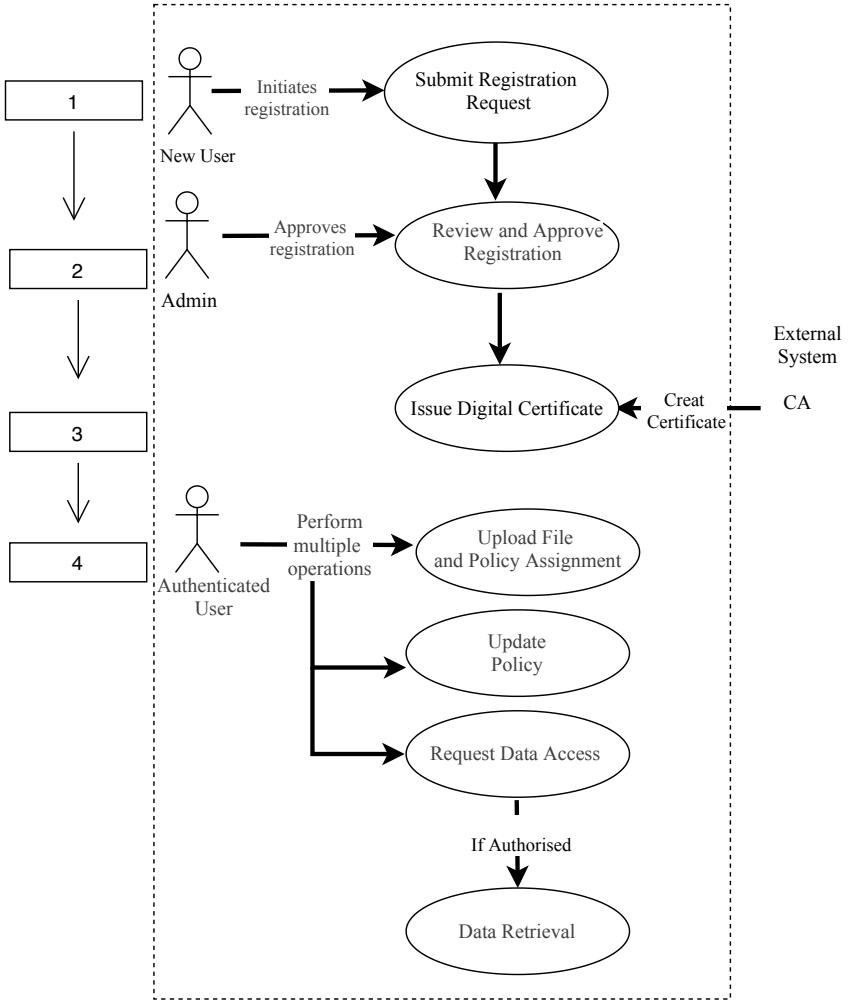


Figure 3.6: **Functional Use Case Diagram.** The figure illustrates the system design and workflow.

(a) The user visits the registration page.

(b) The user completes the following information.
- username
- email
- password
- organisation
- role or attribute.

(c) The user submits the form.

(d) The system stores the request in a "pending" queue/database.

2. **Use Case 2: User Registration and Identity Provisioning**

Actors: System administrator, certificate authority (CA)
Description: A system administrator approves the registration of new stakeholders and issues digital certificates using the Fabric CA.
Preconditions: - The Fabric CA is active.
- Admin is authenticated.
Flow:

(a) Admin logs into the frontend and accesses the pending registrations list.

(b) Admin reviews the user details and selects either "Approve" or "Reject".

(c) If approved, the backend sends the request to the Fabric CA. Then the CA is responsible for:

i. validating the identity of the stakeholder.

ii. issuing the corresponding digital certificate.

iii. and updating the Membership Service Provider (MSP).

(d) If rejected, the system sends a rejection notification to the user.

3. **Use Case 3: File Upload and Policy Assignment**

Actors: Authenticated stakeholder user
Description: A registered user uploads a file to the IPFS and assigns access control policies based on their roles or attributes, along with metadata.
Preconditions: The user is registered and authenticated.
Flow:

(a) The user selects a file and defines access policies via the frontend.

(b) The file is uploaded to the IPFS, generating a unique hash.

(c) IPFS Hash (CID) along with the Metadata (File description, Timestamp, Uploader ID, Access control policy) are submitted to the blockchain.

(d) Chaincode validates the user's attributes and writes the metadata on-chain.

4. **Use Case 4: Data Access Request**

Actors: Registered stakeholder
Description: A user requests access to a specific dataset and is granted access if their attributes match the access policy stored on-chain.
Preconditions: File metadata and access policies exist on the blockchain.
Flow:

(a) The user browses the available datasets and requests access to the needed one.

(b) Chaincode evaluates the request based on:

- The user's certificate attributes.

- The defined access policy for the data.

(c) If authorised, the system returns the IPFS hash.

(d) If not authorised, the system sends a notification.

5. **Use Case 5: Data Retrieval**

   Actors: Authorised stakeholder
   Description: After being granted access, a user can retrieve a file using the IPFS hash.
   Preconditions: The user holds valid authorisation.
   Flow:

   (a) The system returns the IPFS hash and directs the user to the IPFS interface.

   (b) The IPFS returns the requested content.

6. **Use Case 6: Policy Update by User**

   Actors: Authenticated stakeholder
   Description: The data owner user modifies the access policy of an already uploaded file.
   Preconditions: The user is authenticated and owns the dataset.
   Flow:

   (a) The user logs in and selects the file to manage.

   (b) The user defines a new access policy, such as modifying roles or organisations allowed access.

   (c) The system verifies the user's rights (using Attribute-Based Access Control, checking roles and ownership embedded in the user certificate).

   (d) The system updates the access policy securely on-chain by invoking the chaincode, which stores the updated policy in the ledger state.

Access Control Use Case Flow

This use case illustrates the conceptual design of the integration of ABAC into a permissioned blockchain system to enforce fine-grained access to sensitive data.

Actors: Authenticated stakeholder
Description: This use case describes how the ABAC model is incorporated into the system design to manage user permissions dynamically.
Preconditions: - The user has a digital identity issued by the CA and a set of attributes (organisation, role, level) associated with it.

- Access policies for each dataset have been defined by the owner and stored on-chain.
Flow:

1. The user interacts with the DApp to request access to a specific dataset. This request includes the user's identity and the requested resource.
2. The system extracts attributes (role, org, level) from the user's X.509 certificate via Fabric APIs.
3. Access policies are logical rules encoded in chaincode; for example, access to the (`LiDAR Front 01`) sensor data may be restricted to users with the role of *investigator* in Org2.
4. If the attribute conditions match the access policy, the user is granted access. If not, access is denied.
5. If access is granted, a content-addressed hash (not the actual data) is returned. The user can then retrieve the data from the IPFS using that hash, adhering to the key design principles of secure, off-chain storage and verifiable integrity.

3.3 Implementation Scope of This Research

This thesis builds on the blockchain infrastructure provided by HLF and its standard deployment templates. While the underlying blockchain network components (e.g., peers, ordering service, and membership services) follow the architecture provided by the framework, several key elements of the proposed system were designed and implemented as part of this research. These include the development of the smart contracts responsible for data integrity and access policy management (e.g., *Save_hash*, *Git_ipfs_hash*, and *Update_Policy*), the integration with off-chain storage using IPFS, and the design of customised endorsement policy configurations. In addition, the application logic and experimental evaluation scenarios were implemented specifically for this work. Performance benchmarking was conducted using the framework provided by Hyperledger Caliper, while the workload configurations and test scenarios were designed for the proposed system. The implementation details of these components are described in Chapter 4.

3.4 Conclusion

This chapter presents the conceptual design of the proposed access control system, which combines blockchain, the IPFS and DAap, together with privileges from the ABAC mechanisms, to ensure secure and verifiable data sharing. The design is structured to enable interoperability between the blockchain ledger, off-chain data storage and access control logic. In addition, the chapter describes the phased methodology followed during the system's development and has outlined the core design requirements derived from the literature. It has also mapped system components to the defined requirements and introduced a representative use case to demonstrate how attribute-based access control is conceptually embedded within the system. This design chapter provides the foundation for the implementation and evaluation stages, which will be covered in the following chapters.

Chapter 4

System Implementation

Based on the principles of system design illustrated in the previous chapter, this chapter presents a comprehensive description of the proposed system implementation. It details the system components, modules and key algorithms that together propose secure and access-controlled data sharing across multiple stakeholders. Specifically, this chapter addresses Research Question **RQ1d** by demonstrating how the identified components are integrated and implemented to meet the design requirements. It focuses on the implementation of the components developed in this research, as outlined in Section 3.3, including the smart contracts, application logic, and system integration.

This chapter begins with 4.1, which provides an overview of the system architecture. It continues with the Technology Stack and Deployment Details (4.2), followed by the System Workflow (4.3) describing the system processes. Next, Frontend Implementation (4.4) details the frontend design, while Blockchain Network Configuration (4.5) and Chaincode Implementation (4.6) explain the blockchain setup and smart contract development. The chapter also covers Integration and Interactions (4.7) between components, describes the Data Set (4.8). In addition, the benchmarking tools setup is presented in 4.9, including Caliper Configuration and Benchmarking Setup 4.9.1) and Performance Evaluation Using k6 4.9.2. Then, the key assumptions are outlined in (4.10). Finally, the chapter concludes with the Key Challenges (4.11) faced during implementation.

4.1 Functional Architecture

To achieve the study objectives, the system was implemented based on the requirements for scalability, security, usability and seamless integration with blockchain and distributed storage components. This section addresses the high-level system architecture, illustrating the key components as follows:

- **Frontend (React.js).** Users interact with the platform via a React-based web interface that facilitates data input, file upload and transaction submission. It provides real-time feedback and validates user inputs before sending requests to the backend. React.js was chosen because it enables responsive, component-based user interfaces and provides efficiency

in handling dynamic updates, which aligns with the system's usability and performance requirements.

- **Backend (Node.js).** This acts as middleware, processing frontend requests, validating data and permissions, and invoking HLF chaincode through Fabric SDK. It also manages communication with the MySQL database and the IPFS network. Node.js was selected for the backend due to its asynchronous, non-blocking I/O model, which supports scalability and efficient handling of multiple concurrent requests. Its alignment with the chaincode language also simplifies development and integration.

- **Hyperledger Fabric (HLF) Network.** This comprises five organisational peers, orderers and a CA. It enforces multi-organisation endorsement policies and executes chaincode (smart contracts) to maintain an immutable ledger of transactions, user roles and file metadata. Hyperledger Fabric is used as the underlying blockchain platform, providing permissioned identity management, endorsement policies, and smart contract execution.

- **Access Control Model.** The system adopts an Attribute-Based Access Control (ABAC) model, implemented through Hyperledger Fabric's support for X.509 certificate attributes. Access decisions are enforced within the chaincode by evaluating attributes such as user role and organisational affiliation at the time of each transaction. This approach enables flexible and fine-grained control over data access in a multi-organisational environment. Furthermore, access policies are defined by the data owner during data upload and can be updated dynamically, ensuring that control over shared data remains decentralised and adaptable to changing requirements. The implementation defines three levels of roles inside the organisation: *admin*, *employee* , and *investigator*. These roles were selected to model a simplified yet realistic hierarchy of stakeholders involved in AV data sharing. The *admin* represents system governance and identity management, the *employee* represents data producers within organisations, and the *investigator* represents auditing entities requesting access.

- **Chaincode (Smart Contracts).** Implemented in JavaScript (Node.js) and running within Docker containers on peers, this enforces the ABAC dynamically by extracting roles/attributes from user certificates and validating permissions during transactions. Chaincode was developed in Node.js to maintain consistency with the backend technology stack, reducing integration complexity and enabling rapid development while ensuring strong enforcement of access control policies.

- **InterPlanetary File System (IPFS, via Pinata).** This comprises a decentralised file storage system in which uploaded files are stored and pinned to guarantee availability. The IPFS Content Identifier (CID) is recorded on-chain, providing tamper-proof references to the actual content. IPFS is used for decentralised off-chain storage of datasets, where files are stored and referenced via content identifiers (CIDs). Pinata was used for reliable pinning services to guarantee file persistence.

- **MySQL Database.** Relational off-chain database holding transaction metadata and file references to complement ledger queries. While IPFS provides decentralised, content-addressable storage for files, it does not support structured data management or application-level state handling. In this system, MySQL is used to manage application data, such as user registration requests and frontend related information that is not appropriate to store on-chain. This separation ensures that the blockchain is reserved for immutable and verifiable records (e.g. IPFS hashes), while MySQL supports lightweight, mutable application logic. As such, the database acts as a complementary component to improve system organisation and usability, without affecting the trust model or decentralisation of the core data-sharing process

This architecture facilitates secure, transparent, and auditable data management without centralised control, meeting the research goals of decentralisation, access control, and multi-organisational governance.

4.2 Technology Stack and Deployment Details

Following on from the previous section, which presented the functional architecture of the system, this section focuses on the concrete technologies, frameworks and infrastructure used to build, configure and deploy each component.

Programming Languages

1. **JavaScript / Node.js v18+:** Used for backend services and chaincode logic due to its asynchronous architecture and native support in the HLF SDK.
2. **React.js:** Selected for the frontend to deliver a responsive and interactive user experience.

Frameworks and Tools

1. **Hyperledger Fabric (HLF) v2.5.10:** The primary blockchain framework for permissioned data sharing and smart contract execution. It is configured with five organisations and tailored endorsement policies.
2. **InterPlanetary File System (IPFS):** Used for storing uploaded datasets off-chain, ensuring content integrity and decentralisation.
3. **Pinata:** A pinning service used with the IPFS to guarantee file availability.
4. **MySQL:** Employed for maintaining off-chain metadata and improving search efficiency.
5. **Docker V20.10+:** A container platform for consistent deployment of all system components.

6. **Docker Compose V1.29+:** It is a tool built on top of Docker that enables defining and managing multi-container applications using a single YAML file. It simplifies orchestration by allowing the start and stop of multiple containers together, configure networks, volumes, and environment variables in one place.
7. **Caliper v2.5.0:** It is a Performance benchmarking tool used to evaluate blockchain throughput and latency under varying workloads.
8. **K6:** It is an HTTP/REST API load testing. K6 scripts can simulate user scenarios hitting REST endpoints, monitor response times, and identify bottlenecks under high user loads.

Infrastructure and Platforms

1. **Operating System:** Ubuntu Linux (20.04 LTS), running on AWS EC2 instances.
2. **Hosting Environment:** Amazon EC2 (`t3a.2xlarge`, 8 vCPUs, 32 GB RAM) used for hosting the Fabric network, web server and database.
3. **Storage:** AWS Elastic Block Store (EBS) for persistent data storage for application data and Docker volumes.

The implementation of the proposed system is built using open-source technologies and official frameworks. In particular, the blockchain layer is developed using Hyperledger Fabric and its official Node.js SDK and chaincode APIs.

The network configuration and initial deployment were based on the official Fabric sample network (`fabric-samples/test-network`), which was adapted and extended to support the multi-organisation architecture and customised endorsement policies required in this research.

4.3 System Workflow

Figure 4.1 presents the system workflow as a sequence diagram, illustrating the interactions between system components across three main phases: (i) user registration and identity setup, (ii) data upload and blockchain transaction processing, and (iii) data access and retrieval. The following provides a detailed step-by-step explanation aligned with the numbered sequence in the figure.

Registration and Identity Setup

- (1) The user initiates the process by submitting a registration request through the client application. The system supports two main categories of users: a system administrator and authorised stakeholders. The system administrator is responsible for managing enrolment and identity issuance through the Hyperledger Fabric Certificate Authority (CA), while stakeholders interact with the platform to upload, share, request, and retrieve data. Each user is associated with an X.509 digital certificate, and the attributes

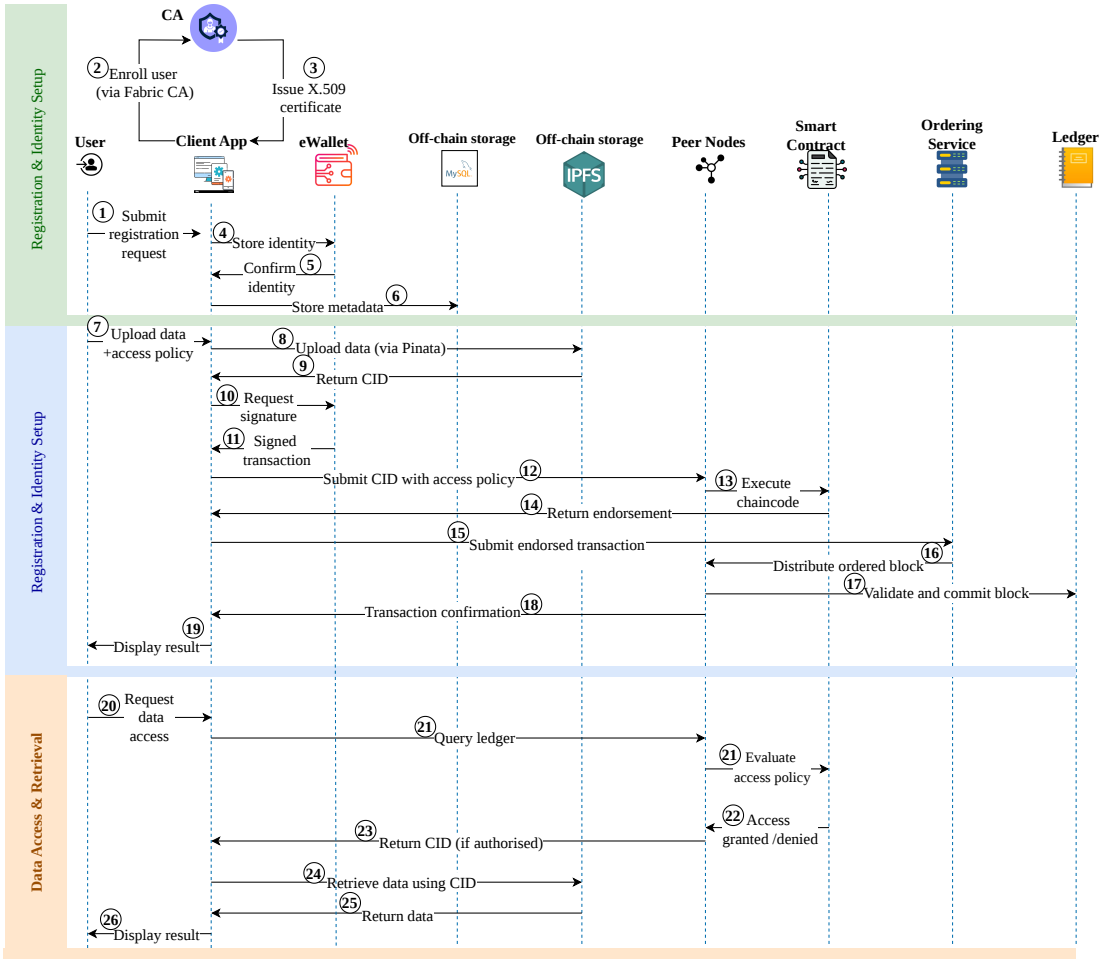


Figure 4.1: **System Workflow.** diagram illustrating the end-to-end operation of the proposed system, including user enrolment via Hyperledger Fabric CA, off-chain data storage using IPFS, on-chain transaction processing and access control enforcement through Hyperledger Fabric, and authorised data retrieval.

embedded in this certificate are used by the chaincode to enforce attribute-based access control (ABAC).

- (2–3) The client application enrols the user via the Hyperledger Fabric Certificate Authority (CA), which issues an X.509 digital certificate representing the user's identity.
- (4–6) The issued credentials are stored in the user's electronic wallet (eWallet), enabling secure transaction signing. Relevant metadata may also be stored in the off-chain database (MySQL).

Data Upload and Blockchain Transaction Processing

- (7–9) The user uploads data through the client application, which stores the file in IPFS via Pinata. IPFS returns a unique Content Identifier (CID) representing the uploaded data.
- (10–11) The client application requests a transaction signature from the eWallet, and a signed transaction is returned.
- (12–14) The client submits a transaction containing the CID and associated access control policy to the peer nodes, where the chaincode is executed and the transaction is endorsed.
- (15–17) The endorsed transaction is forwarded to the ordering service, which orders transactions into blocks and distributes them to peer nodes for validation and commitment to the ledger.
- (18–19) Once committed, a transaction confirmation is returned to the client application, and the result is displayed to the user.

Data Access and Retrieval

- (20–21) A user requests access to a dataset by submitting a query containing the data identifier. The client application forwards the request to the peer nodes to query the ledger.
- (21–22) The chaincode evaluates the request by enforcing attribute-based access control (ABAC), checking the attributes embedded in the user's digital certificate.
- (23) If the request is authorised, the corresponding CID is returned to the client application.
- (24–25) The client uses the CID to retrieve the data directly from IPFS.
- (26) The retrieved data is presented to the user.

Overall, the workflow demonstrates the integration of off-chain storage and on-chain verification to support secure and efficient data sharing.

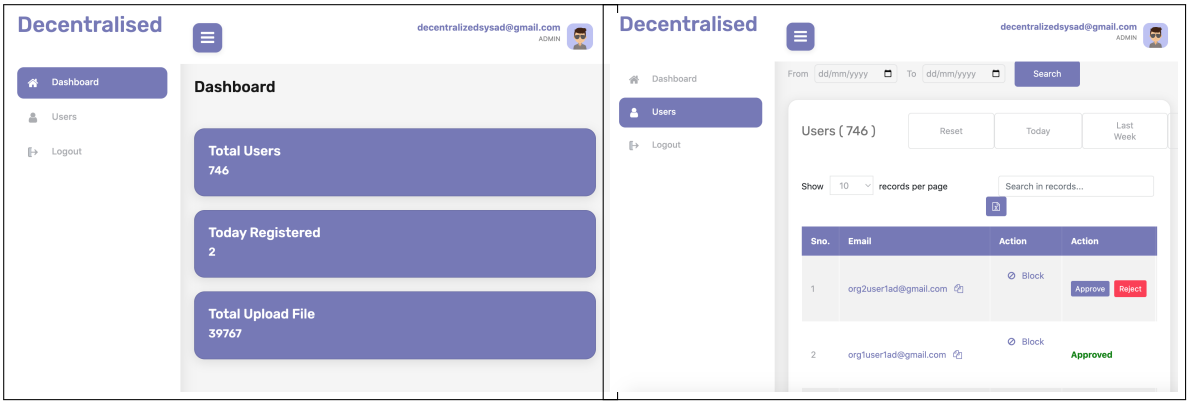
4.4 Frontend Implementation

The frontend of the system is developed using React.js, a widely adopted JavaScript library for building dynamic and responsive web interfaces. The user interface serves as the primary entry point for stakeholders to interact with the decentralised system and offers features such as user registration, login, data upload with access policies, viewing for all available datasets, and submission of access requests. React's component-based architecture enables efficient rendering and state management, making it ideal for integrating with backend APIs and blockchain events in real time. The rest of this section presents the interface for each use case in the system.

Admin Dashboard Interface

The admin dashboard is a secure, role-restricted interface exclusively accessible to the system administrator. It provides a control panel to manage and authorise user registrations, which acts as a gatekeeping mechanism before invoking the *Register_User* chaincode function deployed on the HLF network. This function is responsible for securely writing the user identity and associating it with their attributes (username, role, organisation) in coordination with HLF CA to generate the user's digital certificate. The internal process by which the administrator verifies the authority or legitimacy of users and stakeholders is considered outside the scope of this work.

Upon user sign-up, the request is stored in a pending state in the backend database (MySQL). Pending requests are displayed in a table within the dashboard, as illustrated in Figure 4.2, showing the username. The admin reviews each request and chooses to approve or deny it: **approve** triggers backend logic to contact the HLF CA, register the user, enrol them and generate a digital identity certificate; **deny** marks the request as rejected, no certificate is issued, and the user is notified of the decision.



(a) Summary view of the admin dashboard (b) Pending and approved requests

Figure 4.2: Admin Dashboard.

User Authentication Interface

The user authentication flow is designed to ensure secure entry into the system. Users initiate the registration and login process through a dedicated form-based interface. Upon registration, users provide identifying details and select their organisation and role. Users are processed by the backend to invoke identity creation on Fabric CA. The user interface supports both new registrations and existing user login, connected to Node.js backend services. Once successfully authenticated, a session is established, and the users will be able to react with the authenticated user interface, giving them access to all functionalities, as shown in Figure 4.3:

- Dashboard: Shows a summary of the files uploaded.
- Data Upload: Where the user uploads a dataset.
- Data List: Where the user can view all the available datasets uploaded by other stakeholders.
- My Data List: Where the user can explore and make changes to all their own uploaded data.
- Logout: Where the user logs out and the session ends.

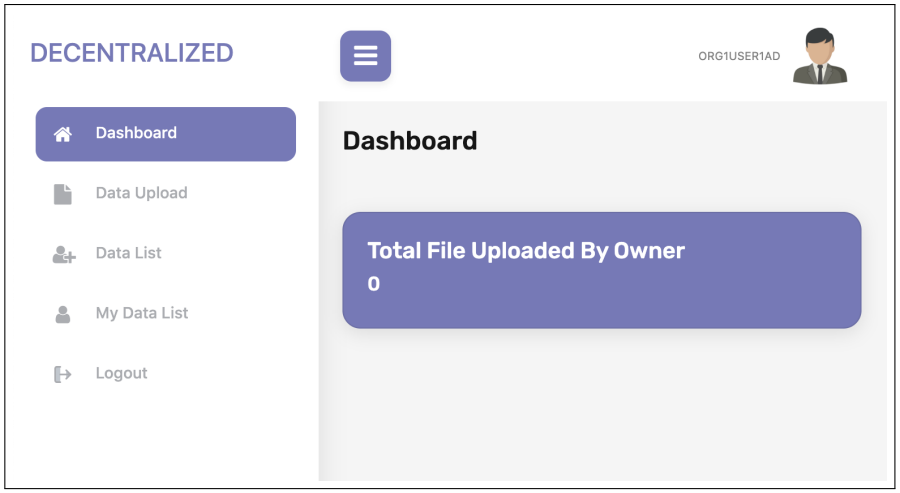


Figure 4.3: Authenticated User Interface After Login.

Data Upload Interface

Authenticated users can submit their encrypted files and define custom access control policies for the data. As shown in Figure 4.4, the uploading interface includes fields for file selection as follows:

- Title: A short name or label for the dataset.
- Data Type: Specifies the nature of the data, such as the sensor name or category.
- Description: Additional details, including sensor ID, location, or any relevant metadata.

- Access Policy: Defines the access permissions, specifying which organisations and role levels within those organisations are allowed to access the data.

These sections are validated before passing to the backend for processing and submission to the IPFS and blockchain.

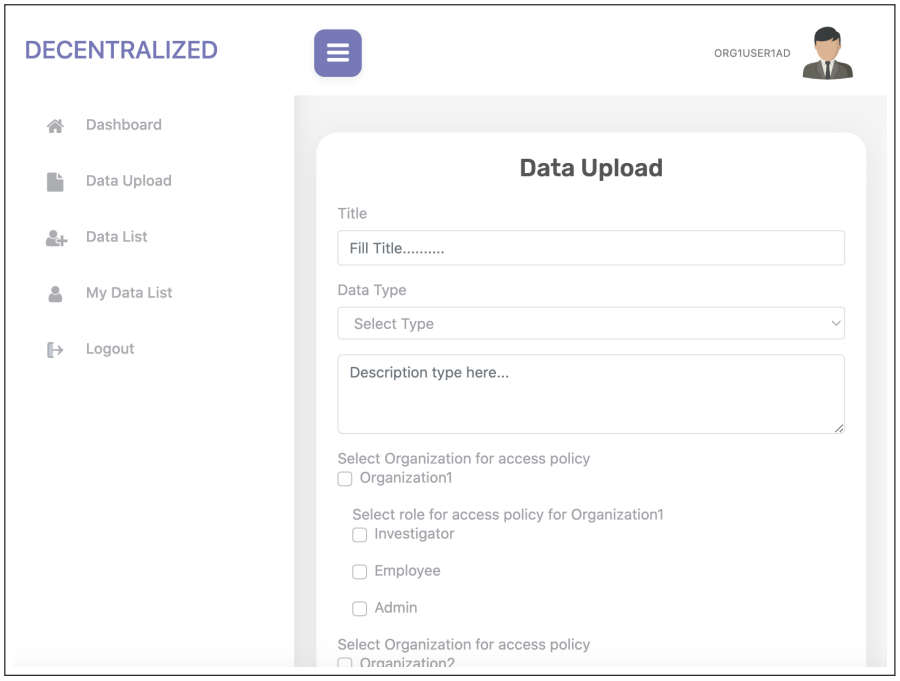


Figure 4.4: User Interface for Uploading Data.

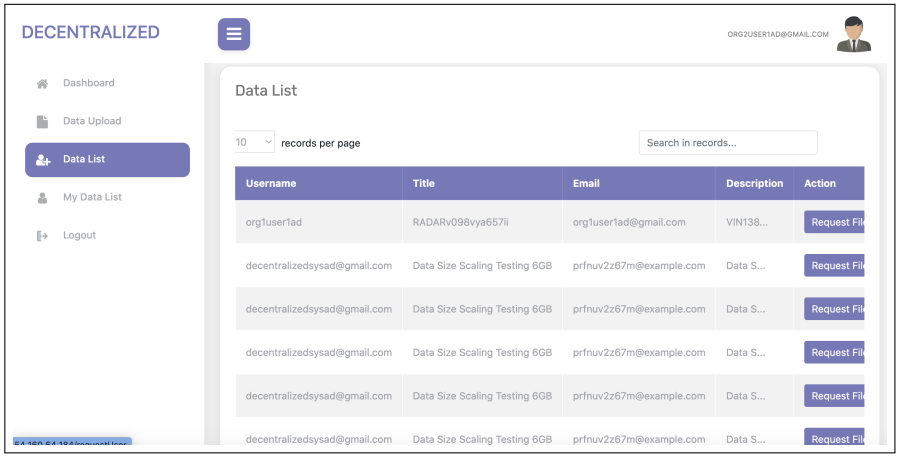


Figure 4.5: Dashboard View: Displaying Available Data Lists for Authenticated Users.

Dataset Discovery and Access Request

Users can browse a list of available datasets shared by others within the system. The frontend queries both the blockchain and the off-chain MySQL database to retrieve file metadata and determine access permissions. As shown in Figure 4.5, each dataset entry

includes an access request button. If a user meets the required ABAC rules, the IPFS hash is returned to the frontend for data retrieval.

Rejected Data Request Interface

When a user submits a request to access a dataset in the interface previously shown, it is processed by chaincode logic based on ABAC policies. If the request is rejected due to a policy violation (e.g. missing required attribute, unapproved organisation), a notification of the decision is displayed to the user, as illustrated in Figure 4.6

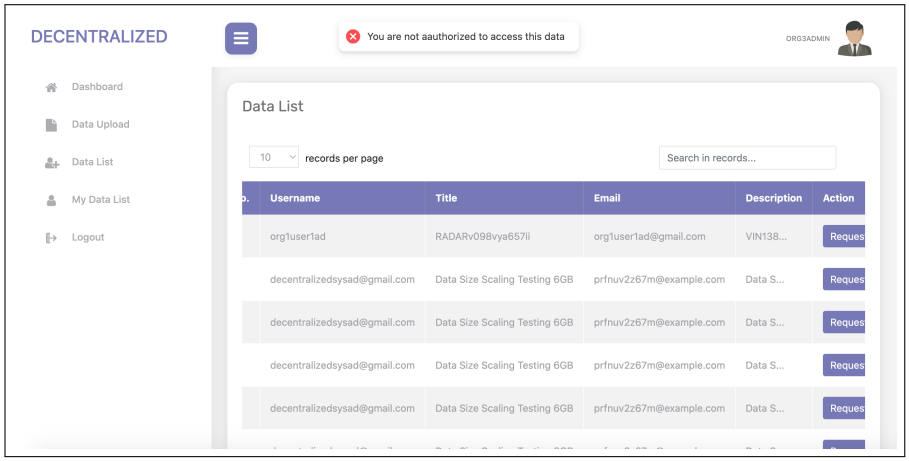


Figure 4.6: Rejected Data Request Interface and User Notification

These frontend components, supported by client-side validation and interactive design, ensure a seamless and secure user experience. The above figures illustrate the key elements of the user journey, from registration to data upload and policy definition.

4.5 Blockchain Network Configuration

This section outlines the configuration of the HLF network used in the system, including the organisations, peers, ordering service, consensus mechanism, state database and identity management components. The network was deployed on a Linux/amd64 environment using Docker containers, running HLF version 2.5.10 (Commit SHA: 1a81040), with Go version 1.23.1. The Docker base label was `org.hyperledger.fabric`, within the official Hyperledger namespace. These details ensure consistency with the recommended Fabric standards.

4.5.1 Peers and Organisations

The HLF network configuration evolved in two phases:

- **Phase 1.** A minimal setup of **three organisations**, each hosting a single peer node.

This configuration was used for initial functional testing and baseline performance evaluation.

- **Phase 2.** Based on the insights gained in Phase 1, the hardware and software components were upgraded, and the network was scaled up to five organisations, each still maintaining one peer node. This more realistic configuration reflects typical multi-organisational collaborations and was used for evaluating performance and scalability under broader access and endorsement scenarios.

All peer nodes act as both:

- **Endorsing peers**, which execute chaincode and endorse transactions, depending on the defined endorsement policy.
- **Committing peers**, which validate and commit transactions to the ledger.

This setup ensures decentralisation of responsibility while maintaining a manageable infrastructure for testing. Although each organisation is limited to one peer for resource efficiency, in production environments, multiple peers per organisation are recommended for fault tolerance, resilience, and load balancing.

4.5.2 Ordering Service

The network uses a single orderer node configured with the *Raft consensus protocol*. Although Raft supports crash fault tolerance when deployed across multiple orderer nodes, this system utilises a single-orderer configuration. This choice is justified by the controlled testing environment and the experimental nature of the system. The primary goal of the deployment is to assess system functionality, integration and performance under realistic but manageable conditions, rather than to achieve full production-grade fault tolerance. Deploying a single orderer significantly simplifies network setup and reduces operational overhead, which is particularly beneficial during iterative development phases. It also lowers resource consumption, a key consideration when using cloud-based infrastructure with limited computational resources. However, it is important to mention that this setup introduces a potential single point of failure. In a production scenario, this limitation can be mitigated by scaling the ordering service to a multi-node Raft configuration, enabling high availability and fault tolerance. Additionally, the use of a single-orderer deployment may result in optimistic latency and throughput measurements, as it does not incur the communication and consensus overhead present in multi-node ordering services. Therefore, the reported performance results should be interpreted as indicative of prototype-level behaviour rather than a fully distributed production deployment. This limitation is further acknowledged in Chapter 9.

4.5.3 Network and Protocol Parameters.

The ordering service in Fabric determines when to cut a block based on time, transaction count, and block size thresholds. In this system, the default Fabric configuration was used, as shown in Table 4.1.

Table 4.1: Orderer block cutting parameters (default configuration).

Parameter	**Value**
BatchTimeout	2s
MaxMessageCount	10 transactions
AbsoluteMaxBytes	99 MB
PreferredMaxBytes	512 KB

In practice, this means that a block is created either every two seconds, once ten transactions have been received, or when the size of pending transactions reaches the specified thresholds. These parameters were adopted in this system to provide a well-balanced baseline between transaction latency and throughput. In addition, they are recommended by Fabric documentation and widely used in prior research, making them suitable for benchmarking and comparative evaluation. Using default values also ensures that the observed system performance is not biased by aggressive tuning, allowing a fair assessment of the proposed architecture. While these defaults were sufficient for our experimental deployment, they can be tuned in production systems depending on workload requirements (e.g., reducing latency by lowering the timeout, or increasing throughput by raising the transaction count).

4.5.4 Consensus Algorithm

The ordering service in this system is based on the *Raft consensus algorithm*. Raft was selected over older alternatives, such as *Solo*, which is limited to single-node development use, and *Kafka*, which introduces additional external dependencies and complexity. In particular, Raft offers several advantages that make it suitable for both current testing and future production deployments, as follows:

- Suitability for production environments and future scalability due to its capacity for crash fault tolerance with multiple orderers.
- Its native integration with Fabric, eliminating the need for third-party components.
- A simplified setup compared to older options like Kafka.

Overall, Raft provides a modern, robust and Fabric-aligned consensus mechanism that facilitates smoother network evolution than other options. This choice ensures that while the current setup remains lightweight for testing, it can be easily extended to a production-ready configuration by adding orderers.

4.5.5 State Database

Each peer node is configured with *CouchDB* as the state database. CouchDB offers a flexible and document-oriented data storage solution, enabling enhanced data visibility, auditability and complex querying capabilities compared to the default LevelDB, which is a key-value

store with limited query support. The use of CouchDB was particularly advantageous for this project because it enabled rich queries over asset attributes, supporting Attribute-Based Access Control (ABAC) enforcement and enabling audit-friendly data retrieval. In addition, CouchDB is better suited for applications that require complex access patterns or integration with external systems. This setup facilitates easier tracking of transactions and supports advanced data operations.

4.5.6 Membership and Identity (MSP)

Identity and access control across the network are managed through the *Membership Service Provider (MSP)* framework, which relies on certificates issued by the Hyperledger Fabric Certificate Authority (CA). In this system, the default Fabric CA was used to issue *X.509 digital certificates* for peers, orderers, and client applications. The CA also provides administrative certificates for organisation administrators and manages certificate revocation through a Certificate Revocation List (CRL). In the context of our system, the integration of the MSP and CA is critical because the entire framework depends on Attribute-Based Access Control (ABAC) to regulate data access and policy updates. Without this mechanism, ABAC-based enforcement would lack trustworthiness and consistency, undermining the ability of the system to securely differentiate roles, responsibilities, and levels of authority among stakeholders.

The workflow for identity management proceeds as follows:

- **Registration:** Users and components (peers, orderers) are first registered with the CA, which generates their enrollment credentials.
- **Enrollment:** The CA then issues a digital certificate and corresponding private key.
- **MSP Integration:** These certificates are stored in the organisation's local MSP directory, which contains root and intermediate CA certificates, node-specific keys, and organisational policies. This ensures that every entity in the network can be authenticated and authorised consistently.

Applications interact with the blockchain network using the Hyperledger Fabric Gateway SDK for Node.js. The SDK retrieves a user's identity from a local wallet or keystore (populated during enrollment) and authenticates it against the target organisation's MSP before allowing interactions with peers or orderers. This ensures that all client requests are cryptographically bound to valid organisational identities.

In addition, the MSP plays a central role in transaction validation, which is particularly important in our case of customised Endorsement Policies (EPs). Because EPs define which organisations must endorse a transaction, peers and orderers must be able to verify that endorsements come from the correct, trusted organisations. The MSP ensures that endorsement signatures correspond to valid organisational identities and that the rules of the EP are strictly enforced.

4.6 Chaincode Implementation

This section outlines the core chaincode modules developed for the system. Each function is responsible for managing data storage, access control, or policy updates in alignment with the system requirements.

1. `AssetExists`

   Purpose: Checks whether a specific asset with a given ID exists in the ledger.
   Inputs/Outputs:

   - Input: `id`.
   - Output: Boolean (`true` if asset exists, otherwise `false`)

   Method: Uses `ctx.stub.getState(id)` and returns "true" if the response is non-empty.

   Key Snippet:

```
async AssetExists(ctx, id) {

const assetJSON = await ctx.stub.getState(id);
return assetJSON && assetJSON.length > 0;
}
```

2. `CreateAsset`

   Purpose: Creates and stores a new asset in the ledger.
   Inputs/Outputs:

   - Inputs: `id`, `user_id`, `unique_name`, `data`, `owner`
   - Output: Transaction ID and stored asset as JSON

   Method: Constructs an object and stores it using `putState`.
   Key Snippet:

```
 async CreateAsset(ctx,id, user_id, unique_name, data, owner) };
await ctx.stub.putState(id, Buffer.from(JSON.stringify(asset)));
```

3. `ReadAsset`

   Purpose: Retrieves a stored asset from the ledger.
   Inputs/Outputs:

   - Input: `id`
   - Output: JSON string representing the asset

   Method: Uses `getState`, with error handling if asset does not exist.

```
async ReadAsset(ctx, id) {
const assetJSON = await ctx.stub.getState(id);
if (!assetJSON || assetJSON.length === 0) {
throw new Error(‘The asset ${id} does not exist‘);
}
return assetJSON.toString();
}
```

While *AssetExists*, *CreateAsset*, and *ReadAsset* follow standard patterns derived from the Hyperledger Fabric chaincode programming model and official documentation, their implementation here has been adapted and extended with custom fields and logic to support the proposed ABAC-enabled data sharing system.

4. `Register_User`

   Purpose: Registers a new user identity on the blockchain ledger.
   Inputs/Outputs:

   (a) Inputs: `email`, `username`, `role`, `organizations`

   (b) Output: Transaction ID and user data

   Method: Creates and stores user profile using email as the key.

Listing 4.1: JavaScript implementation of the `Register_User` function, which stores a new user's identity and associated details on the Hyperledger Fabric ledger

```
const asset = {
    email: email,
    username: username,
    role: role,
    organizations: organizations
};

await ctx.stub.putState(email, Buffer.from(JSON.stringify(asset)));
```

5. `Save_Hash`

   Purpose: Stores a data hash (e.g. IPFS CID) with metadata.
   Inputs/Outputs:

   - Inputs: `id`, `user_id`, `ipfsHash`, `type`, `role`, `organisations`
   - Output: Transaction ID and metadata

   Method: Writes structured metadata into the ledger.

Listing 4.2: JavaScript implementation of the `Save_Hash` function, which stores a data hash (e.g., an IPFS CID) along with associated metadata on the Hyperledger Fabric ledger using the `id` as the unique key.

```
const asset = {
    id: id,
    user_id: user_id,
    ipfsHash: ipfsHash,
    type: type,
    role: role,
    organizations: organizations
};

await ctx.stub.putState(id, Buffer.from(JSON.stringify(asset)));
```

6. `Get_ipfs_Hash`

   Purpose: Retrieves IPFS metadata for a given asset, with access restricted based on client identity attributes.

   Inputs/Outputs:

   - **Input:** `id` — the unique identifier of the asset.
   - **Output:** JSON string representing the asset's IPFS hash metadata.

   Method:

   - Uses HLF's `ClientIdentity` object to inspect the invoking client's attributes.
   - If the client's orgnisation, or `role` attribute, is not present or does not belong to allowed roles (`admin`, `employee`, or `investigator`), access is denied.
   - If the client is authorised, the function retrieves the asset from the ledger and returns its IPFS hash metadata.

   Access Control: Implements attribute-based access control (ABAC) using `ClientIdentity`. This access control logic is encapsulated within an additional wrapper contract `ABACWrapper`, which calls the underlying `GetipfsHsh` function only if the access conditions are met.

   Key Snippet from `ABACWrapper.js`:

Listing 4.3: JavaScript implementation of the `GetIPFSHashWithABAC` function, which enforces attribute-based access control (ABAC) before retrieving IPFS metadata.

```
async GetIPFSHashWithABAC(ctx, id) {
  const cid = new ClientIdentity(ctx);
  const role = cid.getAttributeValue('role');

  if (!role || !['admin', 'employee', 'investigator'].
  includes(role.toLowerCase()))
  {
    throw new Error(`Access denied: unauthorized role '${role}'`);
  }

  return await this.servicer.GetipfsHsh(ctx, id);
}
```

Key Snippet from the core `Get_ipfs_Hash` function:

Listing 4.4: Core `Get_ipfs_Hash` function that retrieves the asset from the ledger.

```
const assetJSON = await ctx.stub.getState(id);
if (!assetJSON || assetJSON.length === 0) {
  throw new Error(`The asset ${id} does not exist`);
}
return assetJSON.toString();
```

7. `Update_Policies`

Purpose: Updates access policies and metadata for an existing asset.
Inputs/Outputs:

- Inputs: `id`, `user_id`, `ipfsHash`, `type`, `role`, `organizations`
- Output: Updated JSON metadata

Method: Verifies asset existence, then overwrites the state.

Key Snippet:

Listing 4.5: JavaScript implementation of the `Update_Policies` function, which updates an existing asset's metadata and access policies on the Hyperledger Fabric ledger.

```
const exists = await this.AssetExists(ctx, id);
if (!exists) {
 throw new Error(`The asset ${id} does not exist`);
}

const updatedAsset = {
 user_id: user_id,
 ipfsHash: ipfsHash,
 type: type,
 role: role,
 organizations: organizations
};

let txID = ctx.stub.getTxID();
await ctx.stub.putState(id, Buffer.from(JSON.stringify(updatedAsset)));
return JSON.stringify(updatedAsset);
```

The chaincode modules presented in this section collectively implement the system's core logic for asset management, access control, and policy enforcement. While some functions, such as *AssetExists*, *CreateAsset*, and *ReadAsset*, follow standard Hyperledger Fabric patterns, others, such as *Save_Hash*, *Get_ipfs_Hash*, and *Update_Policies*, extend these patterns with custom logic tailored to the requirements of attribute-based access control and secure data sharing. A consolidated overview of all chaincode functions, their purposes, and their access types is provided in Table 4.2.

Table 4.2: Summary of Chaincode Logic and Access Types.

Function Name	Purpose	Access Type
`AssetExists`	Checks if an asset exists in the ledger	Read
`CreateAsset`	Creates and stores a new asset	Write
`Register_User`	Registers a new user on the ledger	Write
`Save_Hash`	Stores the IPFS hash and related metadata	Write
`ReadAsset`	Reads a stored asset	Read
`Get_ipfs_Hash`	Retrieves IPFS metadata	Read
`Update_Policies`	Updates access policy metadata	Read + Write

4.7 Integration and Interactions

This section presents how the various components of the proposed system are integrated to provide a secure and decentralised architecture. It explains the communication protocols, data exchange mechanisms and tools used to ensure seamless interaction between the system's layers, including the frontend interface, backend services, chaincode, and on-chain and off-chain storage.

Communication Protocols and APIs

To ensure reliable and secure communication among components, as illustrated in Table 4.3, the following protocols and APIs were used:

gRPC: Used internally within the HLF network for P2P communication and between the chaincode and peer nodes.

RESTful APIs: Exposed by the Node.js backend to allow frontend clients to interact with blockchain services (e.g. register users, submit hash, update policies).

Fabric SDK (Node.js): Handles identity management, wallet operations, chaincode invocation and query requests from the client to the blockchain network.

HTTP/S (Pinata API): Used to upload and pin data to the IPFS through authenticated requests to Pinata's REST API endpoints.

TLS/SSL: All internal and external communications are secured using TLS to ensure confidentiality and prevent tampering.

Table 4.3: System Communication Summary.

Communication	Protocol	Tech Used	Notes
Frontend ↔ Backend	HTTPS (REST)	JSON + JWT	Secure API calls
Backend ↔ Fabric	gRPC	Fabric SDK (Node.js)	Uses X.509 certs and MSP
Backend ↔ Pinata	HTTPS (REST)	Pinata API	File upload + hash return

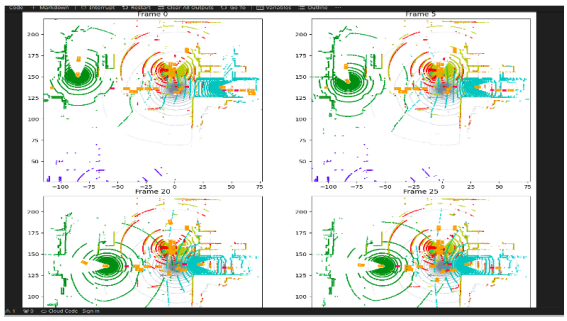
4.8 Data Set

The system was assessed using the V2X-Sim dataset supplied by Artificial Intelligence for Civil and Environmental Engineering (AI4CE) [123]. Figures 4.7a and 4.7b illustrate examples of LiDAR point cloud and camera-based perception outputs derived from the dataset. This dataset consists of simulated vehicle data designed to realistically represent V2X communication scenarios within urban settings, encompassing interactions among vehicles, infrastructure and pedestrians.

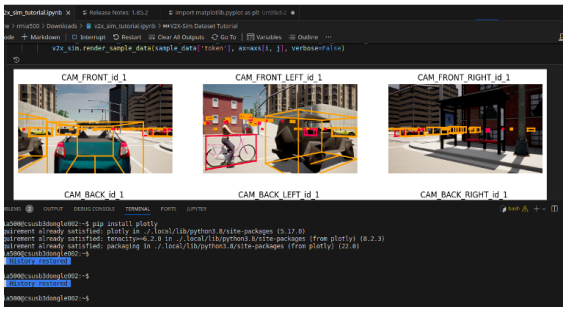
The dataset was selected primarily for its compatibility with the test environment used in our evaluation. Its structured format allowed seamless integration into our workload generation scripts, enabling consistent and repeatable experiments.

The V2X-Sim dataset is organised as a structured collection of files containing LiDAR recordings, metadata, and annotated perception outputs. In the context of this research, these files were packaged and uploaded as dataset archives during the experimental evaluation. This reflects a typical workflow in which autonomous vehicle datasets are shared after collection as stored files rather than real-time sensor streams. The proposed framework therefore, manages these datasets by storing the files in IPFS while recording their cryptographic hash values on the Hyperledger Fabric ledger to ensure integrity and traceability.

It was employed to evaluate the system's scalability and performance under various endorsement policy configurations.



(a) LiDAR point cloud for 3D mapping.



(b) Camera data with object detection.

Figure 4.7: Examples from the V2X-Sim dataset showing LiDAR and camera modalities for autonomous driving.

4.9 Test Environment Setup

This section describes the experimental setup used to evaluate the performance of the proposed system. Since the design involves storing data files in the InterPlanetary File System (IPFS) and saving only the corresponding hash values on the Hyperledger Fabric ledger, two complementary tools were required to obtain a complete performance profile. Hyperledger `Caliper` was employed to benchmark the blockchain network and measure metrics such as throughput, latency and success rate of the chaincode invoke transactions. However, `Caliper` does not capture the performance of data uploads into IPFS, which is

a critical component of the system. To address this gap, the `k6` load testing tool was used to simulate realistic user behaviour by concurrently uploading files of varying sizes while measuring throughput, response times, and scalability. Together, these tools provide both blockchain-level and application-level performance insights, ensuring that the evaluation covers the full workflow.

4.9.1 Caliper Configuration and Benchmarking Setup

This section outlines the steps and configurations required to set up HLF Caliper for performance benchmarking on an HLF network. It includes the necessary configuration files and JavaScript workload modules used in the benchmarking process.

Prerequisites

The following software components must be installed and properly configured to enable the deployment and benchmarking of the Hyperledger Fabric network:

- **Node.js** (v18+)
- **Docker** (v20.10+)
- **Hyperledger Fabric binaries** (v2.5.10)
- **Caliper CLI**

To conduct the benchmarking process, several configuration files and modules are required, each serving a distinct role in defining the network and workload scenarios:

A) **Network Configuration File:** `(test-network.yaml)`

The purpose of this file is to define the Fabric network topology, channel, chaincode and participating organisations. .This configuration ensures that benchmarking interactions are correctly mapped to the target blockchain environment. The implementation of this file is presented in 4.6.

B) **Benchmark Configuration File:** `(config.yaml)` This benchmark file is to define the test scenarios, including transaction types, transaction rates, and the number of workers. By configuring these parameters, different workload conditions can be simulated to evaluate system performance under varying levels of demand. The implementation of this file is shown in Listing 4.7.

C) **Workload Module:** `(theworkload.js)` This file, as shown in listing 4.8, is to simulate a transaction workload, such as querying or invoking any chaincode functions. It simulates realistic application behaviour by invoking specific chaincode functions.

Listing 4.6: Network Configuration

```
name: Caliper Benchmarks
version: "2.0.0"

caliper:
 blockchain: fabric

channels:
 - channelName: mychannel
 contracts:
 - id: mycc

organizations:
 - mspid: Org1MSP
 identities:
 certificates:
 - name: 'Org1'
 clientPrivateKey:
 path: '/home/ubuntu/fabric-samples/.../keystore/key_sk'
 clientSignedCert:
 path: '/home/ubuntu/fabric-samples/.../signcerts/cert.pem'
 connectionProfile:
 path: '/home/ubuntu/fabric-samples/.../connection-org1.yaml'
 discover: true
```

Listing 4.7: Benchmark Configuration

```

test:
 name: saveHash benchmark
 gateway: true
 workers:
 number: 1
 rounds:
 - label: saveHash
 txNumber: 100
 rateControl:
 type: fixed-rate
 opts:
 tps: 50
 workload:
 module: benchmarks/scenario/mycc/theworkload.js
 arguments:
 numberOfAssets: 100
```

For each test run, both the benchmark configuration file and workload module are typically updated, for example, to change the number of users or the chaincode function being invoked. To illustrate the benchmarking results generated by Caliper, Figure 4.8 shows a sample excerpt of the terminal report from one of the test runs. This summary includes throughput, latency, and success/failure metrics. These metrics are essential for assessing the effectiveness of the proposed system and identifying potential performance bottlenecks under varying load conditions.

Listing 4.8: Workload Configuration

```
'use strict';

const { WorkloadModuleBase } = require('@hyperledger/caliper-core');

class SaveHashWorkload extends WorkloadModuleBase {
 constructor() {
 super();
 this.txIndex = 0;
 }

 async submitTransaction() {
 this.txIndex++;
 const assetID = `asset_${this.txIndex}`;
 const request = {
 contractId: 'mycc',
 contractFunction: 'saveHash',
 invokerIdentity: 'Org1',
 contractArguments: [
 assetID,
 `user_${this.txIndex}`,
 `QmExampleHash${this.txIndex}`,
 'document',
 'auditor',
 'Org1MSP'
 ],
 readOnly: false
 };
 await this.sutAdapter.sendRequests(request);
 }
}

module.exports = SaveHashWorkload;
```

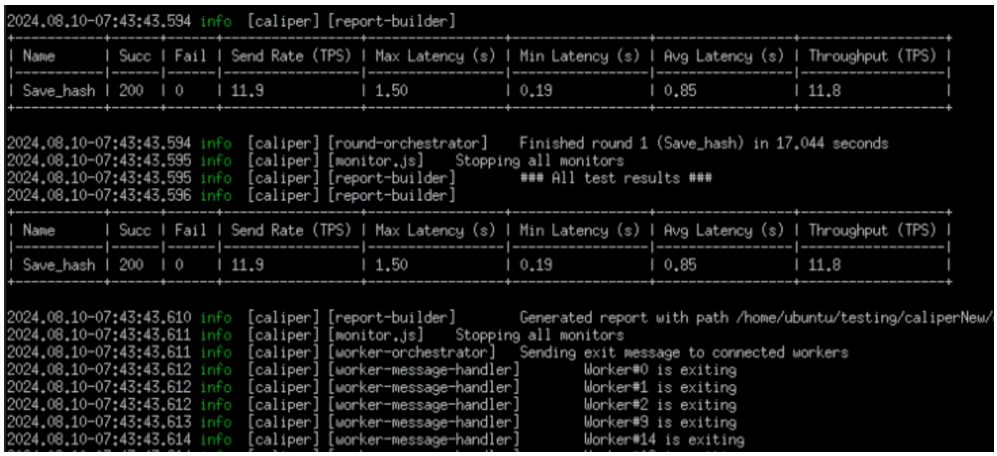


Figure 4.8: Example of Caliper terminal output report.

4.9.2 Performance Evaluation Using k6

As part of the system evaluation, the performance of the proposed system was validated using k6. Since gaining full insight into the time taken for IPFS file uploading is crucial to understanding overall system responsiveness, k6 was selected to support end-to-end assessment. The experiments simulated multiple users performing data uploads concurrently, allowing the measurement of throughput, response times, and system behaviour under high load conditions. The evaluation focused primarily on file upload performance under concurrent user load. The scripts used for these experiments are described in the following.

A) **File Upload Performance Under Concurrent Users**

The core evaluation focused on measuring file upload performance when multiple users interact with the system simultaneously. Using k6, virtual users were configured to upload files ranging from several megabytes up to multiple gigabytes. The experiments captured key performance metrics such as throughput, response times, and success rates. Two approaches were implemented as follows:

1. **Direct Upload Method:**
   Files were uploaded in a single request to evaluate system behaviour under straightforward load conditions, as illustrated in Listing 4.9.
2. **Chunked Upload Method:** Large files were split into smaller chunks (e.g., 1 MB) to validate the system's capability to handle extremely large datasets efficiently, as illustrated in Listing 4.10.

Listing 4.9: k6 script for uploading large files with multiple virtual users

```
import http from 'k6/http';
import { check, sleep } from 'k6';
import { FormData } from 'https://jslib.k6.io/formdata/0.0.2/index.js';

export let options = {
  scenarios: {
    upload_test: {
      executor: 'constant-arrival-rate',
      duration: '360s',
      preAllocatedVUs: 500,
      rate: 100,
    },
  },
};
var file = open('3gb.zip', 'b');
export default function () {
  const body = new FormData();
  body.append('picture', file, '3gb.zip');

  const res = http.post('http://54.160.64.184:8010/api/addsumofile',
  body.body(), {headers: { 'Content-Type': 'multipart/form-data;
  boundary=' + body.boundary },
  });
  check(res, { 'status 200': (r) => r.status === 200 });
  sleep(2);
}
```

Listing 4.10: k6 script for uploading large files in chunks

```
import http from 'k6/http';
import { check } from 'k6';
import { FormData } from 'https://jslib.k6.io/formdata/0.0.2/index.js';

var file = open('example-6gb.zip', 'b');
var chunkSize = 1024 * 1024; // 1 MB
export default function () {
  let offset = 0;
  const body = new FormData();

  while (offset < file.byteLength) {
    const chunk = file.slice(offset, offset + chunkSize);
    body.append('picture', chunk, 'chunk.zip');
    offset += chunkSize;
  }
  const res = http.post('http://54.160.64.184:8010/api/addsumofile',
  body.body(), {
    headers: { 'Content-Type': 'multipart/form-data; boundary=' +
    body.boundary },
  });
  check(res, { 'status is 200': (r) => r.status === 200 });
}
```

B) **User Registration** This test evaluated the ability of the system to handle a large

number of concurrent user registrations. Random user accounts were dynamically generated, and multiple registration requests were submitted in parallel to simulate real-world scalability conditions. This is illustrated in Listing 4.11

Listing 4.11: k6 script for dynamically generating and registering users

```
import http from 'k6/http';
import { check } from 'k6';

function getRandomUser() {
  const username = Math.random().toString(36).substring(2, 15);
  const email = `${username}@example.com`;
  const password = Math.random().toString(36).substring(2, 10);
  return { username, email, password };
}
export default function () {
  const user = getRandomUser();
  const payload = JSON.stringify({
    username: user.username,
    email: user.email,
    password: user.password,
    confirm_password: user.password,
    role: { organization: 'organization1', role: ['investigator'] },
    organizations: 'organization1',
  });
  const res = http.post('http://54.160.64.184:8010/api/newuserregister',
  payload, {
    headers: { 'Content-Type': 'application/json' },
  });
  check(res, { 'status 200': (r) => r.status === 200 });
}
```

C) **End-to-End Workflow Validation** This evaluation validated the complete workflow: user registration, account approval, and subsequent file upload as in Listing 4.12. It ensured that all system components integrated correctly under concurrent workloads.

Figure 4.9 shows a sample excerpt of the k6 terminal summary. This summary includes key metrics such as request rates, response time percentiles, and success ratios, which provide immediate insights into system performance under concurrent loads.

The combination of `Caliper` and `k6` ensured that the evaluation covered both the blockchain infrastructure and the application layer. `Caliper` provided fine-grained transaction metrics at the network level, while `k6` validated the system's robustness, scalability, and responsiveness under realistic user workloads. This dual approach offers a holistic understanding of system performance, forming the basis for the detailed analysis presented in the next chapters.

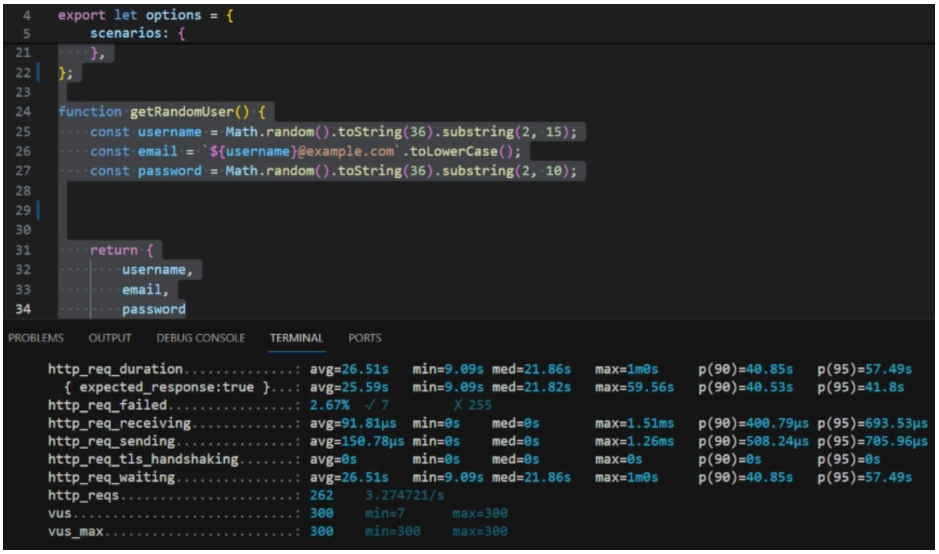


Figure 4.9: Example of k6 terminal output summarising request rates, response times, and success ratios.

Listing 4.12: k6 script for user registration, approval, and file upload

```
import http from 'k6/http';
import { check, sleep } from 'k6';
import { FormData } from 'https://jslib.k6.io/formdata/0.0.2/index.js';

export default function () {
  // User Registration
  const user = { username: 'testuser', email:
  'test@example.com', password: '123456' };
  let res = http.post('http://54.160.64.184:8010/api/newuserregister',
 JSON.stringify(user), {
    headers: { 'Content-Type': 'application/json' },
  });
  check(res, { 'registration status 200': (r) => r.status === 200 });

  // File Upload
  var file = open('10mb.zip', 'b');
  const body = new FormData();
  body.append('picture', file, '10mb.zip');

  res = http.post('http://54.160.64.184:8010/api/addsumofile', body.body(), {
    headers: { 'Content-Type': 'multipart/form-data; boundary=' + body.boundary },
  });
  check(res, { 'upload status 200': (r) => r.status === 200 });
  sleep(2);
}
```

4.10 Assumptions and Considerations for Real-World Deployment and Resource Scaling

This section outlines the key assumptions made during the design and implementation of the prototype, as well as considerations for scaling and deploying the system in real-world environments. The following assumptions highlight the trust, security, and operational requirements necessary for the framework:

1. **Users upload encrypted data to the IPFS**
   Although IPFS provides content-addressed and immutable storage, it does not offer built-in encryption. Therefore, the system assumes that users encrypt their files locally before uploading them to IPFS or via the backend. Decryption is performed on the client side after authorised access is granted. This design choice ensures that sensitive data remains confidential even when stored in a distributed and potentially untrusted storage environment. Integrating encryption within the backend or client layer is technically feasible; however, secure key management, including key distribution, revocation, and recovery, introduces significant complexity and security challenges. As a result, encryption is treated as an external assumption in this prototype and is identified as an important area for future work.

2. **Administrator is trusted and authorised**
   The system assumes that the admin user is a trusted entity with the authority to perform sensitive operations such as user registration. Their credentials and permissions are verified before invoking privileged chaincode functions (e.g. `Register_User`).

3. **User role/ attributes are accurately assigned and managed**
   It is assumed that user roles (`Admin`, `Employee`, `Investigator`) are correctly assigned at registration via ABAC. These roles are securely embedded in the user's X.509 certificate by the CA and used consistently within the chaincode logic to enforce access policies.

4. **Endorsement Policy agreement among all organisations**
   It is assumed that all participating organisations have mutually agreed on the defined endorsement policy (EP) before joining the network. This agreement ensures consistency in transaction validation rules and avoids conflicts during chaincode invocation and endorsement processes.

The following considerations reflect potential extensions and resource scaling strategies beyond the prototype stage.

1. **Infrastructure can be scaled based on demand**
   During development and testing, AWS EC2 instances were used, with upgrades to CPU, RAM and EBS volume size applied as needed. It is assumed that in a real-world deployment, the infrastructure can be scaled vertically or horizontally to support increased user activity, transaction throughput and data volume.

2. **Use of Pinata for decentralised Off-Chain Storage**
   Pinata, a managed IPFS pinning service, was integrated to simplify off-chain storage

and ensure reliable file availability. This assumes that for small- to medium-scale deployments, relying on a hosted service reduces operational overhead while still leveraging IPFS's decentralised content addressing.

3. **Possibility of self-hosting the IPFS in large-scale environments**
   While Pinata was suitable for the prototype, it is assumed that in large-scale production environments with adequate resources, organisations could deploy their own IPFS nodes or clusters. This would reduce reliance on third parties, increase control over data lifecycle and retention, and potentially reduce long-term costs.

4. **Prototype scope versus Real-World channel and peer configurations**
   The current implementation assumes a simple network with a single channel and one peer per organisation. However, in real-world deployments:

   - An organisation may participate in multiple channels with different partners. For example, a supplier may cater to several manufacturers.
   - Organisations may operate multiple peers for load balancing, high availability, or geographic distribution.
   - Channels may use multiple ordering service nodes for fault tolerance and scalability.
   - The network may include more than five organisations per channel, depending on business needs.

This research contribution provides staged scaling that presents valuable insights into system behaviour, endorsement policy flexibility and performance under increasing user and transaction load. These evaluations inform expectations for real-world scalability.

4.11 Key Challenges

- **Resource Constraints and Scaling.** The initial deployment used AWS EC2 `t2.medium` instances (2 vCPUs, 4 GB RAM) for peers and the backend server. However, during high-load testing and after expanding to five organisations in Phase 2, resource limitations became apparent, especially in handling concurrent transactions and chaincode execution. To address this, the instances were upgraded to `t2.large` (2 vCPUs, 8 GB RAM) and adding vCPUs/RAM as needed to handle increasing workloads and heavy load testing, which improved performance and stability but at a high cost as a trade-off.

- **Limited Control over IPFS Storage.** Initially, it was intended that the system architecture would use a self-hosted IPFS node deployed on AWS to manage off-chain data. However, this approach introduced challenges due to limited infrastructure resources. Hosting and maintaining a reliable, always-online IPFS node was challenging, especially during scaling. To overcome these limitations and ensure consistent performance during high-load tests, the system was migrated to a managed IPFS gateway (Pinata). Pinata simplified the process by automating file pinning and providing a stable API for uploading and retrieving hashes. This greatly enhanced system scalability and reliability, especially under heavy test scenarios. However, once the test scenarios required large-scale uploads and pinned data persistence, the Pinata free tier was

quickly exhausted, leading to additional financial overhead for paid API usage. Moreover, reliance on a third-party service slightly reduced the level of decentralisation originally intended, which might not be needed in real-world scenarios.

- **Selecting an Autonomous Vehicle Dataset.** One of the major challenges in testing the proposed platform is identifying and selecting a suitable AV dataset. Autonomous driving systems generate massive, complex, and heterogeneous data that include sensor inputs such as LiDAR, radar, camera images, GPS, and vehicle dynamics. However, publicly available datasets often differ in format, scope, and quality, making it difficult to find one that aligns with my platform's requirements.

4.12 Conclusion

The implementation was successful in demonstrating a multi-stakeholder system designed, as illustrated in the previous chapter, to address the core goal of secure and efficient AV data sharing among multiple stakeholders, including manufacturers, suppliers and regulatory bodies. The system met critical requirements, such as data security via fine-grained access control, and scalable decentralised storage. It also ensured usability, allowing stakeholders with different technical expertise to interact with the platform seamlessly, and maintained flexibility to accommodate future extensions or additional participants.

By integrating an HLF blockchain with a Node.js backend and a secure, friendly frontend interface, the system enabled trusted on-chain storage of data hashes alongside off-chain storage of large AV data files via the IPFS. This combination of technologies provided a balance between decentralisation, performance, and security, while also simplifying development and integration through the use of a consistent technology stack.

Despite the challenges encountered, such as infrastructure resource limitations and the integration of managed pinning services, the prototype demonstrated the feasibility of implementing a permissioned blockchain solution tailored to the complex requirements of multi-stakeholder AV data sharing. Additionally, the implementation provided insights into practical considerations, such as system scalability, transaction throughput, and access policy enforcement, which can guide future development and optimisation of similar systems. Overall, the implementation not only validates the proposed functional architecture but also offers a foundation for further experimentation and evaluation in large-scale or production environments.

Chapter 5

Baseline Performance Evaluation

While the previous chapters focused on designing and implementing the system, this chapter presents the evaluation of the developed prototype. The objective of this experimental phase was to assess the baseline performance, functionality, and policy behaviour of the proposed system in a controlled and realistic multi-organisation setup. By using a network of three and then five organisations, the experiment aimed to simulate a typical permissioned blockchain environment in which data security, access control, and consensus mechanisms play critical roles. This evaluation directly addresses the research questions regarding the system's scalability, reliability, and effectiveness in enforcing access control policies **RQ2a and b**, providing empirical evidence on whether the proposed architecture meets the identified requirements.

The evaluation in this phase primarily investigated the system's response under varying transaction loads, data loads, user loads, and default endorsement policies, aiming to identify its effectiveness and limitations. This included analysing throughput and latency under different configurations. In addition, the baseline evaluation considered the usability of the system, which addresses the Research Question **RQ3**, examining how easily stakeholders with different needs could interact with the platform, submit transactions, and retrieve data, thereby assessing whether the system meets practical operational needs.

The results established initial benchmarks that enable comparison with alternative endorsement policies, facilitating a systematic assessment of both performance and scalability in later phases of this research. In addition, they served as the foundation for further enhancements and more complex deployments in subsequent experimental stages.

This chapter presents the following: the section 5.1 outlines the main objectives in 5.1.1, guiding the evaluation. Blockchain Layer Evaluation (5.1.2) focuses on evaluating the performance of the blockchain layer, while API Layer Evaluation (5.1.3) addresses the API layer. Scaling to Five Organisation (5.3) extends the evaluation to examine scalability by including five organisations. Moreover, this chapter covers the usability objective and assessment in 5.5. Findings are discussed in 5.4 to offer benchmarking insights. Finally, the chapter conclusion is presented in 5.6

5.1 Evaluation

This evaluation was essential to validate the design and implementation of the system. In addition, it provided a performance baseline to be used for comparison in subsequent evaluations in the following chapters, particularly when applying alternative endorsement policies. By identifying performance bottlenecks and scalability limits, this evaluation supported the research objective of examining the efficiency of permissioned blockchain environments for secure access control and reliable data management in multi-organisational, data-sharing settings, such as those found in AV ecosystems.

5.1.1 Evaluation Objectives

The primary aim of this evaluation is to evaluate the performance and behaviour of the system implemented under a set of defined scenarios. Specifically, the evaluation focused on the following aspects:

1. **System Performance:** Understanding how the system performs with different operations. This includes measuring throughput, latency, and error rates to identify potential bottlenecks and ensure that performance targets are met for real-time AV data sharing.
2. **Scalability:** Examining how the system performed when increasing the following:
   (a) Number of concurrent users.
   (b) Transaction rate.
   (c) Volume of data uploaded to IPFS.
3. **System Responses:** Evaluating the system API responsiveness, scalability, and robustness. This assessment helps determine whether the system can consistently provide timely and correct responses to user requests.
4. **Usability:** The purpose of usability evaluation is to understand how effectively, efficiently, and satisfactorily users can interact with the system. This includes user experience considerations such as clarity of interface, intuitiveness of workflow, and error handling, which are critical for adoption and practical deployment in AV data-sharing environments.

By addressing these dimensions, the evaluation establishes a comprehensive understanding of the behaviour and performance of the system. It also lays the groundwork for deeper analysis in subsequent chapters, where results will be interpreted to identify trade-offs, strengths, and potential areas for future optimisation in both the system architecture and configuration.

Error Bars: Purpose and Use in Evaluation

Error bars are graphical representations that are used to visualise the variability or uncertainty in a set of data points. In performance evaluation, particularly in comparison or benchmarking, error bars help assess the consistency and reliability of the reported results. They provide important insights into the stability of performance metrics across multiple runs or experimental repetitions.

There are two commonly used types of error bars: **Standard Deviation (SD)** and **Confidence Interval (CI)**. In this work, we report error bars using SD, which is used to represent variability between repeated experiments. General visualisations comparing the stability of different endorsement policies represent the spread of individual observations around the mean. It is a measure of variability and indicates how much the measured values differ from each other. SD is calculated as:

$$\text{SD} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

where x_i are individual measurements, $\bar{x}$ is the sample mean, and n is the number of observations. For example, for Approach 2 at TPS 200 with latency values $[0.83, 0.88, 0.86, 0.90, 0.82]$, the mean is $\bar{x} = 0.858$. The squared deviations sum to 0.00448, giving a variance of 0.00112 and $\text{SD} = \sqrt{0.00112} \approx 0.033$.

Thus, error bars enhance the interpretability of results and ensure transparency when evaluating the performance and consistency of different system configurations. **Variability is represented using standard deviation (SD), shown either as shaded regions or error bars depending on the figure.**

5.1.2 Blockchain Layer Evaluation

This section explains how the HLF layer was evaluated in this first evaluation phase. This includes tools and metrics, scenarios, and results and observations 5.1.2. This evaluation specifically addresses point 1 evaluation objectives, which is assessing system performance and point 2, which focusses on examining scalability, providing quantitative evidence of how the blockchain layer behaves under different conditions.

Method and Metrics

`Caliper`, as a widely adopted benchmarking tool for blockchain systems, was used in this experiment to generate workloads and capture performance metrics. Its use enabled rigorous and repeatable evaluation of the system under realistic multi-organisation scenarios, providing insights into both the efficiency and reliability of transaction processing. By simulating varying numbers of users and transaction rates, `Caliper` facilitated the assessment of how well the system meets the objectives of scalability, responsiveness, and stability.

In the experiments, the following metrics were used to evaluate the system:

- **Throughput:** Measures the number of successfully committed transactions per second (TPS). TPS is a key indicator of the system's capacity under load.
- **Latency:** Refers to the time taken from when a transaction is submitted until it is confirmed on the blockchain. Both average and maximum latency were measured to assess the response. While the system was not designed for real-time operation, latency remained an important metric to ensure that the operations occur within acceptable timeframes for practical use.
- **Success Rate:** The percentage of transactions successfully committed versus the total sent. This metric helps identify potential failures or bottlenecks in processing. A high success rate indicates that the system can consistently process transactions as expected, offering evidence of its suitability and efficiency.

These metrics together provide a comprehensive picture of the system's operational behaviour under different configurations.

Test Scenarios

When designing the tests, we ensured that all chaincode functions deployed, covering read-only, write-only, and combined read/write operations, were thoroughly evaluated. Each function was tested under varying workloads to observe system behaviour, performance consistency and potential bottlenecks under different operational scenarios. The selection of workload parameters, including user load and transaction rates, follows commonly adopted practices in blockchain performance evaluation. Prior work on Hyperledger Fabric benchmarking applies incremental workload scaling to assess throughput, latency, and resource utilisation under controlled conditions, with tools such as Hyperledger Caliper using progressively increasing transaction rates (e.g., 50–500 TPS) and varying client loads to identify system limits and performance bottlenecks [124, 125, 126]. Therefore, the following scenarios designed primarily for benchmarking and stress-testing purposes rather than to directly represent real-world AV data-sharing patterns.

Based on this methodology, test scenarios, in the baseline stage, were defined as follows:

- **User Load:** 10, 100 and 200 concurrent users. These levels were selected to represent increasing levels of system load across participating organisations.
- **Transaction Rates (TPS):** 50, 100 and 200. These values were chosen to assess system throughput under increasing demand and to evaluate how well the platform handled rising transaction volumes.
- **Chaincode Functions:**
  - `Register_User`

- `Save_hash`
- `Git_IPFS_hash`
- `Update_Policies`

These functions represent the core operations of the system. Testing each one ensured comprehensive coverage of typical data interaction scenarios relevant to the use case. Each scenario was executed in multiple rounds (typically 5) to assess the consistency of results and identify potential performance thresholds.

This baseline enables meaningful comparison with subsequent evaluations presented in later chapters, particularly when assessing the impact of alternative configurations such as endorsement policies. Therefore, the observed results reflect system performance within the tested ranges and should be interpreted as indicative of system behaviour under controlled experimental conditions, rather than as a direct representation of real-world AV data access frequencies or deployment scenarios. In addition, establishing this baseline ensures that observed improvements or degradations in performance can be accurately attributed to specific changes in the system rather than variability in testing conditions.

Results and Observations

The main observation was that all functions executed reliably across all tested scenarios; however, their performance characteristics varied with changes in user load and transaction rates. These patterns are illustrated in Figure 5.1, which compares the throughput of all functions across user levels. Throughput for the `Register_User` chaincode function remained consistent, maintaining stability around 19–20 TPS at 100 VUs and dropping to approximately 11 TPS at 200 VUs. At lower loads, i.e. 10 VUs, throughput peaked near 30 TPS, 60% of the target TPS rate configured in this scenario, benefiting from reduced concurrency and system overheads.

While all the other chaincodes followed similar trends, `Get_IPFS_hash` maintained the highest and most stable throughput, whereas `Update_Policies` exhibited the greatest decline under heavier loads. This is another important initial observation regarding the impact of the complexity of chaincode operation on the performance.

The latency metric was also affected by the users load, as shown in Figure 5.2, which compares the latency average for all chaincodes. For example, `Register_User`, ranged from around 0.3 seconds at 10 VUs to 1.19 seconds at 200 VUs. This increase aligns with expectations concerning concurrency. The figure also displays that the `Update Policies` function shows the highest latency, peaking at nearly 2 seconds. This function is the most complex, containing read and write operations.

To better isolate the effect of varying target TPS on system performance, we fixed the number of users in this set of experiments. This approach allowed us to analyse how the system behaved under different throughput demands without the added variability introduced by changes in the number of concurrent users. Controlling this variable was essential to

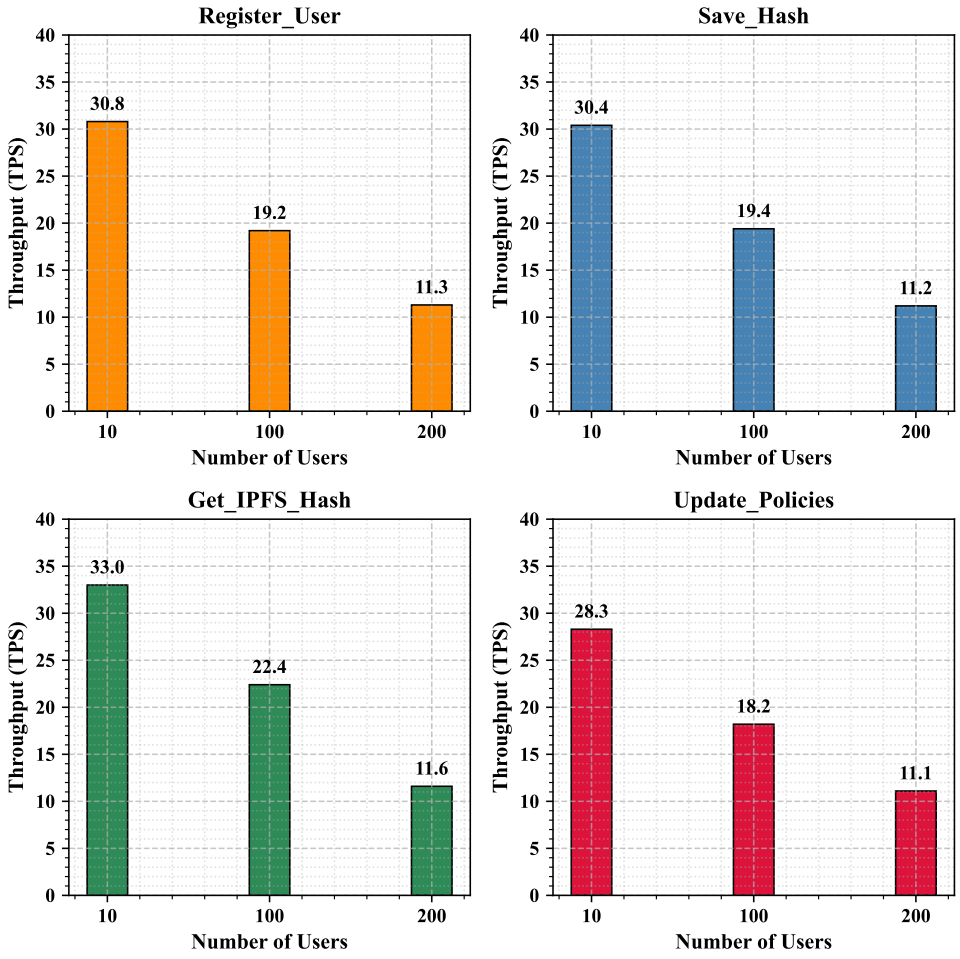
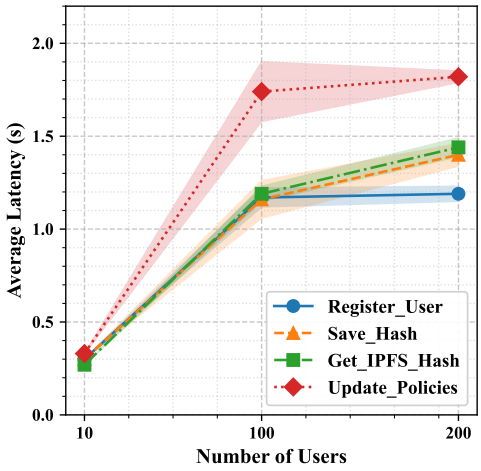


Figure 5.1: **Throughput Across Varying Virtual User Levels.** This figure compares the throughput performance (in transactions per second, TPS) of four chaincode functions: `Register_User`, `Save_Hash`, `Get_IPFS_Hash`, and `Update_Policies`, under different virtual user (VU) loads (10, 100 and 200 users).



Average Latency (s)

	10 U	100 U	200 U
Register_User	0.30	1.17	1.19
Save_Hash	0.30	1.16	1.40
Get_IPFS_Hash	0.27	1.19	1.44
Update_Policies	0.33	1.74	1.82

Figure 5.2: **Average Latency under Different User Loads.** This figure compares the average latency of four chaincode functions as the number of virtual users increases from 10 to 200, with shaded regions representing the standard deviation (SD) to illustrate variability and consistency across repeated experiments.

understand the true scalability limits of each function and to ensure that any increase in latency or performance degradation was attributable to the TPS target rather than compounded load from user concurrency.

Figure 5.3 shows that under increasing target TPS values (50, 100, 200), with the number of users held constant (100 VUs), the overall trend is for all functions to maintain relatively stable throughput across different TPS rates; however, they do not scale well.

The latency trends followed an expected pattern, increasing gradually with more virtual users and higher TPS targets, but remaining within a generally acceptable range of 0.3 to 2.5 seconds under heavy load (100–200 VUs). Another noteworthy observation is that, among the four functions, `Get_IPFS_Hash` consistently outperformed the others in terms of throughput under stress. As seen in Figures 5.1 and 5.2, it maintained the highest and most stable throughput at high TPS and user levels. This is likely due to its read-only nature, which avoids costly state updates. In contrast, `Update_Policies` showed the most performance degradation under load. Throughput dropped significantly under 200 VUs.

In general, the results reveal a clear throughput ceiling across all chaincodes. As shown in Figures 5.3, throughput levels off regardless of the targeted TPS, particularly beyond 100 TPS. This suggests that simply increasing the transaction submission rate does not improve system performance, motivating us to optimise and further evaluate different scenarios.

5.1.3 API Layer Evaluation

This section aims to evaluate the performance of the API layer and its integration with the underlying blockchain components. The objective is to analyse how the system behaved under different scenarios.

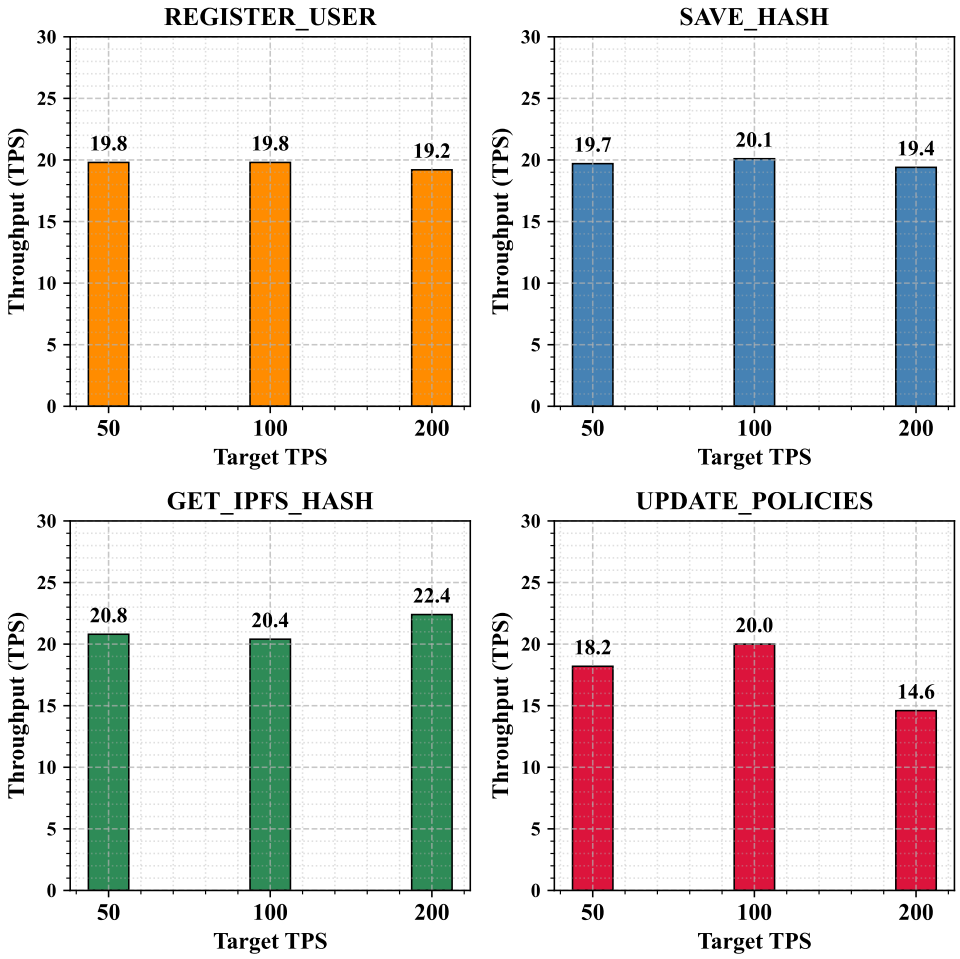


Figure 5.3: **Throughput at Different TPS Levels.** This figure presents the throughput performance for four chaincodes: *Register_User*, *Save_Hash*, *Get_IPFS_Hash*, and *Update_Policies* under 50, 100 and 200 TPS.

Method and Metrics

To evaluate the responsiveness, scalability and robustness of the API layer, two performance testing tools were used: **Apache JMeter** [127] and **K6** [128].

These tools allowed us to simulate concurrent users interacting with the API endpoints responsible for file upload and download, and to measure key performance indicators under various load conditions. A set of performance metrics was measured to capture both network level and application level behaviour. These metrics included standard indicators such as response time, throughput, error rate, and latency, as well as more detailed measurements provided by K6, such as
texttthttp_req_duration, `http_req_blocked`, `http_req_connecting`. Table 5.1 summarises these metrics, their descriptions, and the tools used for collection.

Table 5.1: Metrics Measured in API Evaluation.

Metric	**Description**	**Tool**
Response Time	Time from request initiation to completion (s)	Both
Throughput	Number of requests handled per second	Both
Error Rate	Percentage of failed requests	Both
Latency	Delay experienced during data transfer	JMeter
`http_req_duration`	Total request time including network and processing delays	K6
`http_req_blocked` / `http_req_connecting`	Network and connection overhead	K6

Test Scenarios

To test both the API and the responsiveness of the frontend, the following scenarios were designed and evaluated:

1. Registration and logging in operations with and without prior approval to measure the performance of the system when handling the access function. The test used `K6` to simulate 1, 10, 100 and 1000 VUs, measuring how the system responded under increasing loads.

2. Full user journey across the platform to evaluate the performance and responsiveness covering eight key modules: *Sign Up, Login, Dashboard, Data Upload, Request User, Owner List, Profile,* and *Change Password*. The test used `Apache JMeter` to simulate 1, 10, 100 and 1000 VUs, measuring how the system responded under increasing loads.

3. Evaluation of the uploading data performance under varying user loads and data sizes. The test used `K6` to simulate 1, 10, 100 and 1000 VUs, measuring how the system responded under increasing loads and data volume.

Results and Observations

To evaluate the system's performance for both authorised and unauthorised access, including account registration and login, we measured the average response time across different user loads. These entry-point operations were critical for verifying the system's stability and responsiveness under concurrent access. As shown in Figure 5.4, both the `Register` and `Login` modules demonstrated functional stability, with no errors reported across 10, 100, and 1000 VUs. However, the `Login` operation shows a sharper increase in response time compared to `Register`, likely due to additional overhead such as token verification or session handling during the authentication process. In the figure, the error bars represent the variability in response time across experimental runs.

Furthermore, key application modules, such as `Dashboard`, `Profile`, `Change Password`, and `OwnerList`, were also evaluated for their runtime performance under the same user load conditions. Figure 5.5 presents the average response times for several of these modules. Notably, `Dashboard` and `Login` operations consistently exhibit higher response times across all test cases, suggesting their increased computational or data handling requirements. In contrast, modules like `Profile` and `Sign Up` demonstrate better scalability and lower latency as the user load increases. The `Change Password` module remains one of the slowest throughout, likely due to additional validation tasks. Despite these variations, the system as a whole maintained consistent throughput and exhibited no operation-level failures at any user level.

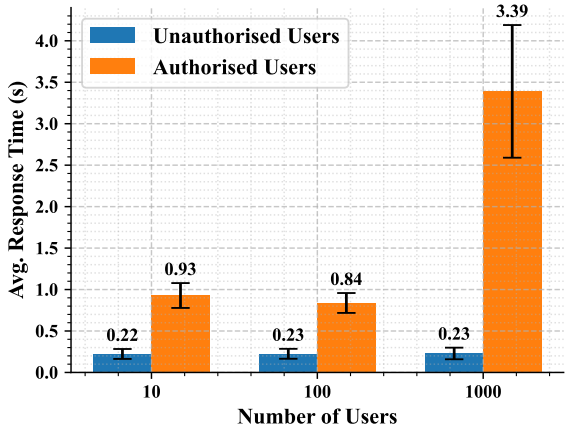


Figure 5.4: Average response time for unauthorised and authorised `Register` and `Login` access under different user loads (10, 100, 1000 VUs). The error bars represent Standard Deviation (SD).

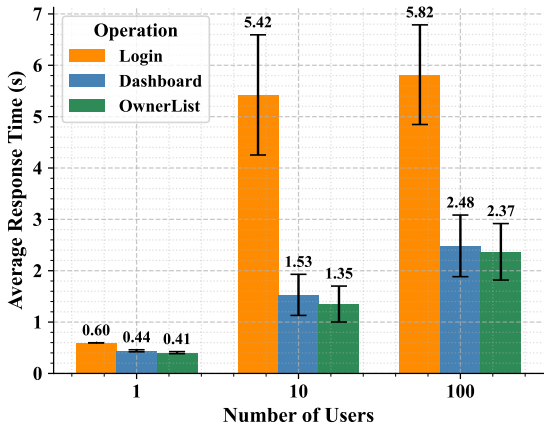


Figure 5.5: Average response time (in seconds) for the **Login**, **Dashboard**, and **OwnerList** operations across 10, 100, and 1000 virtual users. The error bars represent Standard Deviation (SD).

5.2 Implications for Next Phases

This phase of evaluation provided a clear baseline for system behaviour under varying user loads and transaction rates. While the system demonstrated stable and reliable

performance, its throughput did not increase proportionally with higher VU counts or higher TPS. Consequently, the next phase of testing was planned with three main objectives: first, to upgrade system resources to better handle moderate loads and improve overall throughput; second, to scale the network to five organisations to evaluate the impact of increased decentralisation and multi-stakeholder governance on performance; and third, to test the system's ability to maintain consistent transaction processing and policy enforcement under these enhanced conditions.

This next phase serves as a critical step in validating the system's scalability, resilience, and suitability for real-world deployment, while also providing insights into potential optimisations and design improvements for both the blockchain and off-chain components. These steps directly relate to the research questions on system scalability, performance under load, and multi-organisation policy enforcement (RQ1 and RQ2).

5.3 Scaling to Five Organisation

The HLF network was extended to include five distinct organisations. Each organisation operates its peer and participates in consensus and endorsement processes, simulating a real-world multi-stakeholder consortium. This configuration introduces additional complexity in governance, transaction propagation, and chaincode execution, providing a more realistic environment for assessing the system's operational limits. The aim of this enhanced setup is to evaluate how the system performs under increased decentralisation, both in terms of throughput and latency, and to identify any challenges in coordinating multiple organisations within the permissioned blockchain. This builds directly on the baseline evaluation, where tests were limited to three organisations, 200 users, and 200 TPS, and provides a stronger foundation for analysing the scalability of the system.

5.3.1 Evaluation Objective

The objective of this evaluation stage is to evaluate the throughput performance and scalability of critical chaincode functions in the newly scaled five-organisation HLF network. Specifically, the evaluation seeks to determine whether the system can maintain efficient transaction processing and handle increased organisational diversity without significant performance degradation.

This phase establishes the basis for comparing different endorsement policies and system configurations in subsequent experimental chapters.

5.3.2 Blockchain Layer Evaluation

The blockchain layer was examined to determine how effectively it handles increased organisational diversity and transaction load in a larger consortium setup. By introducing five independent organisations, the endorsement policy becomes more demanding, requiring more

peers to validate each transaction. This not only introduces endorsement overhead but also affects the sending and ordering of transactions across the network.

Test Scenarios

Each function: `Register_User`, `Save_hash`, `Git_IPFS_hash`, and `Update_Policies` was tested under varying scenarios to measure how well the system handles different stress levels in a multi-organisation setting. This testing aims to identify potential performance bottlenecks introduced by scaling and ensure that throughput remains within acceptable bounds as the network grows in complexity. Building upon the baseline evaluation methodology defined in Section 5.1.2, the following increases both user load and TPS to further assess system scalability under higher stress conditions:

1. **Number of Users** Multiple user loads were simulated to assess the system's throughput and latency when subjected to increasing demand. While the baseline tests were bounded above 200 users, this phase extended user loads up to 500, allowing for a more comprehensive scalability analysis.
2. **Varying TPS** The system was stressed with varying TPS rates. This phase expanded the TPS range from the previous 200 limit to 500 TPS to observe how well it could maintain performance at different input rates.
3. **Upload Data** An end-to-end functional stress test has been conducted to assess both the backend and blockchain layers to get insights into how loads affects user experience and system scalability.

Each scenario was executed in multiple rounds to account for result consistency and to identify potential performance thresholds.

Results and Discussion

The evaluation of the blockchain layer under varying workloads provides insight into the performance and scalability of the implemented chaincode modules. The results indicate a consistent trend of decreasing throughput as load increases across all chaincode operations.

All modules demonstrated functional reliability under low to moderate loads as illustrated in Figure 5.6. The figure shows that under a uniform light load of 50 TPS and 100 users all modules revealed similar throughput levels (15.7–16.2 TPS). However, as the number of users or the input transaction rate increases, a performance degradation is observed. In particular, throughput does not scale linearly with increasing load, indicating bottlenecks in processing. The system performance degraded significantly as user concurrency increased, even though the transaction rate stayed the same. For example, as in Figure 5.7. The throughput performance of the `Save_Hash` function degrades as the number of concurrent users increases from 100 to 500 while the targeted TPS rate maintains at 50.

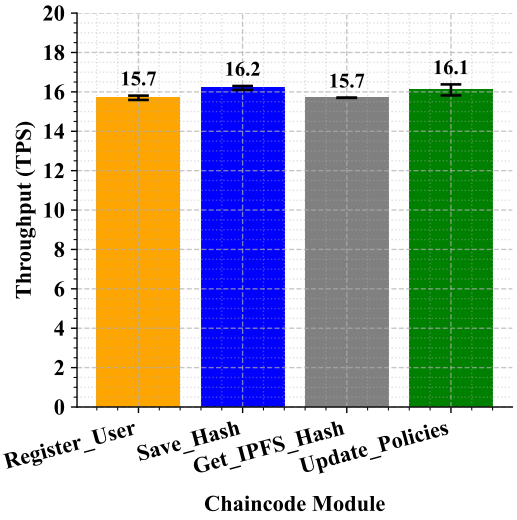


Figure 5.6: **Throughput Comparison of Chaincode Modules at 50 TPS with 100 Users.** This bar chart compares the throughput (transactions per second) of four chaincode modules under identical workload conditions. Error bars represent the standard deviation (SD), indicating the variability across repeated experimental runs.

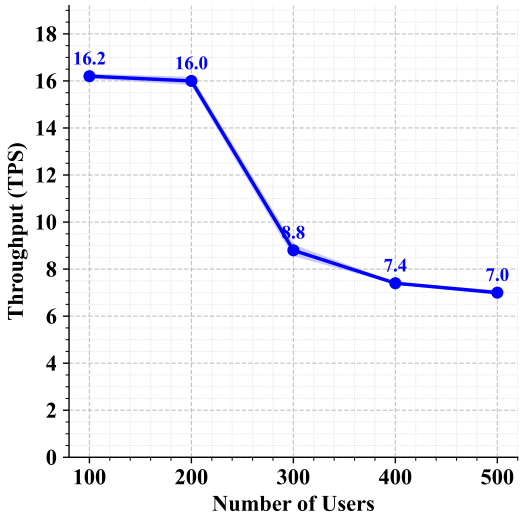


Figure 5.7: **Save_Hash Module Throughput Performance at 50 TPS.** This figure illustrates how the throughput of the `Save_Hash` function degrades as the number of concurrent users increases from 100 to 500. The shaded region represents the standard deviation (SD), highlighting the variability and stability across repeated experiments.

In contrast, the performance of the system is less sensitive to higher TPS rates than to increases in user concurrency. Figure 5.8 demonstrates how the throughput of the `Update_Policies` chaincode module behaves under increasing the TPS rates, with the number of users fixed at 200. As the TPS rate increases from 100 to 500, the throughput remains relatively stable, ranging between 11.8 and 12.1 TPS.

To further evaluate the responsiveness and scalability of the API layer, as mentioned in section 5.3.2, series of load tests were conducted using the K6 framework. The aim was to simulate concurrent user requests invoking the `Save_Hash` function, which upload the data on IPFS and internally saves hashes to the blockchain network. These tests reveal the backend's capacity to handle increasing user load and transaction rates, and also indirectly reflect the stress placed on the underlying Hyperledger Fabric components.

Figure 5.9 illustrates the impact of increasing the number of concurrent users on the `Save_Hash` function when handling a 3GB file. As the number of users rises from 100 to 500, throughput declines from 16.2 TPS to 3.9 TPS, clearly demonstrating the system's sensitivity to user load and the limitations of parallel request handling for large payloads.

These figures provide a comprehensive view helping to identify performance constraints.

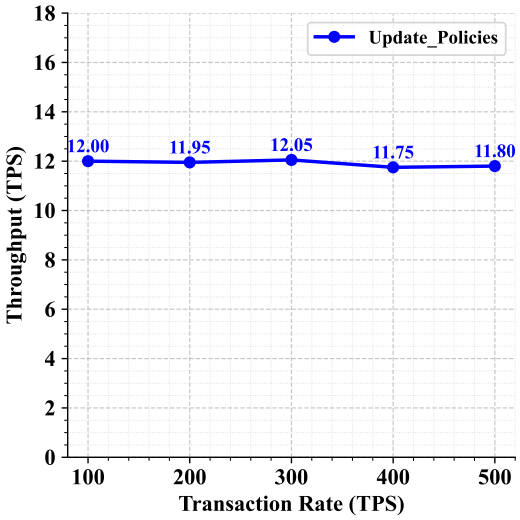


Figure 5.8: **The Update_Policies Throughput Performance at Varying TPS Rates** This line graph illustrates the average throughput of the `Update_Policies` chaincode module as the send rate increases from 100 to 500 TPS while keeping the number of users constant at 200.

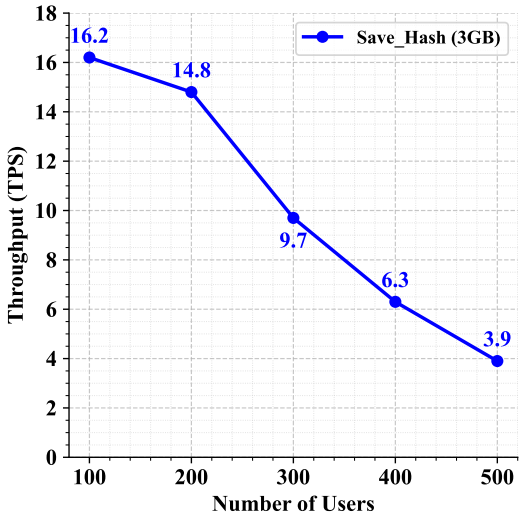


Figure 5.9: **Throughput vs Number of Users for Save_Hash with 3GB File Size** The figure shows how throughput decreases as the number of users increases. The highest throughput is achieved at 100 users (16.2 TPS), gradually dropping to 3.9 TPS at 500 users.

5.4 Discussion and Benchmarking

In this chapter, the baseline evaluation has been provided in two network configurations. The throughput comparison between the three organisations and the five organisations' network configurations reveals a clear performance impact caused by increasing the number of organisations. As showen in Figure5.10, across all chaincode modules, the throughput is consistently higher in the three-organisation setup, with values ranging from approximately 18.2 to 20.8 TPS, compared to the five-organisation network, where throughput drops to between 15.7 and 16.15 TPS. This decrease is primarily due to the additional communication and endorsement overhead introduced by the larger network size, which increases transaction processing time and reduces overall throughput. Among the modules, `Get_IPFS_Hash` shows the smallest relative throughput decline, suggesting it scales better with increased organisations, likely due to its read-only nature. While `Update_Policies` experiences a more noticeable reduction due to its complexity.

Notably, the system demonstrated a clear performance ceiling, where throughput did not scale proportionally with higher virtual user counts or targeted TPS levels, particularly beyond moderate thresholds. This plateau effect was more pronounced as the system scaled from three to five organisations, suggesting that bottlenecks and increased coordination overheads contribute to this limitation. This is expected in distributed ledger systems because increased concurrency and workload lead to higher resource contention, and queuing delays. Thus, these trends emphasise the need for careful consideration of user concurrency in permissioned blockchain deployments.

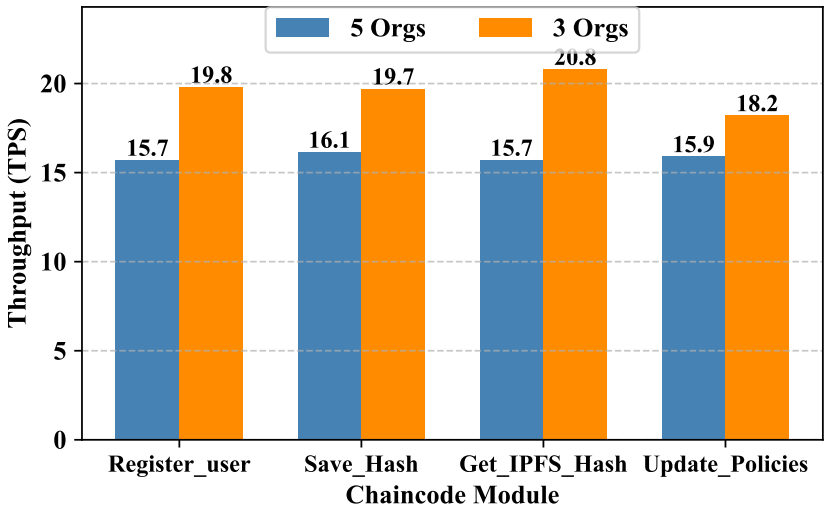


Figure 5.10: **Throughput Performance for 3 vs. 5 Organisations** Throughput (in TPS) of chaincode modules with 100 users and 50 TPS rate for two network configurations: 3 and 5 organisations.

5.5 Usability Assessment

Usability represents a fundamental aspect of system quality, as it directly influences user satisfaction, efficiency, and the overall acceptance of the system. Evaluating usability provides insights into how effectively users can achieve their goals while interacting with the system. In the context of this research, usability is examined in relation to **RQ3:How usable is the proposed blockchain-based data-sharing system from a user perspective?**. The SUS evaluation provides a standardised measure of perceived usability, allowing an initial assessment of how easily users can interact with the system. While usability is an important factor influencing satisfaction and potential adoption, this study focuses specifically on measuring usability rather than directly assessing adoption behaviour.

Prior research has shown that usability remains a major challenge for blockchain-based systems. For instance, Udokwu et al. [129] highlight the complexity of decentralised applications (DApps) and the need for structured design frameworks to reduce usability barriers. Similarly, Saraiva et al. [130] stress the importance of user-friendly interfaces and simplified private key management to expand adoption beyond technically skilled users. Schweiger et al. [131] also emphasise that lowering entry barriers is essential to make blockchain systems accessible to wider audiences. In line with this, Gandhi et al. [132] recommend assessing systems using user-centric methods to ensure that design decisions reflect actual user needs. These works collectively motivate the need to systematically assess usability in blockchain environments.

5.5.1 Method

To evaluate the usability of the system, the *System Usability Scale (SUS)* was employed [133, 134]. SUS is a widely adopted, reliable, and cost-effective questionnaire consisting of ten items that capture users' subjective assessments of usability. The items are deliberately structured in a fixed format, alternating between positively and negatively worded statements. Scores range from 0 to 100, with higher scores reflecting better usability.

There are various usability studies of cryptocurrency wallets and other blockchain applications that use SUS as a formal usability metric [135]. It has been widely applied in the evaluation of DApps, such as studies by Kim et al. [136] and Berne et al. [137], demonstrating its effectiveness in capturing user perceptions of usability in decentralised environments.

5.5.2 Participants

The evaluation involved a group of 100 participants, consisting primarily of students and researchers with general computing knowledge. While most participants had basic familiarity with web-based systems, they were not necessarily experts in blockchain technologies or autonomous vehicle (AV) systems. This reflects a general user perspective rather than a specialised stakeholder group. Prior to the evaluation, participants were provided with a brief introduction to the system's purpose and functionality to ensure they could meaningfully interact with it.

It is important to note that this participant group was selected to provide an initial indication of system usability rather than a comprehensive stakeholder-specific validation. A more extensive evaluation involving domain experts (e.g., automotive engineers, regulators, or industry practitioners) is left for future work.

5.5.3 Procedure

After giving consent and providing their background information, participants were briefed on the objectives of the system and the problem it seeks to address. Following this explanation, they were asked to complete the SUS questionnaire to express their perceptions of the system's usability. They were encouraged to answer candidly, reflecting their understanding of the system's design and intended use.

5.5.4 SUS Questionnaire

The SUS questionnaire comprises 10 statements rated on a 5-point Likert scale, ranging from *Strongly Disagree* to *Strongly Agree.* Following the established SUS scoring procedure, responses were converted into a usability score between 0 and 100. The complete consent and survey form, including the SUS questionnaire, is included in Appendix A.

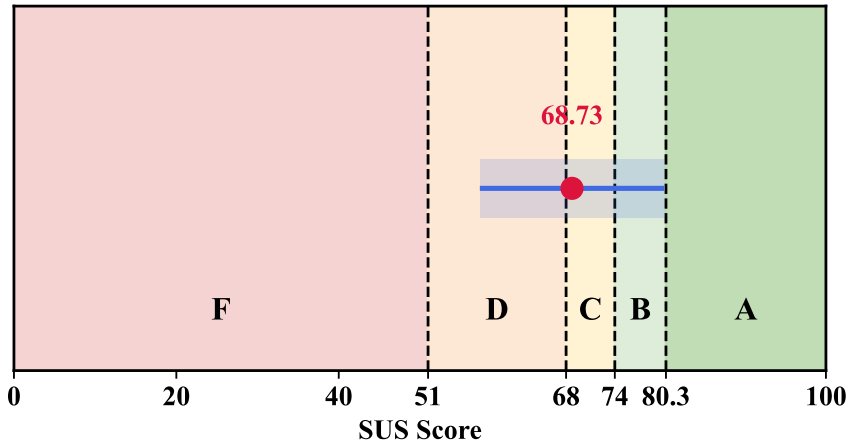


Figure 5.11: Position of the obtained SUS score on the standard SUS grading scale. The mean score (68.73) with standard deviation (SD = 11.35) is illustrated to represent the variability of user responses. The result falls within grade C and is slightly above the average usability benchmark of 68.

5.5.5 Result

The usability evaluation produced a mean SUS score of 68.7 ± 11.35, where the value after ± represents the standard deviation across 100 participants. As illustrated in Fig. 5.11, the obtained score lies within the "C" grade range, indicating acceptable usability that is slightly above the standard SUS average of 68, but below the threshold for high usability. The observed variability is low to moderate, suggesting that participants had reasonably consistent perceptions of the system, although some differences in user experience were present. This indicates that users perceive the system as having an overall acceptable level of usability. In relation to **RQ3**, the result provides an initial indication that users are able to interact with the system without significant usability barriers.

However, it is important to note that SUS measures perceived usability and does not directly assess user satisfaction or adoption behaviour. Therefore, while acceptable usability may contribute to a positive user experience, further studies would be required to evaluate its impact on long-term adoption and stakeholder acceptance.

The result is also consistent with recent usability studies in blockchain systems. For example, Berne et al. [137] reported an average SUS score of 66.25 in DAO platforms, which also indicates acceptable usability. However, such scores also suggest that there is room for improvement in enhancing overall usability and user experience.

5.6 Conclusion

This chapter presented the baseline performance evaluation of the system across blockchain and API layers in three and five organisation networks. The results showed that the system is functionally reliable and capable of sustaining moderate workloads, but throughput plateaued

under higher user loads and transaction rates, with additional decline as the network scaled to five organisations. These findings provide essential benchmarks and reveal scalability bottlenecks

Beyond performance, this chapter also addressed the system's usability, aligning with the research question on system usability (RQ3). Using the System Usability Scale (SUS), the evaluation indicates that users perceive the system as having acceptable usability while also leaving room for improvements in interface design and ease of interaction.

Overall, this evaluation not only validated the baseline feasibility of the proposed architecture but also highlighted both its strengths, reliability, security, and moderate scalability, and its limitations under increased load and chaincode complexity. These insights guide the next phases of experimentation, particularly in testing endorsement policies.

Chapter 6

Customised Endorsement Policy

This chapter builds on the system implementation and evaluation presented in the previous chapter. In particular, it investigates how the system can be further customised and configured to enforce secure and controlled data sharing among multiple stakeholders. It focuses on the management of stakeholder influence and ownership rights within the features of Hyperledger Fabric. In addition, it presents the design, integration, and evaluation of a customised endorsement policy, implemented as a plugin within the proposed system. By examining how different configurations impact throughput, latency, and scalability in multi-stakeholder AV data-sharing scenarios, this chapter directly addresses the Research Question **RQ2a and b**. Section 6.1 provides the motivation and objectives, while Section 6.2 illustrates the design and implementation of the customised endorsement policy. Section 6.3 describes the verification methodology, Section 6.4 presents the evaluation results, and finally Section 6.5 concludes the chapter.

6.1 Motivation for Customisation

As discussed in Chapter 2, data management in AV ecosystems presents critical challenges that require urgent attention, including data storage, data sharing among stakeholders, data privacy, data integrity, and security. Our extensive review of the literature has highlighted the ownership of data as an additional critical issue that is often overlooked. Establishing clear, transparent, and equitable ownership frameworks is crucial for building trust and fostering cooperation among all parties involved in the deployment of AV.

The ownership of AV data varies internationally, shaped by the country-specific legal and regulatory landscapes. For example, in the United Kingdom, legislation such as the Automated and Electric Vehicles Act 2018 and the forthcoming Automated Vehicles Bill primarily address AV safety and liability, but do not directly define data ownership [138]. Although the UK GDPR and the Data Protection Act 2018 regulate the processing of personal data, they do not assign legal ownership of AV-generated data. In practice, manufacturers or service providers often control these data through service-level agreements.

In the United States, vehicle manufacturers typically claim ownership over the data

collected by AVs [139]. This control is often supported by intellectual property rights over software systems embedded in vehicles, which grants manufacturers authority over how data is accessed and used. Although future regulatory changes may shift this balance due to rising privacy concerns, manufacturers currently maintain predominant control.

European Union regulations, particularly the GDPR, emphasise the protection and processing of personal data rather than assigning ownership in the traditional sense. Under GDPR, data controllers, often manufacturers or service providers, determine how data are used, while individuals retain the right to access, correct, or delete their personal data in specific contexts [140]. The regulation also requires businesses to integrate data protection principles into their operations, which directly affects how AV data is managed and governed.

Building on these regulatory approaches, it is essential to recognise the central role of manufacturers in managing AV data. Since they are typically responsible for collecting, processing, and securing the data, they are best positioned to ensure proper handling and protection in accordance with both technical standards and legal obligations.

When designing a data sharing framework based on HLF, governance mechanisms, such as endorsement policies, which define who has the authority to validate transactions, play an inherent role in shaping the concept of data ownership. EPs in HLF are fundamental to its trust and transaction validation mechanisms. They determine which peers are required to approve a transaction before it can be committed to the ledger. The structure and logic of these policies directly impact system security, trust assumptions, and organisational dynamics. As highlighted in [141, 142, 143, 144], the choice and configuration of EPs influence both the trust model between participating entities and the overall security posture of the network. Thus, endorsement policies not only regulate transaction validation but also become a mechanism to encode stakeholder power and ownership rights directly into the system's trust fabric. However Fabric's default endorsement mechanisms, while functional for basic multi-organisation scenarios, fall short when applied to such nuanced ownership models. Thus, we define custom endorsement policies that go beyond Fabric's default static policies. Customisation is needed in AV contexts to reflect asymmetric trust relationships, critical safety considerations, and regulatory compliance requirements, ensuring that data governance aligns with real-world responsibilities and constraints.

The primary objectives of this chapter are:

1. Design and implement a custom endorsement policy that reflects asymmetric authority among participating organisations, particularly privileging a key stakeholder (e.g., the manufacturer).
2. To evaluate the impact of the customised EP on system performance, specifically in terms of transaction latency, throughput, and success rate, under varying workloads.
3. Explore the trade-offs between decentralisation and efficiency by comparing the EP plugin with default endorsement policies.
4. Provide insights into how endorsement policies can be tuned to balance fairness, trust, and performance in real AV data-sharing ecosystems, thereby linking technical evaluation with practical governance considerations.

Thus, this chapter extends the exploration of **RQ2a** by investigating the impact of different endorsement policies on the system performance.

6.2 Design and Implementation of the Custom EP

To align with the specific considerations of the study. `Org1`, which is assumed to represent the manufacturer, is given endorsement privileges to reflect its critical role in the ownership and governance of data. To enable programmable and flexible enforcement of this policy, a separate `YAML` file was implemented that explicitly specifies the use of the `ESCC` (Endorsement System Chaincode) and `VSCC` (Validation System Chaincode). The custom EP is defined as in 6.1. This policy indicates that: *any member of Org1, or any three of the remaining four members of the organisation (Org2 to Org5), must sign the transaction.*

Listing 6.1: Customised Endorsement Policy

```
EndorsementPolicy:
  type: Signature
  rule: |
    {
      "identities": [
        { "role": { "name": "member", "mspId": "Org1MSP" } },
        { "role": { "name": "member", "mspId": "Org2MSP" } },
        { "role": { "name": "member", "mspId": "Org3MSP" } },
        { "role": { "name": "member", "mspId": "Org4MSP" } },
        { "role": { "name": "member", "mspId": "Org5MSP" } }
      ],
      "policy": {
        "1-of": [
          { "signed-by": 1 },
          {
            "3-of": [
              { "signed-by": 2 },
              { "signed-by": 3 },
              { "signed-by": 4 },
              { "signed-by": 5 }
            ]
          }
        ]
      }
    }
```

The first section in the code is the **Identities Section**, which lists the five organisations in the network. The second section specifies the actual endorsement rules. The `1-of` clause at the top level means that either one of the following conditions must be met: `"signed-by": 1` → This refers to **Org1MSP** (it is important to note that indexing starts at 0, but for simplicity we refer to Org1 as 1 throughout this thesis). Thus, if Org1 alone endorses the transaction, it is valid. The `3-of` clause → At least three out of the four organisations (Org2, Org3, Org4, Org5) must endorse the transaction.

Although this introduces a degree of centralisation, it is a deliberate and context-driven

decision to reflect the manufacturer's responsibility to manage sensitive operational data. It is important to note that this endorsement structure reflects one plausible governance model within an AV data-sharing ecosystem, where the manufacturer is assigned a dominant role due to its responsibility for system design, data collection, and regulatory compliance. This assumption is used to guide the experimental design and evaluate its impact on system performance. However, this hierarchy is not intended to be prescriptive. In real-world deployments, alternative distributions of endorsement authority may be adopted depending on stakeholder agreements, regulatory frameworks, or trust relationships among participants. To avoid creating a single point of failure or blocking transactions when Org1 is unavailable, the second part of the policy ensures that a transaction can still be endorsed by any three out of the remaining four organisations (Org2 to Org5). This fallback mechanism maintains operational continuity and balances governance control with collaborative decision-making across the network.

In addition, the design addresses concerns related to transaction failure and scalability. As the number of organisations and required endorsements increases, the likelihood of transaction delays or failures also rises, particularly in environments with high network latency or peer unavailability. Implementing a simplified endorsement strategy enhances throughput and strengthens the resilience of the system under load [145]. Thus, by adopting a more flexible policy that reduces the number of required endorsements, the system can handle higher workloads while minimising bottlenecks and maintaining operational continuity across the network.

6.3 Verification

To ensure that the customised EP behaves as designed and meets the requirements of the AV data sharing use case, a structured verification process is followed as follows:

1. The process begins by defining the new EP tailored to the AV data-sharing scenario. The policy is constructed to reflect the desired control, trust, and operational conditions that govern endorsement decisions.

2. Once defined, the EP is formally specified using a verification tool to model its logic. Following the methodology proposed by Kawahara [146], the Z3 solver [147] is used to express the policy as logical constraints. These constraints capture the exact endorsement requirements, including conditional rules and fallback mechanisms, and the model is verified using satisfiability (SAT) checks to confirm correctness across all relevant scenarios.

3. After the logical model has been validated, performance tests are performed to evaluate how the policy performs in practice. Hyperledger Caliper [125] is used to measure throughput, latency, and success rate in realistic workloads. While Z3 ensures correctness of the logical model, Caliper validates that these rules hold under real workloads, thereby combining formal soundness with practical validation.

4. Finally, the EP undergoes continuous evaluation to ensure it remains aligned with the use case objectives.

These verification steps collectively ensure that the customised endorsement policy is both logically correct and practically effective. By combining formal modelling with performance testing, the approach provides confidence that the policy will enforce the intended control rules in the AV data-sharing network while maintaining operational efficiency under realistic conditions.

6.4 Evaluation

This section evaluates the customised endorsement policy introduced for the AV data-sharing use case. The evaluation aims to confirm that the policy effectively enforces the desired control and trust model while maintaining acceptable performance levels in the operational environment.

6.4.1 Evaluation Objectives

Based on the baseline evaluation objectives established in Chapter 5, Section 5.1.1, this evaluation investigates how modifying the endorsement threshold affects the system performance and operational behaviour. The specific objective is to measure how the customised endorsement policy (EP) influences the performance, scalability, and overall robustness of the proposed system. Although this section provides a comprehensive evaluation of the configuration introduced in this chapter, a detailed comparative analysis of alternative configurations is presented in Chapter 8.

Test Scenarios

The evaluation was carried out on the same smart contract functions used in the prior approach: `Register_User`, `Save_hash`, `Git_IPFS_hash`, and `Update_Policies`. These functions were tested in various scenarios to observe system behaviour and detect any adverse performance impact resulting from the modified EP. The scenarios include:

1. **User Load Variation** Different user loads were simulated to benchmark system responsiveness and processing capability. This approach mirrors the setup used in the earlier evaluation phases to provide comparable results, and what if the EP change introduces regressions?
2. **Transactions Per Second (TPS)** The system was tested with increasing TPS levels to evaluate the network's ability to maintain stability and performance under a more relaxed EP configuration.
3. **Data Upload Operations** End-to-end upload tests were conducted to evaluate both backend and blockchain behaviour under realistic workload conditions. This test aimed to capture the effect of EP changes on user-facing interactions and the overall scalability of the system.

Each scenario was executed over multiple iterations to ensure statistical reliability and to identify system limits under repeatable, high-stress conditions. The results are used to determine the practical trade-offs between reduced endorsement complexity and system performance.

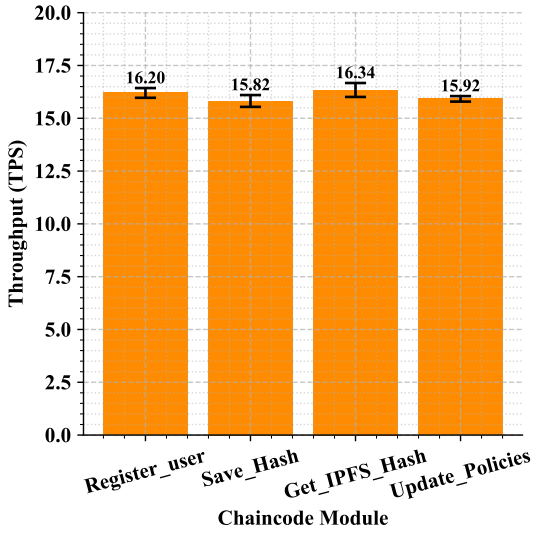


Figure 6.1: Throughput Across Chaincode Modules for 100 User Load and 50 TPS Rate Scenario. Error bars represent the standard deviation (SD) across repeated experimental runs.

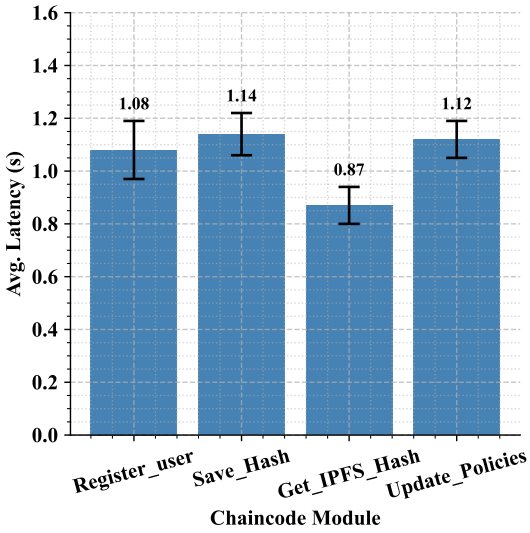


Figure 6.2: Average Latency Across Chaincode Modules for 100 User Load and 50 TPS Rate Scenario, Error bars represent the standard deviation (SD).

6.4.2 Results and Discussion

To evaluate the performance of the implemented chaincode logic, four core modules were tested: `Register_user`, `Save_Hash`, `Get_IPFS_Hash`, and `Update_Policies`. The tests were conducted multiple times under identical conditions to ensure accuracy and consistency in the observed results. Results indicate relatively consistent throughput (transactions per second, TPS) across all four modules. For example, in a moderate load with 100 users and a 50 TPS rate, throughput values range between 12.2 and 12.7 TPS as illustrated in Figure 6.1. This demonstrates the system's stable processing capacity irrespective of the specific chaincode function being invoked. In addition, the average latency (in seconds) is measured and presented in Figure 6.2 for the same test scenario.

Slight variations were observed; for example, the *Register_user* and *Get_IPFS_Hash* modules exhibited the lowest average latency, while *Update_Policies* and *Save_Hash* had relatively higher values. These differences may be attributed to the complexity of internal logic and the nature of ledger interactions within each module. Overall, the system shows balanced performance across key operations in this moderate scenario, supporting its reliability and responsiveness for core transaction types.

Another important observation is that varying user loads show a clear inverse relationship

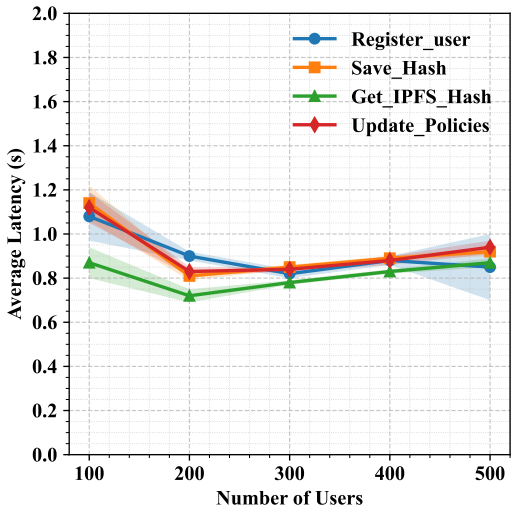


Figure 6.3: **Throughput vs User Load** This figure illustrates the Throughput across all chaincode modules under varying user loads ranging from 100 to 500 users and a 50 TPS rate. The shaded regions represent the standard deviation (SD) across repeated experimental runs.

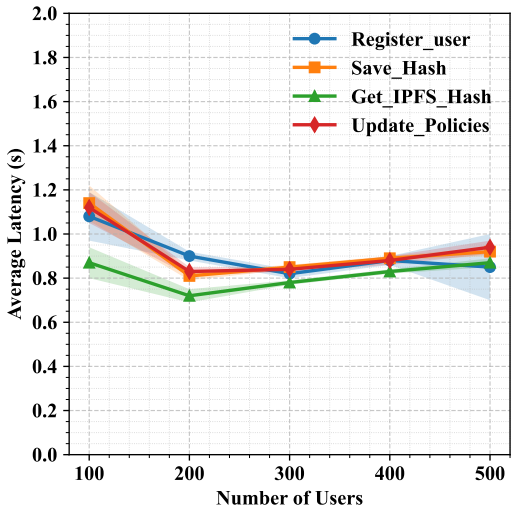


Figure 6.4: **Latency vs User Load** This figure illustrates the average latency (in seconds) for four chaincode modules under varying user loads ranging from 100 to 500 users and a 50 TPS rate. The shaded regions represent the standard deviation (SD) across repeated experimental runs.

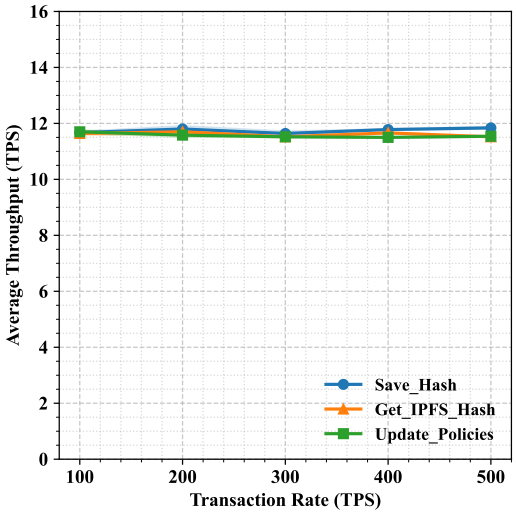


Figure 6.5: **Transaction Rate vs Throughput**. This figure illustrates the relationship between the requested transaction rate (TPS) and the achieved throughput among different modules. The shaded regions represent the standard deviation (SD).

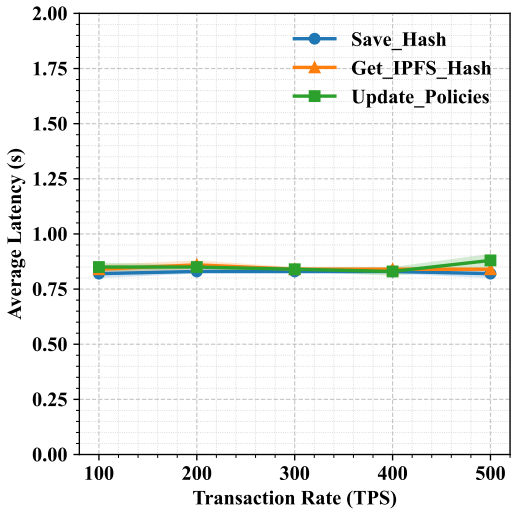


Figure 6.6: **Transaction Rate vs Average Latency** This figure shows how the average latency, measured in seconds, changes as the transaction rate (TPS) increases. The shaded regions represent the standard deviation (SD).

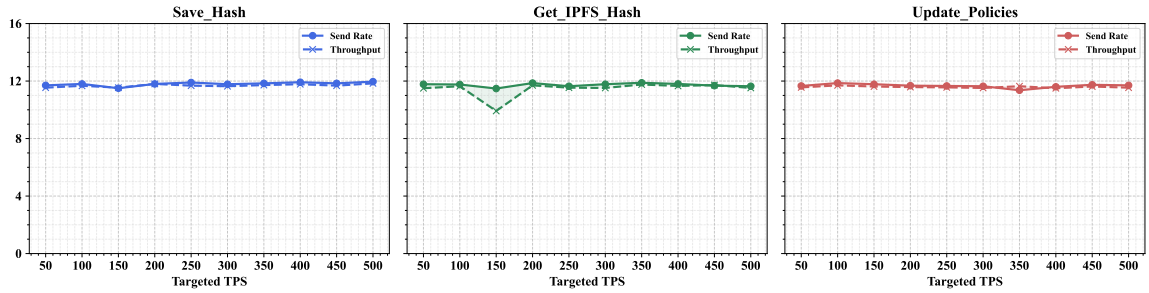


Figure 6.7: **Send Rate and Throughput vs Targeted TPS**. This figure compares the Send Rate and Throughput across different chaincode modules.

between the number of users and throughput. As illustrated in Figure 6.3, throughput steadily declines across all four chaincode modules as the number of users increases from 100 to 500. The latency trend shown in Figure 6.4 highlights the system's scalability limitations and the increasing transaction latency under heavier user loads. While the latency values for all modules remain close, the *Update_Policies* module shows the highest latency at 500 TPS.

On the other hand, Figure 6.5 illustrates the effect of increasing the requested transaction rate (TPS) on the actual throughput achieved by various chaincode modules. The results indicate that the system maintains a relatively stable throughput of approximately 11.5 TPS across varying input rates ranging from 100 to 500 TPS with a fixed user load. This plateau suggests that the system has reached a performance ceiling beyond which increasing the transaction rate does not yield higher throughput. In addition, the shaded regions representing the SD are relatively narrow across all configurations, indicating low variability across repeated experimental runs.

Furthermore, Figure 6.6 presents the average latency observed for the same modules and the same scenario. Overall, all tested modules exhibit relatively stable latency values, consistently remaining below 1 second even at higher loads. This indicates that the system is capable of maintaining low latency under moderate scaling conditions. In addition, the shaded regions representing the SD are relatively narrow, indicating low variability across repeated experimental runs. However, at 500 TPS, the *Update_Policies* module records the highest latency among the coaincode operations, reaching 0.88 s.

While the figures highlight that throughput and latency remain relatively stable, it is also important to consider that the fact that both the send rate and the achieved throughput remain fixed at approximately 11.5 TPS across all requested input rates suggests that the system has reached a saturation point., Thus, to further investigate the observed throughput limit and identify the factors contributing to system bottlenecks, a series of targeted experiments was conducted by varying one parameter at a time. First, in the same previous scenario, the number of users was set at 200, while the targeted transaction rate (TPS) was increased from 50 to 500, but focusing on the send rate values. As illustrated in Figure 6.7, both the send rate and the achieved throughput remained nearly constant, with throughput plateauing at approximately 11.5 TPS for all tested modules. The minimal difference between the send rate and throughput in this setup suggests that the system was operating at its maximum processing capacity, and any additional input load does not translate to increased throughput.

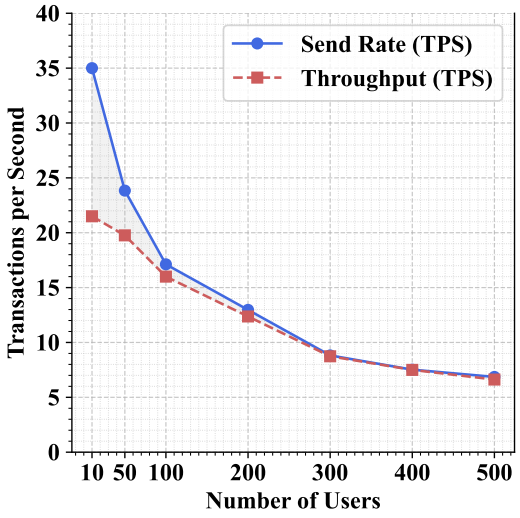


Figure 6.8: **Send Rate and Throughput vs Number of Users**. This figure illustrates how the Send Rate and Throughput change as the number of users increases from 10 to 500, with the targeted transaction rate fixed at 50 TPS.

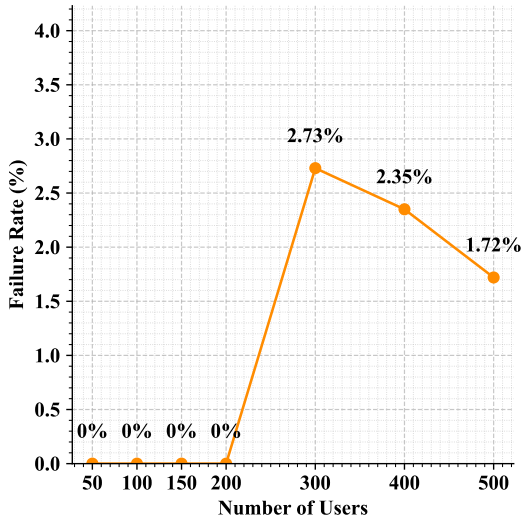


Figure 6.9: **Failure Rate vs Number of Users**. In the `update_Policy` module with a fixed TPS rate of 50, the failure rate remained at 0% for up to 200 users and increased slightly for higher user counts.

Similarly, to evaluate the effect of user concurrency, we have decreased the number of users to 10 and fixed the TPS to 50 while gradually increasing the number of users. As shown in Figure 6.8, both the send rate and the throughput decreased with increasing user counts. The gap between the two curves becomes narrower and more evident with higher user counts, reflecting performance limitations under heavier loads.

These two findings highlight important limitations. First, the system cannot handle high TPS; there is a fixed ceiling. Second, more users mean more pressure, which slows both the sending and processing of transactions.

Using that method of gradually adjusting the input parameters and observing the results, we were able to identify exactly where and why performance starts to degrade, which is critical for understanding the scalability and limitations of the system.

In the tested configurations in the previous chapter, while the system handled many users without failing in simpler modules, the more complex module, `update_policy`, began to show a few failures as the user numbers grew.

It is essential to evaluate the same scenario using the EP approach presented in this chapter. Figure 6.9 illustrates the failure rate in a variety of concurrent users, with the system configured to operate at a constant transaction rate of 50 TPS. To precisely identify the point at which the system begins to degrade, the user load was gradually increased in increments. The results show that the failure rate remained at 0% for user loads up to 200, indicating stable performance. However, as the number of users exceeded 200, the system exhibited a slight increase in the failure rate, settling at 3.13% with 400 users. That suggests that under

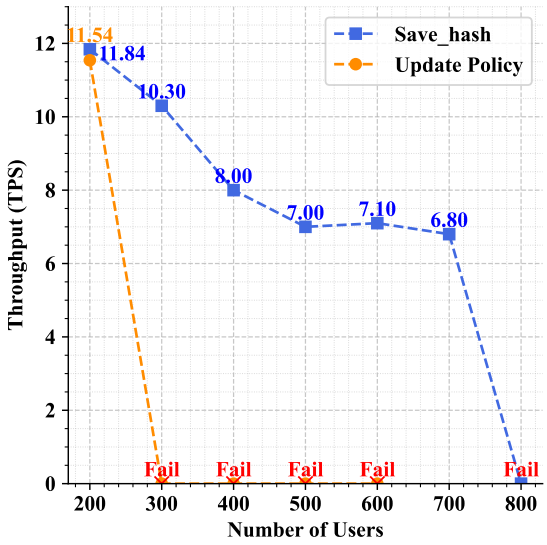


Figure 6.10: **Throughput under Stress Scenario** The Throughput under 500 TPS for `Save_hash` and `Update_Policy`, `Save_hash` throughput decreases as user load increases, while `Update_Policy` fails from 300 users onward.

heavy concurrent usage, modules with heavier logic are more likely to hit blockchain-side bottlenecks like proposal delays or endorsement timeouts. For a better understanding of the behaviour of the system under varying transaction loads, we designed a series of targeted stress scenarios across multiple chaincode modules. Each scenario incrementally increased either the number of users or the TPS. The tests were aimed at exposing performance bottlenecks, functional limitations, and error thresholds specific to each module. The findings were as shown in figure 6.10, `Save_hash` handled higher user count with fewer functional failures but showed transaction timeouts beyond 600 users at 500 TPS. However, `Update_Policies` failed for 500 TPS even for only 300 user counts, indicating that the logic is more sensitive to concurrency and load. These failure thresholds are dependent on the specific Hyperledger Fabric configuration and underlying hardware settings, and should therefore not be interpreted as absolute system limits. In addition, scenarios reporting a 100% failure rate indicate that the system was unable to complete the workload due to overload, rather than all transactions being individually processed and rejected.

In terms of system scalability and data upload efficiency, Figures 6.11 and 6.12 together illustrate the impact of increasing user concurrency on the performance and reliability of the `Save_Hash` module when uploading a 3GB file. This scenario was simulated using the `K6` tool, which represents the frontend upload process and the submission of the corresponding IPFS hash to the blockchain.

Figure 6.11 shows the average time taken to complete a successful data upload operation. As the number of concurrent users increases, this time increases significantly, indicating that system resources such as bandwidth, memory, or processing capacity become increasingly saturated. For smaller user loads (e.g. 200–400 users), the system handles 3 GB uploads efficiently in a few seconds. However, beyond 500 users, the average duration begins to spike, reflecting traffic queuing, contention, or processing delays. Although duration appears

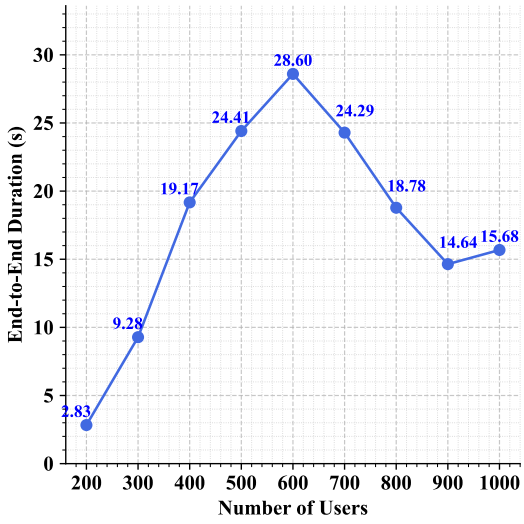


Figure 6.11: Average duration (seconds) for successful upload and hash saving requests as the number of users increases. Shaded regions represent the standard deviation (SD) to illustrate variability.

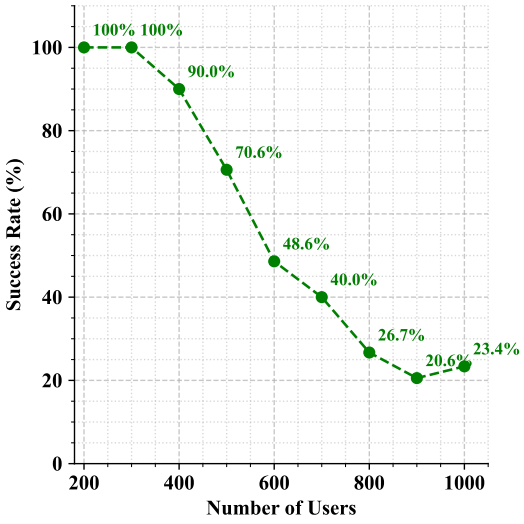


Figure 6.12: Success rate (%) of upload requests with increasing user load.

to decrease again after peaking, it is important to note that this aligns with a significant drop in the success rate, as shown in Figure 6.12. This suggests that fewer transactions are successfully completed under heavy load, which skews the average duration downward due to early failures or timeouts.

6.5 Conclusion

In this chapter, we designed, implemented, verified with formal tools, and evaluated Approach 2 Endorsement Policy, a customised policy tailored to the use-case requirements. This EP explicitly reflects the authority of Organisation 1, representing the manufacturer, in exercising greater control over data access and validation. The evaluation of the core modules was conducted under varying user loads and transaction rates, measuring latency, throughput, send rate, and success rate. The results demonstrated that the system largely retained the stable characteristics observed under the default EP, while showing notable improvements in certain areas, particularly the success rate under higher user loads, where the customised EP reduced transaction failures. These findings confirm that the customised EP effectively supports both reliability and throughput, while also illustrating how policy adjustments can optimise performance in a multi-organisation blockchain environment. In addition, these results highlight the potential of tailoring endorsement policies as a governance tool, not only a performance parameter, in multi-stakeholder AV ecosystems. Beyond validating feasibility, the evaluation provides actionable insights for refining policy configurations and informs the design of future deployment scenarios in more complex, real-world settings.

Chapter 7

Simulated Weighted-Voting Endorsement Policy

Building on the previous chapter, this chapter further examines the customisation of EP to address the challenge of data ownership and control in the management of AV data. Specifically, this chapter explores Weighted Voting (WV) as a concept to simulate varying levels of stakeholder influence within endorsement policies. By embedding weighted voting logic into EP design, the research addresses not only technical trust but also business-level authority and ownership concerns surrounding AV data. This ensures that the blockchain platform reflects the real-world power and responsibility hierarchies found in AV ecosystems, supporting regulatory accountability while respecting the higher control manufacturers have over data and decision-making. Unlike standard HLF EP mechanisms, which treat all peer endorsements equally and lack mechanisms for role prioritisation or weighted influence, weighted voting allows certain stakeholders, particularly manufacturers, to exercise stronger decision-making power. This capability is particularly important given the limitation identified in the custom EP presented in Chapter 6, where Org1 alone could validate transactions. Although the EP enabled operational continuity and simplified endorsement, it concentrated validation authority, creating potential centralisation concerns. Weighted voting provides a more nuanced approach, allowing manufacturers to retain influence without granting unrestrained control. This chapter further addresses **RQ2a-b** by designing, implementing, verifying, and evaluating an alternative endorsement mechanism.

In this chapter, section 7.1 provides background on the related work of existing voting consensus mechanisms. The assumption section (7.2 outlines the key assumptions that underpin the proposed model. The core of the chapter lies in 7.3, where it describes the method and implementation of Weighted Voting simulation within EP design. Sections 7.4 and 7.5 present the verification and evaluation of the proposed solution, respectively. Finally, the chapter ends with the Conclusion section 7.6.

7.1 Background

Weighted voting is highlighted in [148] as a method of voting consensus mechanisms. It discussed how the weight could be assigned to stakeholders who have different stakes or authority in the decision-making process. This method has been used in this scenario in various fields, for example, in [149], where weighted voting is used in the AWARE algorithm to improve the performance of practical Byzantine fault tolerance (PBFT). In particular, it assigns larger voting weights to the well-connected nodes to give them more power in the consensus process. Similarly, in a supervised machine learning classifier, a weighted voting system [150] has been added to modify the K-Nearest Neighbours (KNN) algorithm, which is a supervised machine learning algorithm used for classification and regression tasks. In that work, the closer neighbours have a greater influence on the final classification. Zhoe et al. [151], have used weighted voting in collaborative physical layer authentication (CPLA) by assigning different weights to the decisions made by cooperators based on their reliability and performance. Thus, instead of treating all cooperators equally, the proposed scheme evaluates their past authentication capabilities and assigns weights accordingly. While several studies have proposed modifications to the endorsement policy (EP) within Hyperledger Fabric (HLF) and evaluated their performance [143, 142, 141, 144], none have explicitly integrated the concept of weighted voting into HLF's endorsement process. Incorporating weighted voting introduces a more flexible and balanced trust model, where peers contribute to endorsement decisions in proportion to their assigned weights, reflecting their authority, reliability, or role within the network. Building on the traditional endorsement of HLF, which is based on binary policies using logical AND/OR combinations of peers, this research chapter designs, implements, and evaluates a weighted endorsement simulation that augments these policies by assigning stakeholder-specific weights and evaluating weighted quorums within AND/OR structures.

7.2 Assumptions:

The following are the key assumptions that form the basis of the design, implementation, and evaluation of the proposed system.

1. Trust Model Stability: Organisations accept their assigned weights as representative of their influence in the network channel.
2. Fixed weights: In our case, we assume that the weight assigned to the organisations does not often change.
3. Illustrative Weighting Scheme: The weights assigned to organisations in this study are illustrative and exploratory, and do not represent fixed or universally applicable values. They are used to simulate varying levels of stakeholder influence and to evaluate the impact of weighted endorsement on system performance. In real-world deployments, these weights could be defined differently based on governance agreements, trust relationships, regulatory requirements, or operational considerations.
4. Governance Rationale: The endorsement authority hierarchy and assigned weights in this study represent one plausible governance model for AV ecosystems, where the

manufacturer is assumed to have greater authority due to its responsibility for system design, data generation, and regulatory compliance. The selected weights (e.g., 2 for Org1 and 1 for others) are intentionally minimal and illustrative, providing a simple way to model differentiated influence while preventing unilateral control. This allows the manufacturer to have a stronger influence, but still requires at least one additional organisation to validate transactions. In real-world deployments, alternative authority distributions and weighting schemes may be defined based on governance agreements, trust relationships, or regulatory requirements, as commonly seen in weighted voting systems [148, 149]

7.3 Method and Implementation

Hyperledger Fabric does not natively support direct numerical weighting in its endorsement policies. Instead, it provides a flexible policy language based on logical operators such as `AND`, `OR`, and `OUTOF`, which allow the expression of composite endorsement requirements. To approximate weighted voting within this environment, we designed an alternative simulation method that takes advantage of enumerating valid endorsement combinations and, optionally, repeating MSP identities only when multiple peers exist in an organisation.

7.3.1 Simulation Strategy

The objective is to implement a *Weighted Endorsement Policy (WEP)* where each organisation (peer identity) contributes a different weight or number of votes to the final endorsement decision. The following scenario was defined:

- **Org1MSP** is assigned 2 votes.
- **Org2MSP** to **Org5MSP** are each assigned 1 vote.
- A transaction is considered valid if the collected votes sum to 3 or more.

To implement weighted voting with single peers per org is to enumerate all valid AND clauses explicitly, with the weights implied in the logic.

7.3.2 Implementation in Fabric

The endorsement policies are defined in Fabric using policy syntax such as:

```
AND('Org1MSP', 'Org2MSP')
```

In this example, `Org1MSP` contributes two votes due to its implicit weight assumption, and `Org2MSP` contributes one vote, achieving the required threshold of three votes for a valid endorsement.

7.3.3 Policy Construction

Valid endorsement combinations include:

- `Org1MSP (2) + Org2MSP (1)`
- `Org1MSP (2) + Org3MSP (1)`
- `Org1MSP (2) + Org4MSP (1)`
- `Org1MSP (2) + Org5MSP (1)`
- `Org2MSP (1) + Org3MSP (1) + Org4MSP (1)`
- `Org2MSP (1) + Org3MSP (1) + Org5MSP (1)`
- `Org2MSP (1) + Org4MSP (1) + Org5MSP (1)`
- `Org3MSP (1) + Org4MSP (1) + Org5MSP (1)`

These combinations are then combined using the logical `OR` operator to form the full endorsement policy expression. The complete policy used in the simulation is:

Listing 7.1: Complete Weighted Endorsement Policy

```
Endorsement:
  type: Signature
  rule: |
OR(
  // Org1 (2) + any one other org (1) = 3
  AND('Org1MSP.member','Org2MSP.member'),
  AND('Org1MSP.member','Org3MSP.member'),
  AND('Org1MSP.member','Org4MSP.member'),
  AND('Org1MSP.member','Org5MSP.member'),

  // Any three non-Org1 orgs = 3
  AND('Org2MSP.member','Org3MSP.member','Org4MSP.member'),
  AND('Org2MSP.member','Org3MSP.member','Org5MSP.member'),
  AND('Org2MSP.member','Org4MSP.member','Org5MSP.member'),
  AND('Org3MSP.member','Org4MSP.member','Org5MSP.member')
)
```

This construction ensures that only those combinations of endorsements that collectively meet or exceed the required threshold (3 votes) result in successful transaction validation.

7.4 Verification

In the EOV blockchain context, the endorsement policy (EP) can become a potential vulnerability, affecting data integrity, network trust, availability, and system efficiency. To address these concerns, a structured verification and validation approach is adopted for the newly introduced policies as follows:

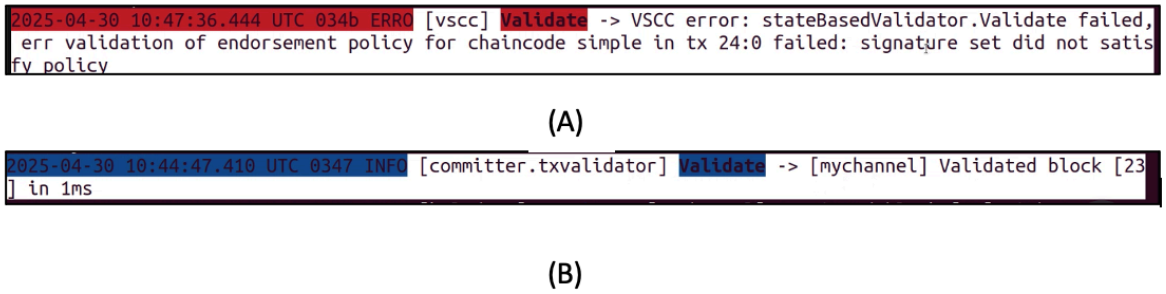


Figure 7.1: **(A)** Transaction failure due to endorsement by two organisations excluding Org1, which does not fulfil the criteria of the WV EP. **(B)** Successful transaction endorsement where Org1 and one other organisation endorsed, satisfying the logic-based WV endorsement policy.

1. Begin by defining the new endorsement policy to align with the specific needs of the use case, such as AV data sharing.
2. Then, use a formal specification tool to model the policy and verify that it accurately captures the required conditions. Following the method outlined by Kawahara [146], the Z3 SMT solver [147] is employed to encode the EP as a set of logical constraints, reflecting the necessary endorsement logic. A comprehensive model is built to describe transaction endorsement scenarios and is validated through satisfiability (SAT) checks to confirm practical viability.
3. The EP is further tested under various combinations of endorsing peers. For example, endorsements from two organisations that exclude Org1 result in a failed transaction (Figure 7.1-A), whereas endorsements from Org1 along with another organisation satisfy the conditions and lead to successful transactions (Figure 7.1-B).
4. Once the modelling is complete, Caliper is used to run performance benchmarks, evaluating the policy's impact on throughput, latency, and scalability to confirm it meets performance expectations.
5. Finally, the EP is continually assessed for compliance with the use case through ongoing evaluation, monitoring, and policy analysis.

7.5 Evaluation

This section presents the evaluation of the proposed system under the newly defined endorsement policy.

7.5.1 Evaluation Objectives

Building on the baseline evaluation objectives established in Chapter 5, Section 5.1.1, the central focus of this evaluation is to analyse the impact of a modified Endorsement Policy (EP) configuration on system performance under similar test scenarios. Specifically, the evaluation aims to determine whether the modified EP affects overall system performance and to examine scalability in terms of the number of concurrent users, transaction rates (TPS), and the volume of data uploaded to IPFS. This section focusses on evaluating the performance and behaviour of the blockchain layer under the proposed endorsement policies, ensuring that the EP not only enforces the required trust model, but also maintains acceptable blockchain performance levels. The same benchmarking tool, Caliper, was used as in Chapter 5 to generate workloads and capture system metrics, thereby providing a consistent basis for comparison between the baseline and the modified EP evaluation. While this section provides a comprehensive evaluation of the configuration introduced in this chapter, a detailed comparative analysis of alternative configurations is presented in Chapter 8.

Test Scenarios

Each function: `Register_User`, `Save_hash`, `Git_IPFS_hash`, and `Update_Policies` was tested under varying scenarios to measure how well the system handles different stress levels in a multi-organisation setting. The scenarios were designed to identify potential performance bottlenecks introduced by the new EP configuration and investigate whether that throughput remains within acceptable bounds as the network grows in complexity. The key factors include:

1. **Number of Users.** Multiple user loads were simulated to assess the system's throughput and latency when subjected to increasing demand. We aim to conduct the same user load as in the previous chapters to benchmark the performance.
2. **Varying TPS.** The system was stressed with varying TPS rates. This phase also expanded the TPS range from the previous baseline limit to 500 TPS to observe how well it could maintain performance at different input rates with the new EP.
3. **Upload Data.** An end-to-end functional stress test has been conducted to assess both the backend and blockchain layers to get insight into how stress affects user experience and system scalability.

Each scenario was executed in multiple rounds to account for result consistency and to identify potential performance thresholds.

7.5.2 Results and Discussion

Figure 7.2 compares the performance of the four chaincode modules: `Register_user`, `Save_Hash`, `Get_IPFS_Hash`, and `Update_Policies`, in terms of throughput under a fixed

test condition of 100 users and 50 TPS. The results show close throughput across the modules, which is relatively lower than the results observed in the previous configuration.

In addition, the results in Figure 7.3 show that the average latency across the modules ranges from 1.36 to 1.5 seconds. In terms of the module complexity, as in the previous tests, `Get_IPFS_Hash` module exhibits the lowest latency (1.36s), due to its simple data retrieval operation, while the `Update_Policies` module has the highest average latency (1.475s), also attributed to the complexity of modifying access control policies, which contain read and write operations. Overall, the results demonstrate that latency and throughput are relatively balanced across modules, with no strong variation observed.

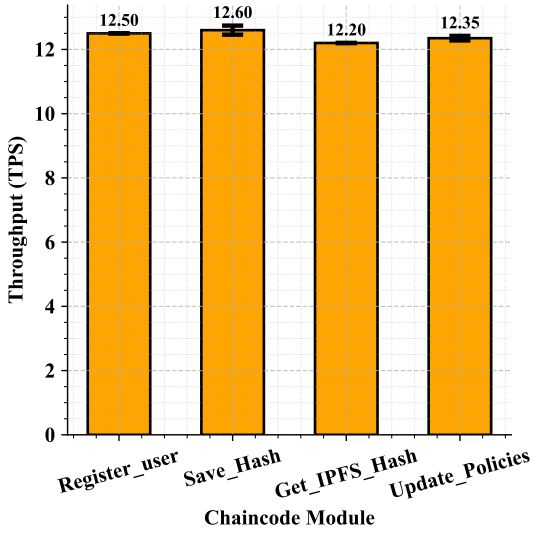


Figure 7.2: Throughput Across Chaincode Modules under a 100-User Load and 50 TPS Rate Scenario with error bars.

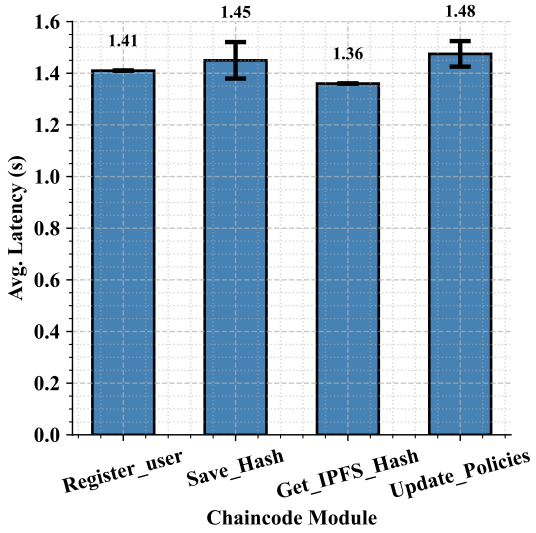


Figure 7.3: Average Latency Across Chaincode Modules under a 100-User Load and 50 TPS Rate Scenario with error bars

As the number of users increases, there is a clear increase in average latency and a decrease in throughput, as illustrated in Figures 7.4 and 7.5. The figures present the average latency observed for the `Save_Hash` and `Update_Policies` chaincode modules under increasing user loads, ranging to 500 users. For `Save_Hash`, the latency increases steadily as the number of users grows, and the module handles up to 500 users successfully without failure, showing stable behaviour under increasing concurrency. A similar latency trend is observed in the `Update_Policies`, with latency increasing as the user count grows, however, the module fails completely at 500 users, with no successful transactions recorded. This failure indicates a scalability limitation of this smart contract due to the increased complexity of operations.

In addition, Figure 7.6 presents an analysis of the relationship between the number of users and the failure rate compared to the effect of increasing the TPS rate on system performance for the `Update_Policies` chaincode. The experiment shows that when TPS increases from 100 to 500 while keeping the number of users constant at 200, the system maintains a 100% success rate with zero transaction failures, and both latency and throughput remain stable and nearly flat. This suggests that the system is capable of handling higher TPSs under moderate user load. In contrast, when the number of users increases from 100 to 500 while

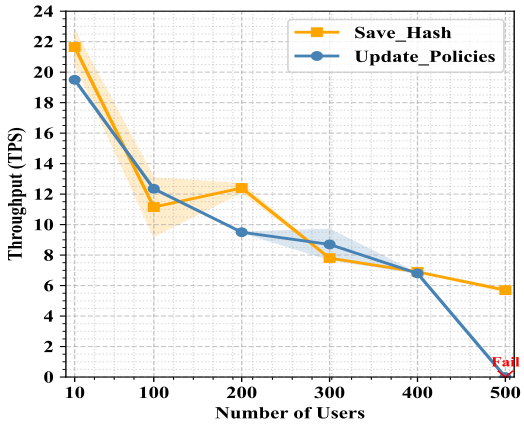


Figure 7.4: **Throughput vs Number of Users** The throughput of the `Save_Hash` and `Update_Policies` modules is shown as the number of users increases from 10 to 500, with a fixed transaction rate of 50 TPS. The shaded regions represent the standard deviation (SD) across experimental runs.

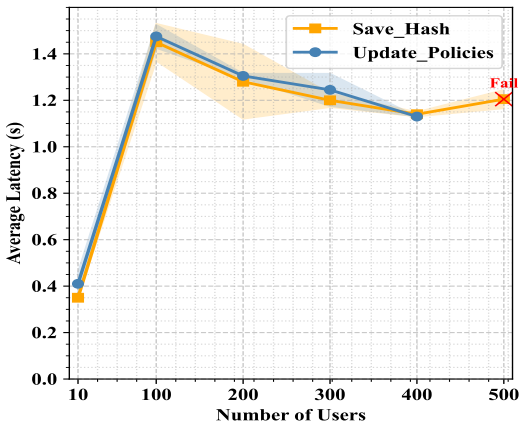


Figure 7.5: **Latency vs Number of Users** The latency of the `Save_Hash` and `Update_Policies` modules is shown as the number of users increases from 10 to 500, with a fixed transaction rate of 50 TPS. The shaded regions represent the standard deviation (SD) across experimental runs.

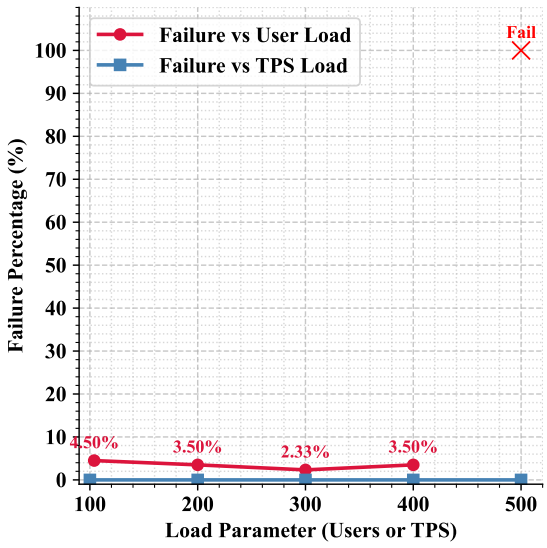


Figure 7.6: **Impact of User Load vs TPS on System Failures** This figure illustrates the contrasting effects of increasing the number of users versus increasing the transaction rate (TPS) on the success and failure of transactions for the `Update_Policies` chaincode.

keeping TPS fixed at 50, the system begins to exhibit failures. At 100 users, the failure rate is minimal, but as the number of users reaches 200 and 300, failures increase to 8 and 14 transactions, respectively. When the load reaches 500 users, the system completely fails, with a failure rate of 100%. This trend indicates that scalability is more sensitive to the number of concurrent users than to the transaction rate, implying that system bottlenecks are likely tied to user-level concurrency (e.g., authentication, endorsement, or identity lookup operations) rather than the transaction rate limits. This observation is critical for system designers, highlighting the importance of optimising user-related components to improve scalability and reliability under high load.

To complement the performance analysis conducted via Hyperledger Caliper, we also employed the K6 load testing tool to evaluate the blockchain layer under various user loads when uploading AV data via the `Save_Hash` chaincode module. The test involved a fixed data size of 3 GB and a constant target TPS of 100. The number of concurrent users ranged from 50 to 500. It is important to note that at this configuration, the system was not able to handle more then 3GB. Further analysis of data uploading tests and comparative discussion in Chapter 8, section 8.1

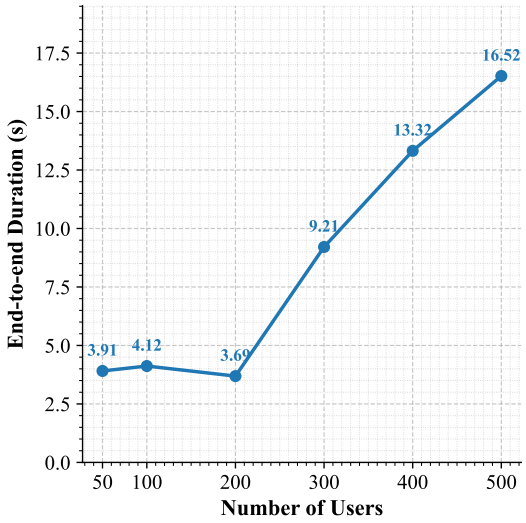


Figure 7.7: End-to-end duration of the `Save_Hash` module during data upload, measured using k6, increases as the number of concurrent users grows.

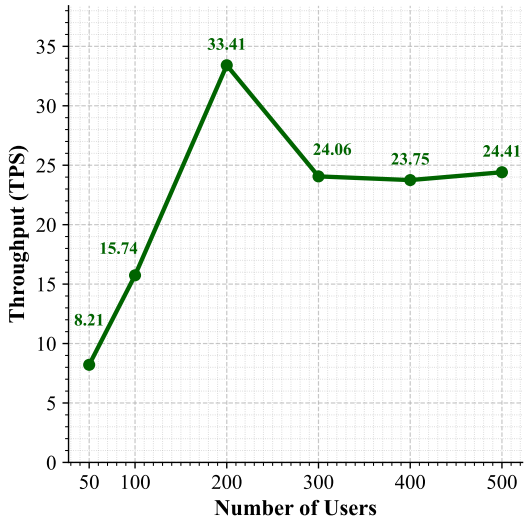


Figure 7.8: Throughput of the `Save_Hash` module during data upload, measured using k6, shows variations under different user loads.

As shown in Figure 7.7, and 7.8, the system demonstrates consistent scalability up to 200 users, after which the duration increases significantly and failure rates begin to emerge. While throughput improves with more users initially, it decreases beyond 300 users due to saturation. This indicates the performance limits of the system in intensive data upload scenarios.

The implementation of weighted voting through repeated identities and logical `AND/OR` expressions introduces additional processing compared to conventional endorsement policies. Although our evaluation showed acceptable performance under moderate and high transaction rates, the added computational overhead was clearly observable when compared with the

baseline configurations presented in Chapters 5 and 6. This overhead could become more noticeable in larger or more complex deployments.

7.6 Conclusion

This chapter introduced a Weighted Endorsement Policy for Hyperledger Fabric as a means to align blockchain governance with the realities of autonomous vehicle ecosystems. The results demonstrate that weighted voting can be integrated without altering Fabric's codebase and that such policies can maintain acceptable performance under realistic conditions. While all tested chaincode modules achieved comparable throughput under moderate loads, their latencies varied, with `Register_user` consistently outperforming `Update_Policies` and `save_hash`. The system showed strong resilience to increases in transaction rate, maintaining consistent performance up to 500 TPS without failures. However, the experiments also indicated that scalability is more constrained by user concurrency, with the failure rate in `Update_Policies` rising significantly beyond 200 concurrent users.

It is important to highlight a consideration about the design and applicability of this EP. While assigning weights enables the system to capture governance structures more realistically, giving a very high weight to a single organisation risks concentrating authority and thereby reducing decentralisation. This trade-off between authority representation and distributed trust needs to be carefully balanced when configuring policies.

Overall, the chapter establishes a foundation for more adaptable, business-aware endorsement models.

Chapter 8

Analysis and Benchmarking

The previous Chapters 5, 6, and 7 present the design, implementation, and evaluation of three different EP setups. Each highlights alternative strategies for introducing trust in a multi-stakeholder setting.

To benchmark the implications of these design decisions, this chapter conducts a comprehensive analysis of the three EP configurations, benchmarking their impact on key performance indicators such as throughput, latency, and transaction success rate.
The findings are then discussed in light of the research objectives and questions, providing insights into both the efficiency of the Fabric-based system for AV data sharing and the extent to which EP design choices influence performance, scalability, and stakeholder trust. Furthermore, to contextualise the results, this chapter includes a comparative analysis with related works in the literature, positioning the proposed approaches within the broader field of blockchain-based data-sharing frameworks.

In summary, this chapter is structured as follows:

Section 8.1 presents the comparative analysis of the system and the three endorsement policy approaches. The key research findings and discussion are described in Section 8.2, which are then linked to the research objectives and questions in Section 8.3. Then section 8.4 presents a comparative analysis with state-of-the-art works. Finally, Section 8.5 provides a summary of this chapter.

8.1 Comparative Analysis of the System Configurations

This section presents a detailed comparison and benchmarking of the system under different endorsement policy configurations, varying experimental conditions, including user load, data volume, and transaction rate. By analysing throughput, latency, and system stability across multiple scenarios, we provide insights into the following.

- Identify the performance limits and scalability characteristics of the system. These performance limits and failure thresholds are configuration-dependent and reflect the

specific experimental setup.

- Trade-offs introduced by different endorsement strategies.

8.1.1 Comparative Evaluation Method

The performance evaluation in the following sections follows a factor-driven testing in which one experimental variable is varied at a time to observe its effect on the behaviour of the system. The three different EP setups are: the Default Policy (Approach 1 explored in Chapter 5), the custom policy (Approach 2 discussed in Chapter 6) and the weighted voting simulation policy (Approach 3 detailed in Chapter 7). Variability is represented using standard deviation (SD), shown either as shaded regions or error bars, depending on the figure. While the results in this chapter help to offer valuable guidance for potential real-world deployments, they reflect prototype behaviour under the tested scenarios and are not statistically generalisable.

8.1.2 Effect of Increasing Number of Users

This section evaluates how varying the number of concurrent users affects the performance of the system under the three endorsement policy configurations. To isolate the effect of user load, other variables, such as transaction type, network configuration, and transaction rate, are kept constant throughout the experiments.

The analysis considers three key performance indicators: throughput, representing the rate at which transactions are successfully processed; latency, capturing the time delay between transaction submission and confirmation; and success rate, measuring the proportion of transactions that complete without error. By progressively increasing the number of users, the experiments reveal how well the system scales, how user concurrency influences responsiveness, and where endorsement policy complexity begins to introduce performance bottlenecks.

Throughput Trends

A central and common observation drawn from the evaluation of the proposed system covering the three Endorsement Policy configurations detailed in Chapters 5, 6, and 7, is the pronounced decline in throughput as the number of concurrent users increases. This trend remains consistent across all tested configurations, regardless of the specific EP applied. Figure 8.1, illustrates the throughput performance of the *Save_Hash* module (measured in transactions per second) in the three different EP setups. The evaluation considers varying numbers of virtual users, ranging from 100 to 500. In that figure, the solid lines represent the mean values from repeated experiments (typically 5 rounds), while the small shaded regions indicate the standard deviation (SD), reflecting the variability of measurements and the stability of each approach.

As user concurrency increases, a general decline in throughput is evident across all three EPs. Among them, Approach 1 consistently demonstrates the highest throughput levels,

particularly under lighter user loads. However, its performance degrades sharply as the number of users increases. In contrast, the Third Approach policy, characterised by its more complex logical structure based on sets AND/OR conditions, yields the lowest throughput across all user scales, primarily due to the additional computational overhead involved in its processing. The Second Approach achieves a more balanced performance; it performs better than the third policy under all conditions and approximates the first policy's throughput at moderate to high user levels, particularly between 300 and 500 users.

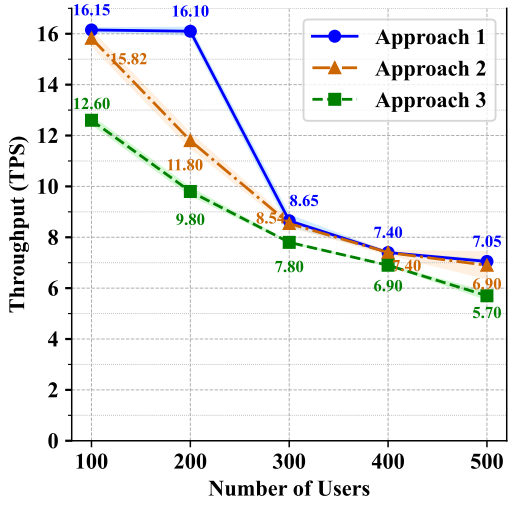


Figure 8.1: **Throughput Performance vs. Number of Users** The figure presents the throughput trends observed under the three endorsement policies for the *Save_Hash* Module.

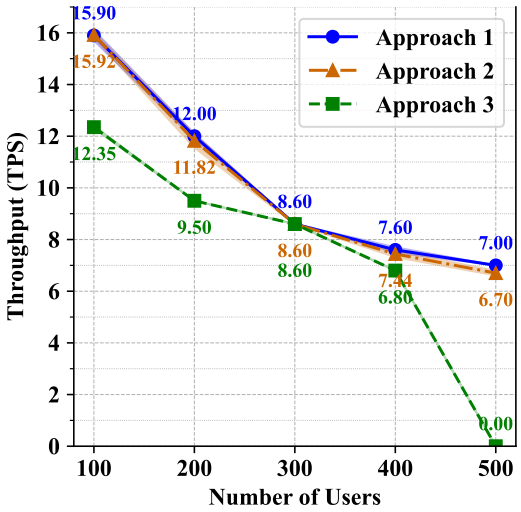


Figure 8.2: **Throughput Performance vs. Number of Users** The figure presents the throughput trends observed under the three endorsement policies for the *Update_Policy* Module.

Another example is for the `Update_Policy` module, which contains read and write operations in Figure 8.2 that presents the throughput performance under the three endorsement policy configurations across the user range of 100 to 500. The figure highlights a consistent decrease in throughput as the number of users increases for all approaches. Approach 1 again achieves the highest throughput in most user loads, while approach 3 shows the lowest throughput values overall, with a significant drop to zero throughput at 500 users, likely due to its more complex policy logic and associated computational costs. Approach 2 offers a compromise between the two, delivering throughput values closer to Approach 1. *These results further confirm that the complexity of the endorsement policy, the user concurrency, and the chaincode design all have direct impacts on system performance* The results emphasie the trade-offs between the expressiveness of the policy and operational efficiency.

Latency Trends

This section examines how the latency of the system responds to increasing numbers of concurrent users under different endorsement policy configurations. As depicted in Figure 8.3, in all endorsement policies, latency trends initially decrease from 100 to 200 users,

reaching their lowest point at 200 users, before beginning to rise again from 300 to 500 users. This behaviour is observed in First and Second EP configurations. For the First Approach, aroundncy reaches 0.90 s at 500 users, suggesting it is most efficient at high load. The Second Approach shows the best performance at lower loads, but experiences a rise beyond 300 users, reaching 0.94s at 500 users. In contrast, Approach 3, while showing an initial latency of about 1.45 seconds for 100 users, which means it is the highest latency average, experiences a steady decline in latency, dropping to approximately 1.21 seconds at 500 users. The Standard Deviation remains relatively small across all user counts, indicating stable performance.

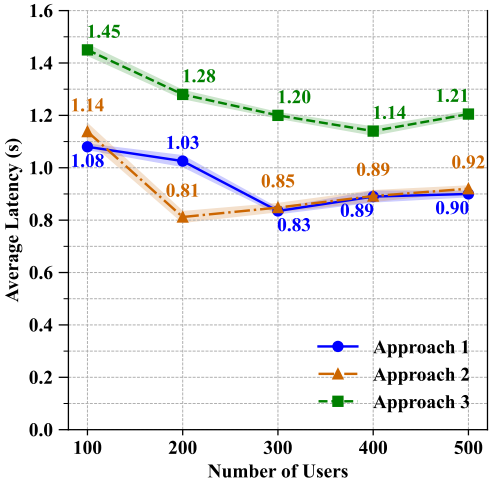


Figure 8.3: **Average Latency vs. Number of Users** The figure displays the average latency under rising user concurrency at a fixed transaction rate of 50 TPS for the *Save_Hash* Module.

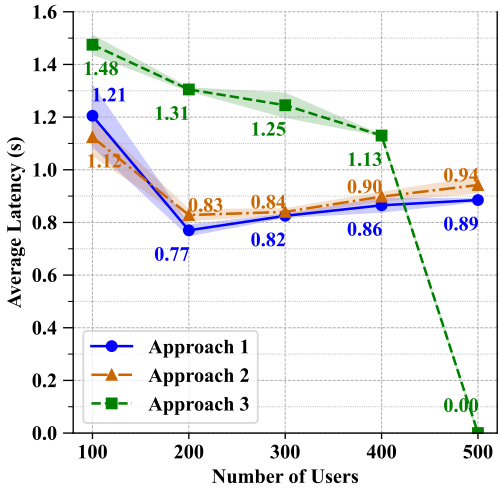


Figure 8.4: **Average Latency vs. Number of Users** The figure displays the average latency under rising user concurrency at a fixed transaction rate of 50 TPS for the *Update_Policy* Module.

This small but noticeable latency decrease occurs in all approaches, although its duration differs, which motivated us to additionally test with only 10 users. In this case, latency was observed to be very low for all three approaches: 0.29, 0.31, and 0.31 seconds, respectively. This trend can be attributed to the batching behaviour of the ordering service. For example, in the First Approach with 100 users, batches are not always filled to the configured BatchSize.MaxMessageCount of 10, causing transactions to wait closer to the BatchTimeout of 2 seconds before being cut [152].

At 200 users, although the number of submitted transactions increases, system throughput drops sharply as shown in Figure 8.1. Consequently, fewer transactions are successfully processed, which prevents long queuing delays from building up and stabilises latency at higher user counts [153]. Beyond this point, the subsequent increase in latency becomes consistent with the observed throughput and user loads.

The points at which latency starts to increase differ among approaches, depending on the interaction of EP complexity and batching. Simple EPs lead to faster per-transaction endorsement, allowing the system to sustain higher user loads before queuing dominates.

In contrast, complex EPs require slower per-transaction endorsement, resulting in initially high latency; however, faster batch filling at moderate loads can temporarily reduce average latency. Thus, the turning point where latency begins to increase depends on how quickly the system saturates the orderer and peer endorsement processes, as well as the queuing behaviour under each EP [153].

Figure 8.4 presents the average latency of three different approaches as the number of users increases for the *Update_Policy* Module. Approach 1 demonstrates a consistent and low average latency, beginning at approximately 1 second for 100 users and stabilising around 0.90 seconds as the user count reaches 500. The standard deviation remains relatively small across all user counts, indicating stable performance. Approach 2 maintains a consistently low average latency, between 1.14 seconds to 0.92 seconds. In contrast, the Third approach maintains the highest latency throughout, reflecting the computational overhead of its complex endorsement logic.

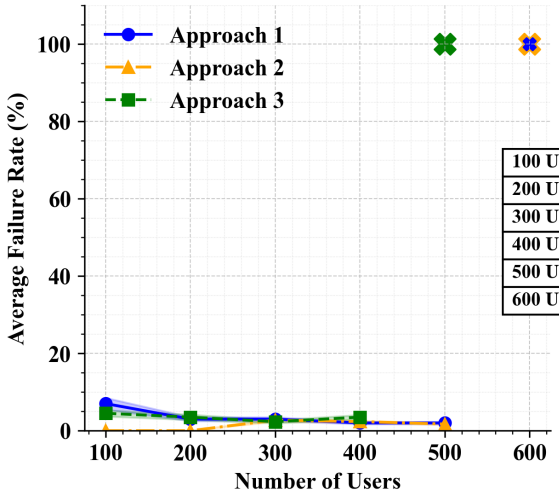
As illustrated in both figures, the observation aligns with the finding of Melo et al. [141], which states that implementing more sophisticated endorsement policies, like K-out-of-N schemes (e.g., 2-out-of-3), enhances latency stability by allowing operations to continue smoothly even if some endorsing peers are unavailable, thereby maintaining system responsiveness with minimal additional delay. Overall, all three approaches show relatively stable latency as the number of users grows.

The main observation in this section is that *even though the throughput decreases and is impacted by the number of users and complexity of the EP and module, the average latency is relatively stable.*

Success Rate

In addition to throughput and latency, another critical aspect of evaluation is the transaction success rate and failure behaviour under stress. Building on our earlier chapters, which focused on the *Update_Policies* as a module that involves both read and write operations, and subsequently faced some failures, Figure 8.5 shows the failure rates for the three EP configurations that have been observed while testing this module. Approach 1 experienced failures from low user loads, but maintained a relatively low overall failure rate. At approximately 500 users, the failure rate jumps to 100% under the tested configuration, indicating a complete loss of system stability. Approach 3 demonstrates a similar pattern, but reaches its critical threshold earlier, at just 400 users, where its failure rate also spikes to 100%. In contrast, Approach 2 performs significantly better, exhibiting the highest resilience to user load. It starts with a failure rate of 0% for both 100 and 200 users. Although it does experience some failures later, its failure rate remains consistently low. Although Approaches 1 and 3 fail completely beyond their respective thresholds, Approach 2 continues to operate with a low, manageable failure rate, until 600 users, where it reaches 100%. This finding underscores the superior reliability of Approach 2, making it the most robust policy under high user loads.

This indicates *that different policies demonstrate distinct thresholds and resilience levels.*



Failure Rates (%)

	Approach 1	Approach 2	Approach 3
100 Users	7%	0%	4.50%
200 Users	3%	0%	3.49%
300 Users	3%	2.73%	2.33%
400 Users	2%	2.35%	3.50%
500 Users	2%	1.72%	100%
600 Users	100%	100%	100%

Figure 8.5: **Failure Rate vs. User Load Across Three Endorsement Policies at 50 TPS.** This figure presents the failure rates for the three EPs configurations using distinct markers: circles, squares, and triangles, respectively. Colored "X" markers highlight the critical thresholds at which the failure rate abruptly reaches 100%, indicating the loss of system stability beyond these user levels.

8.1.3 Effect of Increasing TPS Rate

This section investigates how increasing the target transaction rate affects system throughput and latency. The analysis highlights whether higher submission rates translate into higher achieved throughput, or whether bottlenecks limit performance. By systematically varying TPS levels, the analysis provides insights into the operational limits of the framework and identifies thresholds beyond which performance gains decline. These findings help characterise the scalability of the system and inform practical recommendations for configuring transaction loads in real-world autonomous vehicle data-sharing scenarios.

Throughput Trends

When considering variations in transaction rate (TPS), the key observation is that the impact on performance remains comparatively minor across the three EPs. Figure 8.6 illustrates that in all three approaches, the throughput is not directly proportional to the target TPS. While the target TPS represents the desired submission rate, the system's processing capacity imposes a clear ceiling on achievable throughput. In Approach 1, as the target TPS increases from 100 to 500, the throughput fluctuates slightly but remains consistently around 12 TPS. The system cannot process transactions at a rate faster than this, regardless of the target. Similarly, in Approach 2, the throughput is consistently around 11.6-11.7 TPS. In addition, despite the target TPS being raised, the throughput remains stable, indicating a bottleneck in the system. In Approach 3, the throughput is consistently around 9.5 TPS. Similar to the other approaches, the system reaches its maximum processing capacity and does not scale with the target TPS. Thus, the results reveal that *although the impact of the EP configuration is evidenced, increasing the target TPS has a limited impact on the actual performance of the*

system.

Latency Trends

Figure 8.7 illustrates the average latency in three approaches as the target TPS rate increases from 100 to 500. Approach 1 consistently demonstrates the lowest average latency, remaining remarkably stable between approximately 0.76 and 0.82 seconds, indicating its robust performance and scalability under varying load conditions. Approach 2 exhibits a slightly higher but similarly stable average latency, hovering around 0.84 to 0.86 seconds, suggesting a good balance of performance. In contrast, Approach 3 consistently records the highest average latency, ranging from approximately 1.26 to 1.31 seconds, and shows a slight increasing trend with rising TPS, implying that its design is less efficient in handling increased transactional loads compared to the other two approaches. Overall, the data clearly identify Approach 1 as the most efficient in terms of maintaining low latency, followed closely by Approach 2, while Approach 3 presents significant latency challenges across the evaluated TPS range. While the EPs vary in their latency, they are stable, and this stable latency is indeed a consequence of reaching a throughput ceiling.

This persistent differentiation in stable latency levels, even as the system reaches its maximum throughput capacity, underscores the fundamental differences in the underlying architecture and efficiency of each approach.

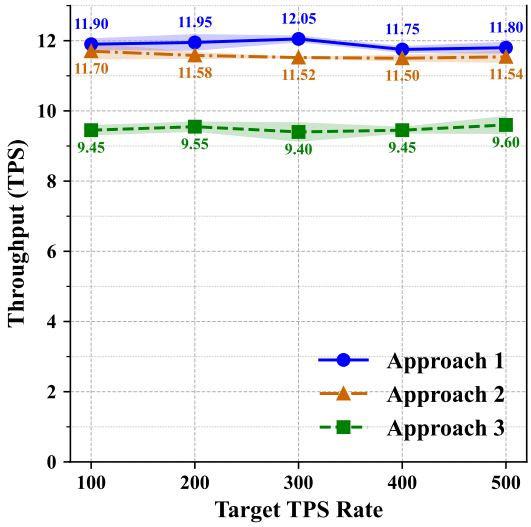


Figure 8.6: **Throughput vs. Target TPS** This figure compares the throughput achieved by testing `Update_Policy` module under the three endorsement policies with a 200-user load and varying TPS rate.

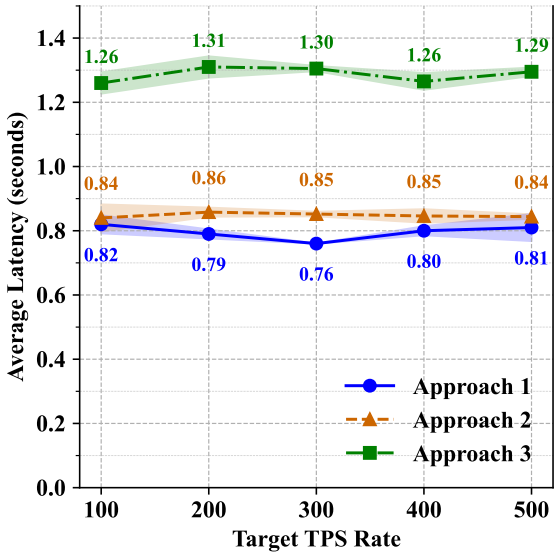


Figure 8.7: **Average Latency vs. Target TPS Rate.** Average latency performance of the three endorsement policies under increasing TPS rates with a fixed user load of 200 users for `Save_hash` function.

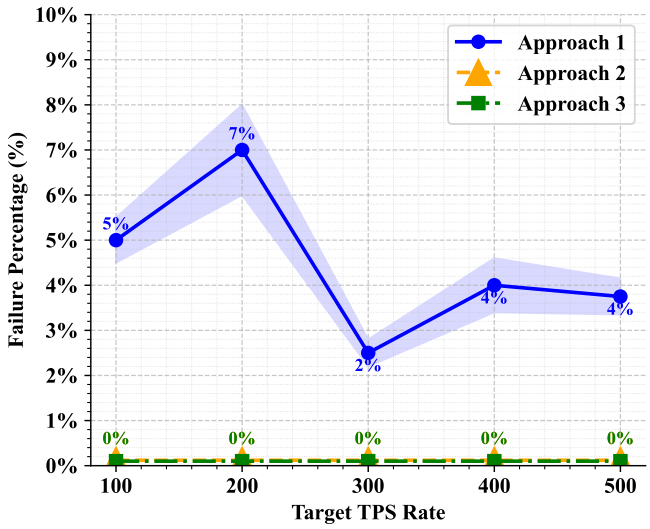


Figure 8.8: **Transaction Failure Rates Under Varying TPS Levels.** Comparison of failure rates for different EP configurations as TPS increases.

Success Rate

As illustrated in Figure 8.8, under increasing TPS conditions, the first EP begins to show a gradual rise in the percentage of failed transactions starting from 5% at 50 and 100 TPS, and increasing to 7% at 200. This indicates that while the system can handle moderate transaction loads, its reliability begins to degrade under higher stress levels. TPS. In contrast, the second and third approaches maintained a flawless 0% failure rate throughout all TPS levels tested, demonstrating their superior resilience and robustness even under sustained high transaction pressure. These observations highlight that maximum throughput and stable latency alone do not guarantee system reliability; failure rates can still increase when the transaction rate exceeds the system's effective processing capacity.

8.1.4 Effect of Increasing Data Size

This section examines how increasing data size affects system performance, focusing on throughput and end-to-end latency as measured using K6. As observed in previous chapters, larger loads increase processing and transmission demands, which in turn affect system efficiency. This section presents a comparison of these impacts under different endorsement policies. These results provide insight into the system's scalability and the trade-offs between data volume, latency, and reliability, which are critical for autonomous vehicle applications that rely on high-volume data.

Throughput Trends

Figure 8.9 illustrates the performance of the system with a fixed 200 user load of the `save_hash` module under Approach 2 configuration, using the k6 tool, as a function of the data size,

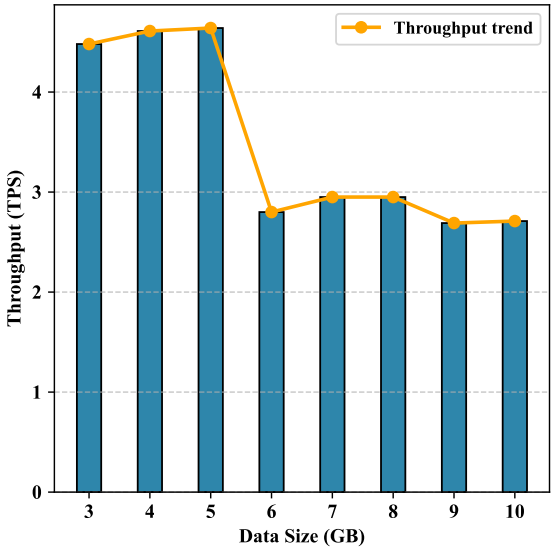


Figure 8.9: **Impact of Data Size on Throughput.** This figure illustrates how the average throughput varies as the input data size increases from 3 GB to 10 GB under Approach 2.

measured in gigabytes (GB). It reveals a clear performance trend, in which the throughput initially increases with data size, peaking at 5 GB. Throughput initially increases with data size, peaking at 5 GB, suggesting optimal utilisation of system resources. However, a significant drop in throughput occurs as the data size increases from 5 GB to 6 GB, after which the throughput stabilises at a lower level between 6 GB and 10 GB. This sharp decline and subsequent plateau suggest that the system reaches a performance limitation. It is important to mention that this approach is the only one that achieves 10 GB; however, we had to reduce the request rate to only 5-10 transactions per second.

For a more in-depth comparative evaluation, comparison tests were conducted to assess the performance of two endorsement policy configurations. Figure 8.10 compares the performance of the first and second configurations when uploading a data set totalling 3 GB, with user loads ranging from 200 to 500. At higher user counts, the first EP achieves greater throughput, largely due to its simplified endorsement logic, which reduces the processing overhead. This highlights a performance trade-off: while the default EP offers higher raw throughput, it does so at the cost of decreased flexibility and robustness. On the other hand, the Second EP handles complexity better and offers a more resilient operational profile at a marginal performance cost. The first approach achieves higher throughput at larger user counts, likely due to its reduced processing complexity.

Overall, these results illustrate that increasing data size imposes measurable performance constraints, and the choice of EP significantly influences throughput under high-volume conditions. Simplified EPs may improve raw throughput but can compromise robustness, whereas more sophisticated EPs handle larger loads more reliably, making them better suited for critical AV data-sharing scenarios where both performance and reliability are essential.

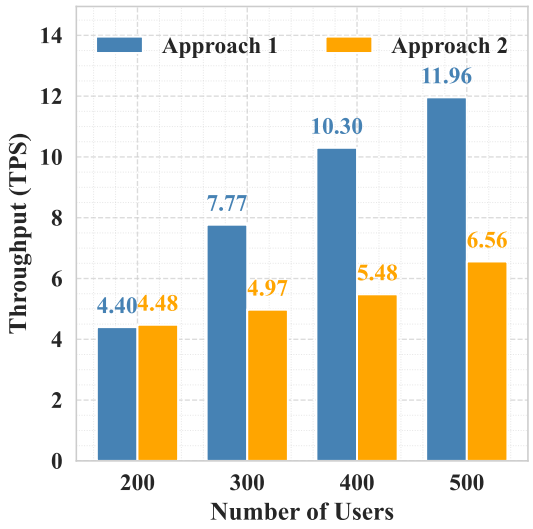


Figure 8.10: **Throughput Comparison Between First and Second EPs During a 3 GB Upload.** This chart compares the throughput under increasing user loads ranging from 200 to 500 users, while transferring a total of 3 GB of data.

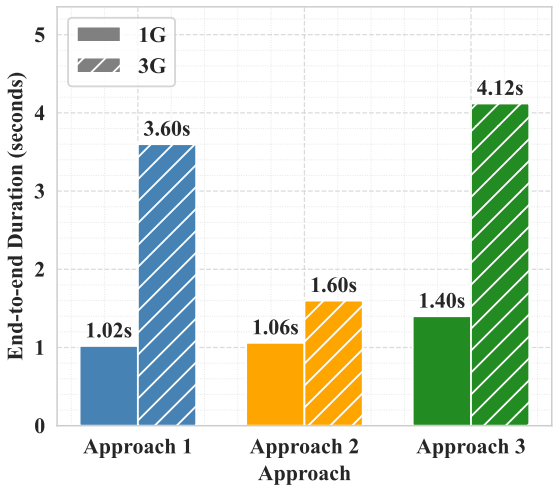


Figure 8.11: **Duration Time vs Data Size.** This figure presents a comparison between the three EP approaches, showing the execution time measured under a workload of 100 users and data sizes of 1 GB and 3 GB.

Latency Trends

Figure 8.11 shows that the system execution time increases with the data workload. For all approaches, the 3GB workload (hatched bars) generally requires more time than the 1G workload (solid bars), indicating that larger data volumes lead to higher processing durations. Although the data size difference between 1 GB and 3 GB is relatively modest, it still produces a noticeable impact on execution time, particularly for Approaches 1 and 3, whereas Approach 2 exhibits a smaller increase. It is important to note that these tests were conducted using K6, measuring end-to-end performance, including data uploading, which reflects real-world system behaviour.

Furthermore, that figure indicates that the choice of endorsement policy approach also impacts execution time. Approach 1 and Approach 3 generally take longer for higher workloads, while Approach 2 shows smaller increases. This reflects the fact that EP complexity directly influences how processing and network resources are utilized: policies requiring more endorsers or higher-authority approvals increase transaction processing time. These trends have important implications for system design. Execution time sensitivity to workload size highlights *the need to optimise both EP logic and system architecture to maintain consistent performance under high-volume, data-intensive conditions.*

8.2 Discussion

It is important to emphasise that the identified failure points and performance ceilings are configuration-dependent. Factors such as the number and type of orderer nodes, peer resource allocation, database choice (e.g., CouchDB vs LevelDB), network conditions, and benchmarking parameters (e.g., Caliper workload generation) all influence when and how system degradation occurs. Consequently, the failure thresholds reported in this chapter should not be interpreted as fixed or inherent scalability limits of HLF, but rather as indicative of system behaviour under the specific experimental conditions considered in this work.
The system is capable of handling a moderate range of users, data volume, and transaction loads effectively. However, performance degradation under high loads indicates that further optimisation or resource scaling may be required for extreme scenarios. In addition, failures occurred when the system was subjected to extreme combinations of users and transaction rates, highlighting the current limits of scalability. These failures were influenced not only by high load but also by the complexity of the chaincode, which increases the processing overhead and affects transaction throughput under stress. .

In addition, its performance varies with the strictness of the endorsement policies, indicating that the system responds reliably to configuration changes. During testing, the system demonstrated a clear relationship between endorsement policies and overall performance. While the modified endorsement policies increase flexibility, overly relaxed configurations could potentially increases centralisation. Conversely, highly complex endorsement policies impose additional communication and validation overheads that reduce transaction throughput, especially under higher workloads. These observations provide an essential context for evaluating the effectiveness of different policy configurations and their impact on system behaviour.

The current system architecture, regardless of endorsement policy, fails to scale with increasing target load. As we have seen, when the target TPS is increased, the send rate often plateaus at a level similar to the observed throughput, or even slightly decreases in some scenarios. As illustrated in Chapter 5 (Figure 5.8), and in Chapter 6 (Figure 6.5), which combined in this chapter in in Figure 8.6, across all modules, the absence of a rising throughput trend indicates that the system is not scaling with increased transaction load, and that the limiting factors are internal system bottlenecks rather than endorsement logic. This underscores the fact that scalability challenges are systemic and cannot be resolved solely by altering endorsement configurations.

An illustrative example of this bottleneck behaviour, in Approach 2, among all chaincode modules, although the target TPS increased from 50 to 500 (with 200 users), the performance remained remarkably stable. For example, `Save_hash` module, hovering around 11-12 TPS. This indicates a ceiling on the system's processing capacity, which is independent of the attempted transaction rate from the client side. The stability of the throughput suggests that once this ceiling is reached, any additional load simply results in increased latency rather than a proportional increase in transaction processing.

To better understand this observation, the send rate reported by Caliper was analysed.

In most cases, the send rate was very close to the achieved throughput, according to the results represented in Figure 6.7 suggesting that the benchmark tool itself did not throttle or saturate before the system. Instead, the system was inherently unable to process more transactions than a relatively low ceiling, regardless of the load generated. This close correlation implies that this stable or declining throughput pattern can be explained by the presence of system-level bottlenecks. Since the endorsement policies varied across the three approaches, but the performance ceiling remained similar, endorsement logic itself is not the primary constraint. Instead, several architectural factors likely contribute to this bottleneck:

- **Peer Resource Constraints.** Different hardware configurations could have a clear impact on system performance [100]. Accordingly, several significant software and hardware upgrades have been implemented during the first approach to address initial challenges related to storage and resource constraints. These upgrades include:
  - Increased Storage Capacity: Upgrading the AWS EBS volume to accommodate larger Docker images and expanded ledger data:
    1. EC2 Instance specs
    2. Instance type: t3a.2xlarge, cpu cores: 8, 32 GB RAM
    3. OS: Ubuntu
    4. Server storage: 350 GB
  - Enhanced Compute Power: Improving vCPU and RAM allocations to optimise the performance of Docker containers, particularly under high load.
  - Node.js Version Upgrade: Updating to Node.js v18+ to ensure better compatibility with the Fabric SDK and overall application performance.

  However, the fact that these foundational improvements did not yield a proportional increase in transactional throughput suggests that the bottlenecks are more complex than resource shortages.

- **Impact of System and Network Configuration**
  - *Orderer:* A single orderer node could potentially represent a significant point of contention and a bottleneck for the entire network's throughput. Regardless of the processing power or number of peer nodes available for endorsing and committing transactions, all transactions typically pass through the orderer service for global ordering. This could contribute to a bottleneck at the ordering stage, where the single orderer's capacity to process and broadcast transactions might become a limiting factor for the entire system's throughput.
  - *CouchDB:* CouchDB as the state database can introduce performance overhead, making the validation and commit phases slower compared to LevelDB, particularly for read-intensive workloads or complex queries. This is because CouchDB, being a document-oriented database, might have different underlying performance characteristics for key-value operations compared to LevelDB [154].
  - *Off-chain storage:* Using IPFS as off-chain storage affects system latency and throughput in different ways, depending on the uploading and retrieval processes. When a transaction involves IPFS, the system first uploads the data, generating a content identifier (CID), which is then stored on the blockchain. Later, the user can

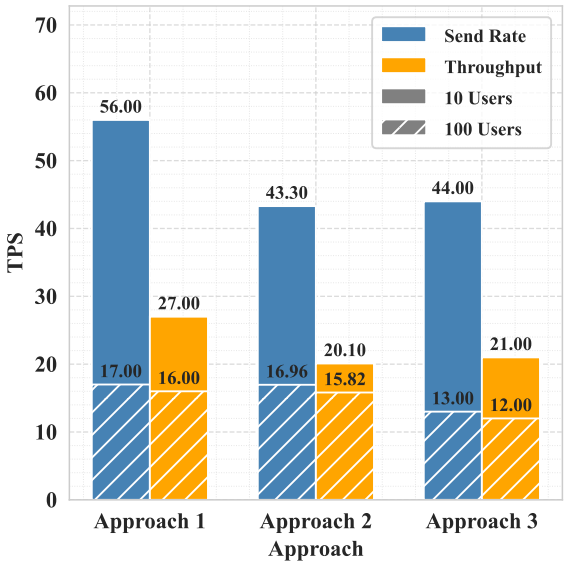


Figure 8.12: Comparison of send rate and throughput for 10 and 100 users across different approaches. Solid bars represent 10 users, while hatched bars represent 100 users.

retrieve the data using this CID from IPFS. Using a pinning service such as Pinata in the proposed system improves both processes by ensuring persistent storage, optimising file chunking and distribution, and providing reliable infrastructure that reduces variability in upload and retrieval times. Nevertheless, even with Pinata, IPFS can still introduce delays due to file chunking, propagation across nodes, and network-dependent variability, which increases end-to-end execution duration and can slightly reduce throughput, particularly for larger data workloads.

- **Impact of User Load on Throughput and Send Rate**

  Further analysis, particularly when the number of concurrent users was reduced, provides more evidence for internal bottlenecks. For instance, decreasing the number of users from 100/200 to 10 users led to a notable increase in both send rate and throughput across all the tested approaches. For example, in the Figure 8.12 for Approach 1 *save_hash*, 10 users, and 100 TPS target, the Send Rate was 56 TPS and Throughput of 27 TPS, significantly higher than the 17 TPS Send Rate and 16 TPS Throughput observed with 100 users. Even for the most complex tested approach, Approach 3, with the same module and scenario (10 users and 100 TPS target); Send Rate of 40-48 TPS and Throughput of 20-22 TPS, also higher than the 13 TPS Send Rate and 12 TPS Throughput with 100 users.

  While the throughput with fewer users is still well below the target TPS, the increase demonstrates that contention points scale with the number of concurrent client connections or transactions. This suggests that components such as the peer nodes' processing capabilities, the orderer's queuing mechanism, database I/O, or network overhead associated with managing a larger number of active transactions simultaneously are likely responsible for these limitations. Reducing the number of users alleviates some of this contention, allowing individual transactions to progress more efficiently through the system, thereby resulting in higher overall throughput for a given user count.

Analysis of Failures and Anomalies

Different types of errors were observed during load testing in chaincode modules. These errors are not the same as the "failed transactions" shown in the Caliper performance report. These are transactions that were submitted but not successfully endorsed, ordered, or committed according to Fabric's normal lifecycle. They directly reduce measured throughput and success rate in the benchmark results. Table 8.1 summarises the main runtime errors observed during stress testing of different chaincode modules. Each error type highlights a specific layer of system stress rather than general misconfiguration. Transaction timeouts is a classic symptom of network and processing congestion. When peers or orderers are overwhelmed, they can't process transactions within the configured timeout period. This is a common bottleneck during stress testing [152]. The ERR_IPC_CHANNEL_CLOSED error reflected Caliper worker thread failures due to excessive inter-process communication load [155]. Connection deadline failures indicated that peers were unable to respond in time, often because of container overload, network latency, or handshake delays. This points to resource issues at the network and peer level. It implies the peer node itself is too busy (due to high CPU/memory usage or network saturation) even to establish or maintain a connection, which is a severe bottleneck. Finally, discovery service failures revealed Fabric's inability to return peer information quickly enough under heavy load. If it's slow to respond under load, it shows that the network's foundational services are strained, preventing new transactions from even being submitted correctly.

Together, these runtime errors provide insight into where the Fabric network experiences bottlenecks when subjected to large-scale concurrent transactions.

Key Finding and Recommendations

The following list summarises the key findings from the evaluation of the system and provides recommendations based on the observed performance, scalability, and trade-offs. The aim is to highlight the most important insights gained from testing and to offer guidance to optimise system configuration and future improvements. These findings provide actionable insights for practitioners and researchers to improve performance, tailor deployment strategies, and anticipate system limitations.

1. *User concurrency is a primary limiting factor for overall system performance, regardless of the specific policy in place.* As shown previously in Figure 8.5, Approach 1 fails at approximately 500 users under the tested configuration, Approach 2 degrades gracefully up to 600 users, and Approach 3 fails earlier (400 users). The throughput decreases as user load increases, while some chaincode modules fail completely at a certain concurrency level. Thus, real-world systems should be stress-tested to identify critical thresholds. Practical deployments must therefore incorporate concurrency testing into acceptance procedures.

2. *The complexity of the chaincode module has an impact on the system performance.* The `Update_Policy` module, which supports read and write operations, has the highest latency, lowest throughput, and lowest success rate as in 6.9, 7.4, 7.5. This highlights

the need to optimise complex modules or isolate the read and write operations to avoid bottlenecks.

3. *The complexity of an endorsement policy has a direct impact on system performance.* Approach 1 (Default) is faster but collapses under high load; Approach 3 is slower due to complexity; Approach 2 balances the two. As in Figures 8.1, 8.28.3, and 8.4.

   This is further supported by the early occurrence of failure under increasing concurrency, as highlighted in point 1. The results suggest that endorsement policy selection should always be workload-aware.

4. *The system can maintain responsiveness even when it is not processing transactions at its maximum possible rate.* The analysis shows that the system latency can remain stable as user concurrency and target TPS increase, even while throughput is declining or has reached its ceiling. Latency is stable across concurrency and TPS scaling, but sensitive to policy complexity, such as in 8.3, 8.4, and 8.7.

5. *Simpler policies tend to have lower latency, while more complex policies have higher latency.* This trade-off reinforces the importance of aligning policy design with expected system load.

6. *The system has a performance limitation related to processing large data volumes, which must be considered when designing systems for data-intensive workloads.* Figure 8.9, 8.10, and 8.11.

7. *Align Policy Choice with Deployment Priorities* from the previous evaluation, Default EP is the best for light, high-speed environments. While Approach 2 is ideal for high concurrency and reliability. Approach 3 EP is useful when expressive policies are needed and performance is secondary. A hybrid or adaptive approach may further improve resilience and balance performance with flexibility.

8.3 Linking Findings to Research Objectives and Questions

This section connects the findings from the evaluation to the research objectives and questions outlined at the beginning of the study. In addition, it demonstrates how the proposed framework addresses the identified gaps in literature and practice. Each research question is revisited in light of the experimental results, allowing a clear mapping between the objectives, the methodological choices, and the outcomes. This structured linkage ensures that the contributions of the research are explicitly aligned with its initial aims, while also highlighting areas where limitations remain and future work may be directed.

RQ1: What are the key components and design principles for developing a secure and scalable multi-party data-sharing framework for autonomous vehicles using blockchain?

As described in Chapter 1, to address this question, four sub-questions were considered:

Table 8.1: Observed Runtime Errors under Stress Testing.

Error Type	Seen In	Meaning
`Transaction Timeout`	- `Save_hash`: from 600 to 1000 users at low TPS.	Peer/orderer too slow to endorse or commit due to concurrency and queuing delays.
`ERR_IPC_CHANNEL_CLOSED`	- `Update_Policies`: 200 users at 600 to 1000 TPS	Caliper worker threads fail due to IPC overload from too many parallel processes.
`Failed to connect before deadline`	- `Save_hash`: 200 users, TPS 600–5000 - `Update_Policies`: 500 users	Peer nodes unresponsive; may result from container overload, Docker network latency, or TLS/gRPC handshake timeouts.
`DiscoveryService failed to return results`	- `Update_Policies`: 500 users at 200 to 500 TPS	The Discovery service cannot return peer information quickly enough under load, reflecting Fabric-level stress.

(a) What are the components, tools, mechanisms, and policies required for the proposed solution?

The framework incorporates the following building blocks:

- **Blockchain network:** A permissioned Hyperledger Fabric infrastructure enabling secure and auditable interactions between stakeholders.
- **Smart contracts:** Chaincode modules such as `Save_Hash` and `Update_Policies` enforce secure data registration and policy management.
- **Access control and policies:** ABAC were implemented for secure data sharing among stakeholders.
- **Configurable endorsement policies** : EPs were designed and evaluated, demonstrating their role in ensuring trust, resilience, and security while adapting to the use case collaboration requirements.
- **Off-chain storage:** The use of IPFS enables scalable handling of large AV-related data, while only metadata (hashes) is maintained on-chain to reduce storage overhead and latency.

Chapters 3 and 4 presented the technical design and implementation of the system. In addition, Chapters 5 to 7 presented the technical design and implementation of each EP in the context of the case study, highlighting how design choices influence system performance.

(b) Which blockchain platform is most suitable for the proposed solution, and why?

Hyperledger Fabric was selected as the underlying platform. Its modular architecture, permissioned model, and built-in support for configurable endorsement policies make it particularly suitable for a multi-party AV environment. Unlike public blockchains, Fabric provides identity management and fine-grained access control, aligning with the security and trust requirements of the use case. Chapter 2 Chapter 3 justifies this selection

(c) What are the types, sources, and flow of the data within the system?

The framework handles two categories of data:

- **On-chain data:** Small metadata and cryptographic hashes, stored within the blockchain for auditability and integrity verification.
- **Off-chain data:** Large datasets (up to 10 GB) relevant to AV operations, stored in IPFS but linked to on-chain metadata for integrity checking.

The data flow involves clients submitting transactions through chaincode modules, which either register new data hashes, update policies, or query existing records. Chapters 3 outlined the system design, data flow and interactions.

(d) How can the proposed framework be implemented to ensure integration of the identified components and fulfil the design requirements?

The implementation of the proposed framework, including smart contract modules, configurable endorsement policies, and hybrid on-chain/off-chain storage, demonstrates how the identified components are integrated into a coherent and functional system. Chapters 4 to 7 show the step-by-step system realisation, including the design decisions, workflows, and performance evaluations that ensure the framework meets its intended objectives.

Overall, RQ1 was answered by developing, implementing, and experimentally evaluating a secure and scalable blockchain-based data-sharing framework tailored for AVs. The design principles, permissioned access, modular smart contracts, configurable policies, and hybrid on-chain/off-chain storage, emerged as the key enablers of both trust and scalability in the system.

RQ2: How does the proposed blockchain-based multi-party framework perform in terms of scalability and efficiency when applied to autonomous vehicle data sharing?

This research question is addressed through the results of the scalability and efficiency experiments, which are experimentally represented in Chapterrs 5 , 6 6, and 7 7 and thoroughly analysed in Sections 8.1.2, 8.1.3, and 8.1.4. These analyses examine the effect of increasing

the number of users, the impact of higher target TPS rates, and the influence of larger data sizes on system performance. The evaluation shows that while the framework maintains stable latency across scenarios, throughput is more sensitive to chaincode operation complexity, endorsement policy design, and the combined user/data load. In addition, among the tested EP approaches, Approach 2 demonstrates the highest resilience under heavy or complex workloads, whereas Approach 1 consistently delivers superior transaction speed under lighter operational conditions.

RQ3: How usable is the proposed blockchain-based data-sharing system from a user perspective?

While the main technical evaluations focused on scalability and efficiency, the System Usability Scale (SUS) test provided a complementary perspective. The SUS results explained in Section 5.5 reflect user perceptions of ease of use, clarity of workflow, and confidence in the system. High SUS scores indicate that, despite differences in performance under varying loads, stakeholders found the system design intuitive and practical for real-world application.

8.4 Comparative Analysis with Related Work

In Hyperledger Fabric, the endorsement policy plays a vital role in the trust and validation framework of the platform. This mechanism defines which peers must endorse a transaction before it can be committed to the ledger. Understanding these policies is crucial, as they directly influence both the security guarantees and operational efficiency of the network. Consequently, several studies have explored the design and optimisation of these policies to enhance security and performance. For example, Androulaki et al. [156] proposed state-based endorsement, enabling fine-grained control by assigning specific policies to different smart contract state variables. To enhance privacy, Dharani et al. [157] introduced a cryptographic approach to conceal both endorser identities and policy definitions, illustrating the potential of EP customisation to balance trust requirements and confidentiality.

Several works have analysed EP strategies across various domains. Soelman et al. [158] examined EPs in supply chain environments, balancing data confidentiality and integrity through multi-tenancy and notary-based designs. Similarly, Mazumdar et al. [159] proposed a threshold-based EP using ring signatures to preserve endorser anonymity, demonstrating how domain-specific requirements can influence policy design. Such studies underscore the importance of tailoring endorsement rules to the operational and governance needs of different industries.

In addition to these domain-specific strategies, performance optimisation has been another major focus, as EP configurations can significantly affect transaction latency and throughput. In terms of performance optimisation, Kwon et al. [160] differentiated between read and write transactions during endorsement. Although this approach achieved up to 60% performance gains for read-dominant applications, it does not provide sufficient guarantees for domains that require high security or highly sensitive data. Yu et al. [161] developed a fuzzy logic–based selection mechanism that dynamically assigns endorsers according to system load. Meanwhile, Lotfimahyari et al. [162] introduced redundant endorsements using spatial diversity to improve reliability. In another approach, Lee et al. [163] proposed allowing endorsers to send

responses directly to the orderer to reduce latency. This trade-off is potentially acceptable in latency-sensitive environments such as IoT; however, it might affect consistency/integrity guarantees of the ledger. Collectively, these approaches reveal how performance-driven EP optimisations must be carefully designed to avoid undermining trust guarantees. The issue of endorser availability and adaptive trust has also been explored. Shimosawa et al. [164] addressed the scarcity of endorsers in private data contexts using BCVerifier, which enables validation even without complete endorsements. Rouhani et al. [165] proposed an adaptive validation model that adjusts the number of validators based on the trust level assessed in the data assets. Furthermore, Thakkar et al. [124] identified inefficiencies in state validation and endorsement processes and recommended improvements through batching and parallel processing. These works highlight the growing need for flexible and resilient endorsement strategies, particularly in dynamic or resource-constrained environments.

Comparative studies have assessed the impact of different EP configurations. Mishra et al. [143] and Harris et al. [166] examined logical combinations such as `OR` versus `AND`, showing that less restrictive policies can reduce latency and boost throughput. Wang et al. [167] supported these findings by identifying the increased computational overhead associated with `AND` policies. These studies show that endorsement strictness directly influences system scalability, making the selection of policies a balance between performance and security.

While these contributions collectively advance the understanding of EP design, there remains a gap in exploring how endorsement policies can be systematically customised to reflect governance structures and real-world trust distributions in safety-critical domains such as autonomous vehicles (AVs). Most prior work has focused on either performance optimisation or domain-specific privacy, whereas the integration of domain governance models into EP design has received limited attention.

In this research, we propose both domain-specific strategies for EP and a comparative study to provide insights that contribute both theoretically and technically in the domain of Hyperledger Fabric for data sharing. Specifically, our work investigates how assigning differentiated endorsement authority, such as privileging manufacturers in AV ecosystems, impacts performance.

8.4.1 Benchmarking

To situate the findings of this study within the broader research landscape, this section presents a comparative analysis between the performance results of our proposed system and those reported in the literature. Table 8.2 situates this work within previous research by comparing it with other HLF-based frameworks in terms of performance, scalability, and system configuration. Compared to previous studies, our framework is evaluated in a larger and more diverse setting: 5 organisations, 5 peers, 500 TPS rate, 500 users and up to 10 GB (in some configurations), while also incorporating explicit endorsement policies, which are often overlooked in other evaluations. These choices make the evaluation more representative of real-world scenarios and enable a more comprehensive understanding of the system's behaviour under different loads and data sizes.

The figure 8.13 illustrates a performance benchmark comparing the average latencies of

Table 8.2: HLF-Based Frameworks: Configuration and Performance

Framework	Perf. Test	API Test	Orgs	Peers	Users	Data Size	EP	TPS	Consensus
[100]	Yes	—	2	4	5000	—	No	900	Raft
[44]	Yes	—	2	2	50	10 MB	No	—	Raft
[93]	Yes	—	—	—	320	—	No	—	—
[95]	—	—	2	4	1000	—	No	50	Raft
[168]	—	—	2	—	—	—	—	—	—
[25]	Yes	Yes	—	—	—	11.46 MB	No	—	—
This work	Yes	Yes	5	5	500	3 GB	Yes	500	Raft

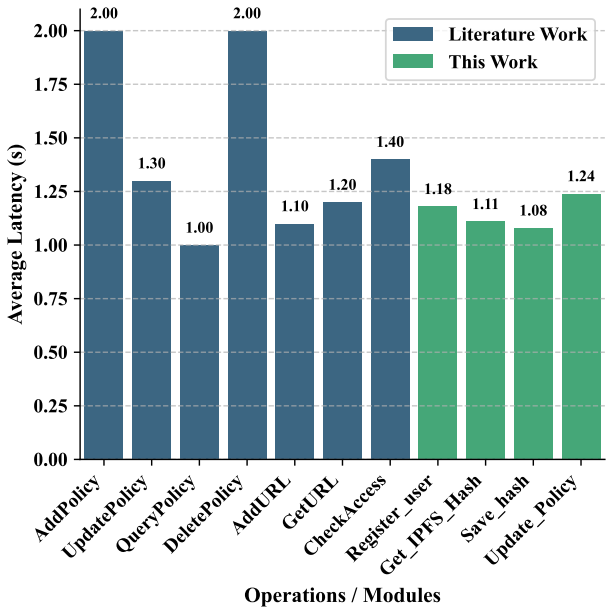


Figure 8.13: **Comparison of Average Latency among Various operations between Literature Work [95] and This Work at a 50 TPS Rate.**

various operations reported in the existing literature [95] with those observed in this study. In their work, the operation latencies ranged between 1.0 and 2.0 seconds, whereas in this study, the average latencies were consistently lower, falling between 1.08 and 1.24 seconds. While the literature does not provide explicit details on the number of concurrent users for their tests, our experiments were conducted with 100 concurrent users under the default endorsement policy (Approach 1) to match the literature configuration. This highlights the efficiency of the proposed system under a clearly defined user load, particularly for operations with comparable functionality such as `Update_Policy`. The results also demonstrate that the proposed system maintains stable latencies even as the workload scales, which strengthens its suitability for deployment in data-intensive environments.

Figure 8.14 compares the throughput (TPS) of operations between the literature [95] and this study.

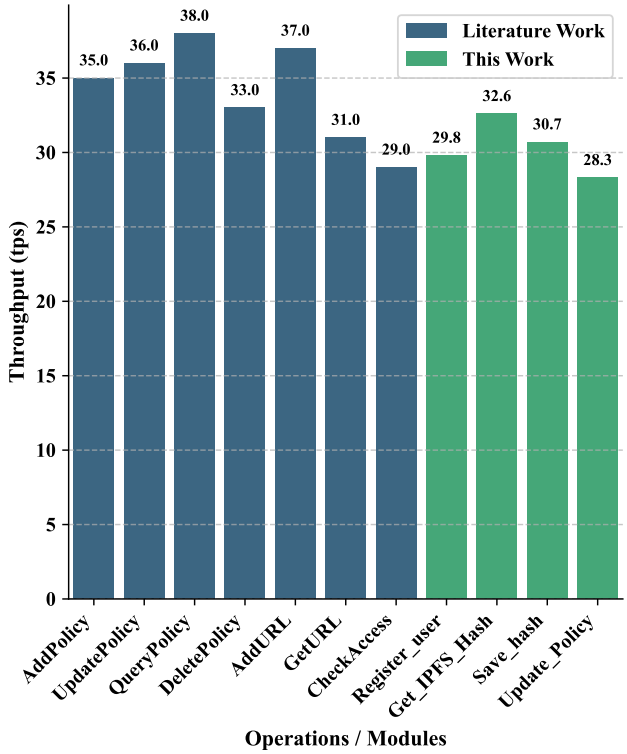


Figure 8.14: Comparison of Performance Throughput between Literature Work [95] and This Work at a 50 TPS Rate among Different Operations / Modules.

For a fair comparison, the results of our system, labelled 'This Work', were obtained using the configuration most similar to that reported in the literature: a 3-organisation, 10-user setup with a steady transaction submission rate of 50 TPS. In contrast, the literature experiments were typically conducted in a two-organisation setup. Although they reported concurrent request numbers (50, 100, 200, 500, and 1000) for cost evaluations, the exact number of concurrent users during their latency and throughput experiments at 5 TPS and 50 TPS was not explicitly stated. This lack of test details makes direct comparisons more challenging. Nevertheless, the results indicate that the performances are generally close under our workloads. Among the comparable operations, only `Get_IPFS_Hash` in our system was higher, achieving 32.6 TPS compared to their `GETURL` operation at 31.0 TPS. Another higher result in our system was observed in a different type of operation, while in the remaining operations, the literature system achieved slightly higher throughput. These outcomes suggest that, although performance levels are broadly comparable, our framework achieves this while operating under more demanding and diverse configurations.

Further contextualising these work findings, Table 8.3 presents a comparison with other summaries of studies that are specifically focused on endorsement policy evaluation. Most prior works typically evaluated only a limited type of endorsement policies, while this study implements and tests 3 configurations. In addition, our evaluation captures a wider range

Table 8.3: Comparison of Endorsement Policy Evaluation Studies

Framework	Channels	EP Tested	EP Types	Orgs	Peers	Metrics	TPS	Users	Key Findings
[144]	16	3	All nested	10–40	–	Throughput, Latency	250	–	Low EP count → faster, less secure; High EP count → slower, more secure
[142]	1	4	AND, OR, OutOf	4	8	Latency	220	2+	OR recommended at high TPS; Higher k in OutOf → ↑latency
[143]	1	3	OR, OutOf, AND	3	12–45	Latency, Throughput, Success	200	100	Larger batch → ↑throughput; More endorsers → slightly ↑latency
This work	1	4	AND, OR, OutOf, Nested	5	5	Latency, Throughput, Success	500	500	Nested EP → moderate speed, ↑concurrency tolerance; Simple EP → faster but less resilient; Complex EP → slower, more resilient

of performance metrics, including throughput, latency, and transaction success rate, under realistic workloads of 500 concurrent users at 500 TPS. This broader scope provides a more nuanced understanding of the trade-offs between policy design, system performance, and security guarantees. By linking EP behaviour with user load and data size, our results extend beyond prior studies and establish more practical insights for large-scale, multiparty deployments.

8.5 Conclusion

This chapter presented a detailed evaluation of our system in different configurations and endorsement policies. By comparing our system with itself, we identified the impact of factors such as the number of users, data size, and transaction rates on throughput, latency, and overall performance. In particular, we observed that increasing the user load had a significant effect on both throughput and success rate, where higher loads tended to saturate the system and increase transaction failures. Similarly, larger data sizes negatively influenced performance by reducing throughput and lowering success rates under intensive workloads. This structured evaluation not only established reliable benchmarks but also highlighted operational bottlenecks that inform both technical improvements and governance-related design decisions.

Among the operations, `Update_Policy` was found to be the most complex and exhibited the lowest performance, whereas operations like `Get_IPFS_Hash` achieved higher throughput under our clearly defined workloads. These findings underline the importance of differentiating

between operations when optimising the system for real-world deployment.

The experiments further demonstrated that complex endorsement policies reduce performance, while simpler or moderately nested policies improved execution speed without severely compromising resilience. This reflects a core trade-off between security and trust guarantees and system efficiency, confirming the central role of endorsement design in balancing trust and scalability.

Based on these analyses, we provide recommendations for selecting endorsement policies and system configurations that balance performance, reliability, and scalability in real-world deployments. Our results suggest that adopting context-aware EP strategies, where governance rules are aligned with stakeholder trust hierarchies, can deliver both robustness and efficiency in data-sharing ecosystems such as autonomous vehicles.

When contextualised with the existing literature, our results generally align with previous observations while also extending them by emphasising the governance dimension of EP design. In contrast to prior studies, our evaluation was conducted under larger and more diverse configurations. This broader scope not only enabled benchmarking against existing works in terms of latency and throughput but also captured additional dimensions such as transaction success rate and the explicit role of endorsement policies. Therefore, this chapter provides actionable insights to guide system architects, researchers, and practitioners in making informed design decisions for subsequent implementation, evaluation, and potential large-scale deployment of Hyperledger Fabric–based multiparty data-sharing frameworks.

Chapter 9

Conclusions

This chapter presents the conclusions and future work. It restates the thesis contributions in Section 9.1 and the summary in Section 9.2. The key limitations are presented in Section 9.3, while future work is discussed in Section 9.4, which highlights possible directions for extension and improvement. The findings of this thesis collectively demonstrate that a permissioned blockchain, when combined with smart contracts and hybrid storage mechanisms, can serve as a viable solution for secure, decentralised, and policy-driven AV data sharing.

9.1 Thesis Contributions

The contributions of this thesis extend across technical development, theoretical advancement, and practical insight. Collectively, they demonstrate how blockchain technology can be adapted and extended to meet the security, scalability, and usability needs of multi-stakeholder AV data sharing. The following list summarises the major contributions of this research.

- **Design and Implementation of a Decentralised Framework for AV Data Sharing.** This thesis presents the design and realisation of a permissioned blockchain-based framework, developed on Hyperledger Fabric, that addresses the challenge of secure and transparent data sharing in multi-stakeholder AV environments. The framework introduces a modular architecture that integrates Web/DApp, storage, and network layers. It enables registration of diverse stakeholders, enforces attribute-based access control, and ensures that data exchanges are auditable and resistant to tampering. By providing a complete prototype system, this work bridges the gap between theoretical models of AV data governance and practical solutions. This contribution includes the following:

  a) **Smart Contract–Based Access Control and Policy Management.** The research develops smart contract modules (chaincode) that implement fine-grained access control mechanisms. These modules allow differentiated access privileges based on stakeholder organisation and attributes and ensure that policy updates and modifications are transparently and immutably recorded on the ledger. This

contributes to the broader understanding of how blockchain can support flexible yet accountable data governance in complex, multi-actor domains.

b) **Hybrid On-Chain / Off-Chain Data Management.** An integration of blockchain with the InterPlanetary File System (IPFS) is introduced to address the scalability challenges of storing large AV datasets. The proposed design anchors data integrity proofs on-chain while storing bulk data off-chain, thereby achieving both performance efficiency and tamper-evidence. This hybrid storage contribution demonstrates how decentralised storage solutions can be practically combined with permissioned blockchains to balance efficiency, cost, and trustworthiness in data-intensive applications.

- **Customisation and Evaluation of Endorsement Policies.** A systematic exploration of endorsement policies in Hyperledger Fabric is conducted, encompassing baseline, customised, and a simulated weighted-voting strategy. This implements and benchmarks these policies within the prototype to assess the viability in AV data-sharing scenarios. The findings highlight the impact of endorsement policy design on system throughput, latency, and trust, thereby advancing the theoretical and practical understanding of consensus flexibility in permissioned blockchains.

- **Comprehensive Performance Evaluation and Benchmarking.** Extensive experimental evaluation is carried out under varying transaction loads, numbers of users, data sizes, and network topologies using state-of-the-art benchmarking tools. The results provide detailed insights into system behaviour across diverse operational conditions, identifying configuration-dependent scalability limits, bottlenecks, and resilience properties. By offering empirical evidence on the trade-offs between performance and trust, this contribution equips future researchers and practitioners with practical guidelines for optimising blockchain-based AV data-sharing frameworks.

- **Human-Centred Usability Evaluation.** Beyond technical validation, this research incorporates a user-focused evaluation of the proposed framework through the System Usability Scale (SUS). The study captures user perceptions of ease of use, interaction quality, and satisfaction, demonstrating that the system is accessible to non-expert users. This contribution highlights the importance of usability in blockchain adoption and provides evidence that a human-centred approach can reduce barriers to engagement with decentralised technologies.

- **Conceptual Advancement through Trade-Off Analysis of Endorsement Policies.** Through the comparative study of different endorsement policies, this work offers a conceptual contribution by articulating the inherent trade-offs between strict security guarantees and system performance efficiency. It demonstrates that endorsement policy design is not merely a configuration choice but a fundamental determinant of trust and performance in blockchain systems. This insight contributes to both academic discourse and practical design decisions in the broader blockchain research community.

Taken together, these contributions provide a holistic advancement to the field of blockchain-based AV data sharing. This thesis provides a complete path from problem and requirement identification to a working prototype and measured evidence. It combines the prototype components, new conceptual understanding, such as the trade-off between endorsement policies that align with use-case requirements and their impact on system

performance, and practical, actionable recommendations that benefit other researchers. It thus bridges the gap between theoretical models of blockchain-based AV data sharing and practical, deployable systems.

9.2 Thesis Summary

AVs generate massive volumes of data from sensors, fused environmental information, and vehicle-to-vehicle or vehicle-to-infrastructure communications. Once generated and retained, these data are essential for reconstructing events, improving system performance, and supporting applications ranging from traffic management to AI model training. However, existing storage and sharing solutions, such as Event Data Recorders (EDRs) and Data Storage Systems for Automated Driving (DSSAD), are limited in scope and capability. They often do not ensure comprehensive data integrity, scalability, and secure access between multiple stakeholders. While prior research has addressed integrity verification, broader challenges, including decentralised and scalable storage, policy-based access control, and multi-party secure sharing, remain insufficiently explored in an integrated manner within multi-stakeholder AV contexts.

This research was motivated by this gap and aimed to develop a comprehensive framework for sharing of stored AV datasets in multi-stakeholder environments. The proposed solution leverages a permissioned blockchain and smart contracts to enable fine-grained, scalable, and decentralised access control for multiple stakeholders, including vehicle manufacturers, regulatory authorities, insurers, and researchers. By tackling challenges from both technical and organisational perspectives, the research emphasises interoperability, accountability, and trustworthiness in AV data ecosystems.

The work began with a Background and Literature Review, which explored the essential information for developing a secure and scalable multi-party data-sharing framework for autonomous vehicles, addressing **RQ1: What are the key components and design principles for developing a secure and scalable multi-party data-sharing framework for stored autonomous vehicles using blockchain?** This chapter explored the essential building blocks of the framework, including the blockchain network, consensus mechanisms, smart contracts, data storage, and access control. It also compared different blockchain platforms, ultimately selecting Hyperledger Fabric as the most suitable option due to its modularity, permissioned structure, and scalability. Furthermore, the chapter classified the types, sources, and flow of AV data, identifying how data is generated, processed, and consumed within the framework. The insights gained highlighted existing gaps in the literature and helped establish the requirements for the proposed solution. It also clarified the rationale for selecting Hyperledger Fabric over other platforms.

Building on this foundation, the research methodology translated these insights into a concrete system design. The blockchain-based framework consists of three integrated modules: the Web module (DApp for stakeholder interaction), the Storage Module (a hybrid on-chain/off-chain solution leveraging IPFS), and the Network Module (a permissioned blockchain ensuring decentralisation, immutability, and secure access control). This approach directly addresses **RQ1** while providing a practical roadmap for implementation and evaluation, demonstrating how theoretical design principles can be operationalised to meet stakeholder

requirements. The system implementation realised this framework, bridging **RQ1** and **RQ2** by deploying Hyperledger Fabric, IPFS, and smart contracts. This practical deployment confirmed the feasibility of the proposed architecture and established a foundation for evaluating scalability and performance. Translating the conceptual design into a functioning system allowed the study to identify how design decisions impacted real-world operation.

The subsequent baseline performance evaluation addressed **RQ2a** and **RQ2b**: How does the framework perform in terms of throughput, latency, and scalability? The evaluation assessed throughput, latency, and success rate in varying transaction rates, number of users, data load, and organisational setups. These experiments established baseline performance behaviour and scalability limitations, highlighting the conditions under which the system performs reliably and where bottlenecks emerge under the tested configurations. Moreover, the evaluation also began addressing **RQ3**: How usable is the proposed blockchain-based data-sharing system from a user perspective? through a System Usability Scale (SUS) assessment. The SUS test provided an initial indication of perceived usability, including stakeholder perceptions of ease of use and quality of interaction.

Further exploration included examining alternative endorsement policies. First, the Customised endorsement policy, extending the exploration of **RQ2** by investigating the impact of different endorsement policies on performance. By comparing baseline configurations with customised policies, the results revealed the trade-offs between complexity, requirements and efficiency.

Then, the simulated weighted-voting endorsement policy introduced and tested an alternative endorsement mechanism, building directly on the investigation of **RQ2a** and **RQ2b**. While the results revealed certain performance trade-offs, they provided valuable insights into the complex relationship between endorsement policy design and system behaviour. Overall, this evaluation of all setups highlighted how endorsement strategies critically shape both the scalability and robustness of blockchain-based AV data-sharing systems, offering practical guidance for optimising policy configurations in diverse operational scenarios

Finally, this research provided a comparative analysis and discussion, offering a holistic interpretation of the results and situating them within the broader literature. It consolidated the findings from all experimental phases, drawing comparisons with existing studies and highlighting the trade-offs between performance and scalability across different endorsement strategies. The discussion emphasised both the strengths and limitations of the proposed solution while offering recommendations for future improvements. It positioned the research results within practical and theoretical contexts.

Through this work, the study answered key research questions regarding the design principles of a secure multi-party data-sharing framework, its scalability and efficiency, and its usability and practical adoption. It also underscored the practical relevance and potential impact of the framework in real-world AV ecosystems.

9.3 Limitations

This study has several limitations, which can be categorised into three groups: (1) prototype-specific limitations, (2) platform-related considerations, and (3) scope limitations.

9.3.1 Prototype-Specific Limitations

These limitations arise from the design and deployment choices of the prototype, rather than inherent characteristics of HLF.

- The evaluation was conducted on a prototype deployed using AWS infrastructure with a limited number of virtual machines (vCPUs, memory, storage, and network bandwidth). While this setup provides flexibility, it does not fully reflect large-scale or production-grade deployments.
- The system was configured with a single ordering service node, which limits fault tolerance and does not represent a fully distributed deployment. This design choice may introduce bottlenecks that are specific to the prototype rather than the underlying platform.
- Large-scale scenarios were constrained by resource and cost limitations, restricting the ability to evaluate behaviour under significantly higher workloads or more complex network topologies.

9.3.2 Platform-Related Considerations

These limitations are associated with the architectural characteristics of HLF itself and reflect design trade-offs inherent in permissioned blockchain systems.

- Hyperledger Fabric relies on endorsement, ordering, and validation phases, which introduce communication and coordination overhead. This can affect scalability under high concurrency and complex transaction workloads.
- The use of external state databases such as CouchDB may introduce performance overhead compared to simpler key-value stores, particularly for complex queries or read-intensive operations.

These factors are inherent to Fabric's design and may influence performance across different deployments, although the exact impact depends on system configuration.

9.3.3 Scope Limitations

These limitations reflect the boundaries of the research and the aspects that were not examined. The scope of this study was restricted to performance and usability evaluation.

Other important aspects, such as energy consumption, regulatory compliance, and ethical considerations of data ownership, were not examined. Additionally, the scenarios considered were relatively simple, and the set of endorsement policies evaluated was limited, focusing primarily on the default and two modified/customised policies. While this enabled controlled analysis and provided insight into the impact of policy changes on system performance, it does not represent the full spectrum of endorsement policy structures supported by Hyperledger Fabric. More complex policies involving multiple organisations, dynamic conditions, or attribute-based rules were not explored and remain an area for future work. These areas remain important for the holistic adoption of AV data-sharing systems and represent opportunities for future investigation. Additionally, the framework codebase developed in this study is currently not publicly available due to ongoing development considerations.

9.4 Future Work

While this thesis has provided a systematic exploration of blockchain-based data sharing for autonomous vehicles, several avenues for future work remain open. The following are some of the potential future work:

- First, the weighted vote endorsement policy proposed in this study was implemented in a static form. A promising extension would be to make the WV-EP dynamic and adaptive to the type of transaction, adjusting the endorsement requirements based on the sensitivity, priority, or criticality of the data being shared. Such an adaptive mechanism could improve both efficiency and security by tailoring trust guarantees to the context of each operation.

- Second, the current deployment relied on a limited set of peers and a single ordering service node (orderer), which restricts both scalability and fault tolerance. Expanding the number of orderer peers would better reflect production-grade networks and provide more realistic information on system resilience, consensus latency, and throughput.

- Third, the framework was deployed with a single communication channel, which simplifies coordination but does not fully capture the complexity of real-world networks. In practice, organisations often participate in multiple channels simultaneously to segregate data access, enforce privacy, and support multiple workflows. Extending the framework to support multi-channel architectures would enable a closer alignment with industry use cases.

- Fourth, while this study focused primarily on throughput, latency, and usability, future research could expand into cost analysis (e.g., resource utilisation, energy consumption, and cloud deployment costs).

- In addition to extending the framework itself, a valuable direction is the development of a Python-based prediction simulator that can study experimental results and then predict performance metrics when test factors change.

- Finally, while this system ensures access control and secure sharing, integrating an encryption layer would further enhance security. Future work could explore client-side

encryption of files before IPFS storage and secure key management, enabling protection against unauthorised access while maintaining the current blockchain-based access control framework.

Appendix A

Consent and Survey Form

System Usability Scale

We would like to invite you to participate in a System Usability Scale (SUS) survey for our data-sharing system. The system is specifically designed for secure sharing of large data sets.

* Indicates required question

Data Sharing Consentection

We assure you that your responses will be kept confidential and your identity will be anonymized. Your data will be used strictly for research purposes and will not be disclosed to any unauthorized individuals or organizations.

1. By proceeding with this survey, you are agreeing to share your data for research purposes. Your personal information will remain confidential and used solely for research analysis. Do you consent to share your data? *

Mark only one oval.

- Yes
- No

Age Verification

To participate in this survey, you must be 18 years of age or older. Please confirm that you meet this requirement.

2. Are you 18 years of age or older? *

Mark only one oval.

- Yes
- No

Demographic Information

3. What is your educational background? *

4. *

What is your professional experience in data sharing or access control?

5. *

In which industry or sector do you primarily work?

Survey

To ensure a comprehensive evaluation, please carefully use the system, upload encrypted large data sets with their own access policies, order data sets, and provide feedback through the following **10 questions**. The collected feedback will help improve the system's usability, security, and overall performance. Thank you for your cooperation

6. 1- I think that I would like to use this system frequently. *

Mark only one oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly Agree

7. 2- I found the system unnecessarily complex. *

Mark only one oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly Agree

8. 3- I thought the system was easy to use. *

Mark only one oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly Agree

9. 4- I think that I would need the support of a technical person to be able to use this system. *

Mark only one oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly Agree

10. 5- I found the various functions in this system were well integrated. *

Mark only one oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly Agree

11. 6- I thought there was too much inconsistency in this system. *

Mark only one oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly Agree

12. 7- I would imagine that most people would learn to use this system very quickly. *

Mark only one oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly Agree

13. 8- I found the system very cumbersome to use. *

Mark only one oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly Agree

14. 9- I felt very confident using the system. *

Mark only one oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly Agree

15. 10- I needed to learn a lot of things before I could get going with this system. *

Mark only one oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly Agree

Thank you for your valuable time and participation in this survey. Your input and insights are greatly appreciated and will contribute significantly to our research.

List of References

[1] N. H. T. S. Administration, "Url: http://www. nhtsa. gov/about+ nhtsa," *Press+ Releases/nhtsa-iihscommitment-on-aeb-09112015 on December*, vol. 2, p. 2015, 2015.

[2] A. Papadoulis, M. Quddus, and M. Imprialou, "Evaluating the safety impact of connected and autonomous vehicles on motorways," *Accident Analysis and Prevention*, vol. 124, pp. 12–22, 2019.

[3] S. Pan, L. M. Fulton, A. Roy, J. Jung, Y. Choi, and H. O. Gao, "Shared Use of Electric Autonomous Vehicles: Air Quality and Health Impacts of Future Mobility in the United States," *Renewable and Sustainable Energy Reviews*, vol. 149, p. 111380, 2021.

[4] C. Winston and Q. Karpilow, *Autonomous vehicles: The road to economic growth?* Brookings Institution Press, 2020.

[5] S. Liu, "The business case for infrastructure-vehicle cooperative autonomous driving," *IEEE Engineering Management Review*, 2022.

[6] UK Government, "Code of practice for on-road trials, available at: https://www.gov.uk/," 2019. Accessed: 15 June 2022.

[7] UK Government, "Rules on safe use of automated vehicles on gb roads available at: https://www.gov.uk/," 2022. Accessed: 29 September 2023.

[8] J. Li, Z. Xue, C. Li, and M. Liu, "RTED-SD: A Real-Time Edge Detection Scheme for Sybil DDoS in the Internet of Vehicles," *IEEE Access*, vol. 9, pp. 11296–11305, 2021.

[9] A. Anwar, T. Halabi, and M. Zulkernine, "A Dynamic Threat Prevention Framework for Autonomous Vehicle Networks Based on Ruin-Theoretic Security Risk Assessment," *Journal on Autonomous Transportation Systems*, vol. 1, no. 4, pp. 1–28, 2024.

[10] S. Aurangzeb, M. Aleem, M. T. Khan, H. Anwar, and M. S. Siddique, "Cybersecurity for Autonomous Vehicles Against Malware Attacks in Smart-Cities," *Cluster Computing*, vol. 27, no. 3, pp. 3363–3378, 2024.

[11] A.-M. Cretu, F. Houssiau, A. Cully, and Y.-A. de Montjoye, "QuerySnout: Automating the Discovery of Attribute Inference Attacks Against Query-based Systems," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pp. 623–637, 2022.

[12] C. Anthony, W. Elgenaidi, and M. Rao, "Intrusion Detection System for Autonomous Vehicles Using Non-Tree based Machine Learning Algorithms," *Electronics*, vol. 13, no. 5, p. 809, 2024.

[13] H. Setia, A. Chhabra, S. K. Singh, S. Kumar, S. Sharma, V. Arya, B. B. Gupta, and J. Wu, "Securing the Road Ahead: Machine Learning-Driven DDoS Attack Detection in VANET Cloud Environments," *Cyber Security and Applications*, vol. 2, p. 100037, 2024.

[14] A. Sultan, S. Tahir, H. Tahir, T. Anwer, F. Khan, M. Rajarajan, and O. Rana, "A Novel Image-Based Homomorphic Approach for Preserving the Privacy of Autonomous Vehicles Connected to the Cloud," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 2, pp. 1936–1948, 2023.

[15] Y. Zhang, J. Zou, and R. Guo, "Efficient privacy-preserving authentication for V2G networks," *Peer-to-Peer Networking and Applications*, vol. 14, no. 3, pp. 1366–1378, 2021.

[16] R. Parekh, N. Patel, R. Gupta, N. K. Jadav, S. Tanwar, A. Alharbi, A. Tolba, B.-C. Neagu, and M. S. Raboaca, "Gefl: Gradient Encryption-Aided Privacy Preserved Federated Learning for Autonomous Vehicles," *IEEE Access*, vol. 11, pp. 1825–1839, 2023.

[17] M. Gheisari, W. Z. Khan, H. E. Najafabadi, G. McArdle, H. Rabiei-Dastjerdi, Y. Liu, C. Fernández-Campusano, and H. B. Abdalla, "CAPPAD: a Privacy-Preservation Solution for Autonomous Vehicles using SDN, Differential Privacy and Data Aggregation," *Applied Intelligence*, vol. 54, no. 4, pp. 3417–3428, 2024.

[18] T. Bai, Q. Yang, and S. Fu, "User-Defined Privacy Preserving Data Sharing for Connected Autonomous Vehicles Utilizing Edge Computing," in *Proceedings of the Eighth ACM/IEEE Symposium on Edge Computing*, SEC '23, (New York, NY, USA), p. 145–157, Association for Computing Machinery, 2024.

[19] Y. Zhao, D. Gong, S. Wen, L. Ding, and G. Guo, "A Privacy-Preserving-Based Distributed Collaborative Scheme for Connected Autonomous Vehicles at Multi-Lane Signal-Free Intersections," *IEEE Transactions on Intelligent Transportation Systems*, 2024.

[20] C. Oham, S. S. Kanhere, R. Jurdak, and S. Jha, "A blockchain based liability attribution framework for autonomous vehicles," *arXiv preprint arXiv:1802.05050*, 2018.

[21] J. Ryu, Y. Lee, and Y. Yoon, "Blockchain Model for Reliable Consensus Algorithm on the Autonomous Driving Data Management," in *2024 IEEE International Conference on Consumer Electronics (ICCE)*, pp. 1–4, 2024.

[22] X. Shen, Y. Lu, Y. Zhang, X. Liu, and L. Zhang, "An Innovative Data Integrity Verification Scheme in the Internet of Things Assisted information Exchange in Transportation Systems," *Cluster Computing*, vol. 25, no. 3, pp. 1791–1803, 2022.

[23] S. Lee, W. Choi, H. J. Jo, and D. H. Lee, "T-box: A forensics-enabled trusted automotive data recording method," *IEEE Access*, vol. 7, pp. 49738–49755, 2019.

[24] W. Liu, W. Shen, L. Harn, and M. Luo, "A fast vanet-assisted scheme for event data recorders," *Security and Communication Networks*, vol. 2022, 2022.

[25] A. Singh, S. Sural, T. Sengupta, and S. Sural, "Trusted Sharing of Autonomous Vehicle Crash Data using Enterprise Blockchain and IPFS," in *Proceedings of the 5th ACM International Symposium on Blockchain and Secure Critical Infrastructure*, pp. 11–24, 2023.

[26] D. Omeiza, H. Webb, M. Jirotka, and L. Kunze, "Explanations in autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[27] I. Gupta and A. K. Singh, "An integrated approach for data leaker detection in cloud environment," *Journal of Information Science and Engineering*, vol. 36, no. 5, pp. 993–1007, 2020.

[28] M. Ali, S. U. R. Malik, and S. U. Khan, "DaSCE: Data security for cloud environment with semi-trusted third party," *IEEE Transactions on Cloud Computing*, vol. 5, pp. 642–655, Oct 2017.

[29] V. Gowadia, E. Scalavino, E. C. Lupu, D. Starostin, and A. Orlov, "Secure cross-domain data sharing architecture for crisis management," in *Proceedings of the Tenth Annual ACM Workshop on Digital Rights Management*, DRM '10, (New York, NY, USA), p. 43–46, Association for Computing Machinery, 2010.

[30] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[31] Consensys, "Quorum Documentation," 2021. Available at: `https://docs.goquorum.consensys.net/en/stable/Concepts`. Accessed: 2024-03-07.

[32] Hyperledger Project, "Hyperledger sawtooth architecture and design," tech. rep., The Linux Foundation, 2018. Electronic White Paper.

[33] H. F. Project, "A blockchain platform for the enterprise: Hyperledger fabric," 2019.

[34] J. Gwehenberger, O. Braxmeier, C. Lauterwasser, M. A. Kreutner, M. Borrack, C. Reinkemeyer, L. Wech, M. Weyde, and P. Salzberger, "Needs and Requirements of EDR for Automated Vehicles: Analysis Based on Insurance Claims Reported to Allianz Germany," tech. rep., Allianz Center for Technology (AZT), 2022. Presented at UNECE EDR Workshop, March 2022.

[35] P. Koopman and M. Wagner, "Autonomous Vehicle Safety: An Interdisciplinary Challenge," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 1, pp. 90–96, 2017.

[36] D. Paula, K. Böhm, T. Kubjatko, and H.-G. Schweiger, "Challenges in forensic reconstruction of traffic accidents involving advanced driver assistance systems (adas)," in *29th Annual Congress of the European Association for Accident Research (EVU)*, 2020.

[37] K. K. G. Buquerin, C. Corbett, and H.-J. Hof, "A generalized approach to automotive forensics," *Forensic Science International: Digital Investigation*, vol. 36, p. 301111, 2021.

[38] M. A. Hoque and R. Hasan, "Avguard: A forensic investigation framework for autonomous vehicles," in *ICC 2021-IEEE International Conference on Communications*, pp. 1–6, IEEE, 2021.

[39] J. T. Correia, K. A. Iliadis, E. S. McCarron, M. A. Smolej, B. Hastings, and C. C. Engineers, "Utilizing data from automotive event data recorders," in *Proceedings of the Canadian Multidisciplinary Road Safety Conference XII, London Ontario*, p. 18, 2001.

[40] Y. Fang, H. Min, X. Wu, X. Lei, S. Chen, R. Teixeira, and X. Zhao, "Toward interpretability in fault diagnosis for autonomous vehicles: Interpretation of sensor data anomalies," *IEEE Sensors Journal*, vol. 23, no. 5, pp. 5014–5027, 2023.

[41] A. Alniamy and B. D. Taylor, "Attribute-based access control of data sharing based on hyperledger blockchain," in *Proceedings of the 2020 The 2nd International Conference on Blockchain Technology*, pp. 135–139, 2020.

[42] T. Alshalali, K. M'Bale, and D. Josyula, "Security and privacy of electronic health records sharing using hyperledger fabric," in *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*, pp. 760–763, IEEE, 2018.

[43] R. Alhabib and P. Yadav, "Data sharing in autonomous vehicles: Hyperledger fabric platform for secure and efficient sharing," in *NDSS VehicleSec 2024*, Usenix, 2024.

[44] S. Liu and Y. Shang, "Secure resource sharing on hyperledger fabric based on cp-abe," in *2021 The 3rd International Conference on Blockchain Technology*, pp. 203–209, 2021.

[45] J. Benet, "Ipfs-content addressed, versioned, p2p file system," *arXiv preprint arXiv:1407.3561*, 2014.

[46] N. R. Fulbright, "The privacy implications of autonomous vehicles." Report, Norton Rose Fulbright, jul 2017.

[47] "Net zero by 2050: A roadmap for the global energy sector," 2021. https://trid.trb.org/view/1856381.

[48] P. S. Perumal, M. Sujasree, S. Chavhan, D. Gupta, V. Mukthineni, S. R. Shimgekar, A. Khanna, and G. Fortino, "An insight into crash avoidance and overtaking advice systems for autonomous vehicles: A review, challenges and solutions," *Engineering applications of artificial intelligence*, vol. 104, p. 104406, 2021.

[49] C. D. Harper, C. T. Hendrickson, and C. Samaras, "Cost and benefit estimates of partially-automated vehicle collision avoidance technologies," *Accident Analysis and Prevention*, vol. 95, pp. 104–115, 2016.

[50] R. Rajamani, *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

[51] D. P. Watson and D. H. Scheidt, "Autonomous systems," *Johns Hopkins APL technical digest*, vol. 26, no. 4, pp. 368–376, 2005.

[52] SAE International, "Sae j3016 updated: Levels of driving automation," 2021. Accessed: 2021.

[53] J. Vargas, S. Alsweiss, O. Toker, R. Razdan, and J. Santos, "An overview of autonomous vehicles sensors and their vulnerability to weather conditions," *Sensors*, vol. 21, no. 16, p. 5397, 2021.

[54] Y. Wang, Y. He, R. Wang, and W. Shi, "Quantitative analysis of storage requirement for autonomous vehicles," in *Proceedings of the 16th ACM Workshop on Hot Topics in Storage and File Systems*, pp. 71–78, 2024.

[55] D. J. Yeong, G. Velasco-Hernandez, J. Barry, J. Walsh, *et al.*, "Sensor and sensor fusion technology in autonomous vehicles: A review," *Sensors*, vol. 21, no. 6, p. 2140, 2021.

[56] J. Fayyad, M. A. Jaradat, D. Gruyer, and H. Najjaran, "Deep learning sensor fusion for autonomous vehicle perception and localization: A review," *Sensors*, vol. 20, no. 15, p. 4220, 2020.

[57] S. Campbell, N. O'Mahony, L. Krpalcova, D. Riordan, J. Walsh, A. Murphy, and C. Ryan, "Sensor technology in autonomous vehicles: A review," in *2018 29th Irish Signals and Systems Conference (ISSC)*, pp. 1–4, IEEE, 2018.

[58] J. Wishart, S. Como, U. Forgione, J. Weast, *et al.*, "Literature review of verification and validation activities of automated driving systems," *SAE Int. J. Connect. Automat. Veh*, vol. 3, no. 4, pp. 267–323, 2020.

[59] V. Kulshrestha and K. R. Jagdale, "Disruptive technology directions for 6g," in *Towards Wireless Heterogeneity in 6G Networks*, pp. 18–32, CRC Press, 2024.

[60] M. Noor-A-Rahim, Z. Liu, H. Lee, M. O. Khyam, J. He, D. Pesch, K. Moessner, W. Saad, and H. V. Poor, "6g for vehicle-to-everything (v2x) communications: Enabling technologies, challenges, and opportunities," *Proceedings of the IEEE*, vol. 110, no. 6, pp. 712–734, 2022.

[61] K. Böhm, T. Kubjatko, D. Paula, and H.-G. Schweiger, "New developments on edr (event data recorder) for automated vehicles," *Open Engineering*, vol. 10, no. 1, pp. 140–146, 2020.

[62] R. Alhabib and P. Yadav, "Data authorisation and validation in autonomous vehicles: A critical review," *Discover Applied Sciences*, vol. 7, no. 7, p. 735, 2025.

[63] S. Kato, E. Takeuchi, Y. Ishiguro, Y. Ninomiya, K. Takeda, and T. Hamada, "An open approach to autonomous vehicles," *IEEE Micro*, vol. 35, no. 6, pp. 60–68, 2015.

[64] W. Xu, N. Souly, and P. P. Brahma, "Reliability of gan generated data to train and validate perception systems for autonomous vehicles," in *Proceedings of the ieee/cvf winter conference on applications of computer vision*, pp. 171–180, 2021.

[65] H. Alghodhaifi and S. Lakshmanan, "Autonomous vehicle evaluation: A comprehensive survey on modeling and simulation approaches," *Ieee Access*, vol. 9, pp. 151531–151566, 2021.

[66] F. Rosique, P. J. Navarro, C. Fernández, and A. Padilla, "A systematic review of perception system and simulators for autonomous vehicles research," *Sensors*, vol. 19, no. 3, p. 648, 2019.

[67] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*, pp. 1–16, PMLR, 2017.

[68] S. Kato *et al.*, "Autoware on board: Enabling autonomous vehicles with embedded systems," *IEEE Intelligent Vehicles Symposium (IV)*, 2018.

[69] Siemens, "Prescan: Simulation platform for adas and automated driving," 2023. https://www.plm.automation.siemens.com/.

[70] VIRES, "Vtd - virtual test drive," 2023. https://www.vires.com/.

[71] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *NeurIPS Workshop on Machine Learning for Intelligent Transportation Systems*, 2017.

[72] NVIDIA, "Nvidia drive sim: Scalable, physically based autonomous vehicle simulation," 2023.

[73] L. Technologies, "Atlas simulation for lidar development," 2023. https://www.luminartech.com/.

[74] P. Sun, H. Kretzschmar, X. Dotiwalla, *et al.*, "Scalability in perception for autonomous driving: Waymo open dataset," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[75] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11621–11631, 2020.

[76] Comma.ai, "Openpilot: Open source driver assistance system," 2023. https://comma.ai/.

[77] A. Lab, "Ai for construction and environment," 2023. https://ai4ce.github.io/.

[78] P. Rogaway and T. Shrimpton, "Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance," in *FSE*, vol. 3017, pp. 371–388, 2004.

[79] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized business review*, p. 21260, 2008.

[80] Ripple, "Ripple: Crypto solutions for business," 2023. Available at: `https://ripple.com/`. Accessed: June 2023.

[81] V. C. Hu, D. R. Kuhn, D. F. Ferraiolo, and J. Voas, "Attribute-based access control," *Computer*, vol. 48, no. 2, pp. 85–88, 2015.

[82] V. Trón and A. Fischer, "Swarm: The decentralised storage and communication system for a sovereign digital society." `https://www.ethswarm.org/swarm-whitepaper.pdf`, 2021. Accessed: 2025-05-21.

[83] M. Buus and the Hypercore Protocol team, "Hypercore protocol documentation." `https://hypercore-protocol.org`, 2021. Accessed: 2025-05-21.

[84] MaidSafe, "The architecture of the safe network." `https://safenetwork.tech/whitepaper/`, 2014. Accessed: 2025-05-21.

[85] S. Wilkinson and T. Boshevski, "Storj: A peer-to-peer cloud storage network." `https://storj.io/storj.pdf`, 2014. Accessed: 2025-05-21.

[86] S. Williams, "Arweave: A protocol for economically sustainable information permanence." `https://www.arweave.org/files/arweave-lightpaper.pdf`, 2018. Accessed: 2025-05-21.

[87] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips, "The bittorrent p2p file-sharing system: Measurements and analysis," in *Peer-to-Peer Systems IV: 4th International Workshop, IPTPS 2005, Ithaca, NY, USA, February 24-25, 2005. Revised Selected Papers 4*, pp. 205–216, Springer, 2005.

[88] O. Abdullah Lajam and T. Ahmed Helmy, "Performance evaluation of ipfs in private networks," in *Proceedings of the 2021 4th International Conference on Data Storage and Data Engineering*, pp. 77–84, 2021.

[89] E. Daniel and F. Tschorsch, "Ipfs and friends: A qualitative comparison of next generation peer-to-peer data networks," *IEEE Communications Surveys and Tutorials*, vol. 24, no. 1, pp. 31–52, 2022.

[90] M. Joshi, K. Joshi, and T. Finin, "Attribute based encryption for secure access to cloud based ehr systems," in *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pp. 932–935, IEEE, 2018.

[91] G. Bianchi, T. Dargahi, A. Caponi, and M. Conti, "Intelligent conditional collaborative private data sharing," *Future Generation Computer Systems*, vol. 96, pp. 1–10, 2019.

[92] S. Sharma, Y. Li, S. Mehrotra, N. Panwar, P. Gupta, and D. Ghosh, "Prism: Privacy-preserving and verifiable set computation over multi-owner secret shared outsourced databases," *IEEE Transactions on Dependable and Secure Computing*, vol. 21, no. 3, pp. 1355–1371, 2023.

[93] H. Guo, W. Li, M. Nejad, and C.-C. Shen, "Access control for electronic health records with hybrid blockchain-edge architecture," in *2019 IEEE International Conference on Blockchain (Blockchain)*, pp. 44–51, IEEE, 2019.

[94] J. Chi, Y. Li, J. Huang, J. Liu, Y. Jin, C. Chen, and T. Qiu, "A secure and efficient data sharing scheme based on blockchain in industrial internet of things," *Journal of Network and Computer Applications*, vol. 167, p. 102710, 2020.

[95] E. A. Shammar, A. T. Zahary, and A. A. Al-Shargabi, "An attribute-based access control model for internet of things using hyperledger fabric blockchain," *Wireless Communications and Mobile Computing*, vol. 2022, no. 1, p. 6926408, 2022.

[96] X. Ma, C. Wang, and X. Chen, "Trusted data sharing with flexible access control based on blockchain," *Computer Standards and Interfaces*, vol. 78, p. 103543, 2021.

[97] X. Liu, X. Yang, Y. Luo, L. Wang, and Q. Zhang, "Anonymous electronic health record sharing scheme based on decentralized hierarchical attribute-based encryption in cloud environment," *IEEE Access*, vol. 8, pp. 200180–200193, 2020.

[98] J. Chen, C. Zhang, Y. Yan, and Y. Liu, "Filewallet: A file management system based on ipfs and hyperledger fabric.," *CMES-Computer Modeling in Engineering and Sciences*, vol. 130, no. 2, 2022.

[99] Z. Wang and S. Guan, "A blockchain-based traceable and secure data-sharing scheme," *PeerJ Computer Science*, vol. 9, p. e1337, 2023.

[100] X. Zhao, S. Wang, Y. Zhang, and Y. Wang, "Attribute-based access control scheme for data sharing on hyperledger fabric," *Journal of Information Security and Applications*, vol. 67, p. 103182, 2022.

[101] M. Zhaofeng, W. Xiaochang, D. K. Jain, H. Khan, G. Hongmin, and W. Zhen, "A blockchain-based trusted data management scheme in edge computing," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 2013–2021, 2019.

[102] S. K. Dwivedi, R. Amin, S. Vollala, and R. Chaudhry, "Blockchain-based Secured Event-Information Sharing Protocol in Internet of Vehicles for Smart Cities," *Computers & Electrical Engineering*, vol. 86, p. 106719, 2020.

[103] H. Yi, "A Secure Blockchain System for Internet of Vehicles based on 6G-Enabled Network in Box," *Computer Communications*, vol. 186, pp. 45–50, 2022.

[104] G. Liu, H. Dong, Z. Yan, X. Zhou, and S. Shimizu, "B4SDC: A blockchain System for Security Data Collection in MANETs," *IEEE Transactions on Big Data*, 2020.

[105] T. Zeng, O. Semiari, M. Chen, W. Saad, and M. Bennis, "Federated learning on the road autonomous controller design for connected and autonomous vehicles," *IEEE Transactions on Wireless Communications*, vol. 21, no. 12, pp. 10407–10423, 2022.

[106] Y. He, K. Huang, G. Zhang, F. R. Yu, J. Chen, and J. Li, "Bift: A blockchain-based federated learning system for connected and autonomous vehicles," *IEEE Internet of Things Journal*, vol. 9, no. 14, pp. 12311–12322, 2022.

[107] J. Xiong, R. Bi, M. Zhao, J. Guo, and Q. Yang, "Edge-assisted privacy-preserving raw data sharing framework for connected autonomous vehicles," *IEEE Wireless Communications*, vol. 27, no. 3, pp. 24–30, 2020.

[108] N. Leveson, "A new accident model for engineering safer systems," *Safety Science*, vol. 42, no. 4, pp. 237–270, 2004.

[109] E. Hollnagel, *FRAM: the functional resonance analysis method: modelling complex socio-technical systems*. Crc Press, 2017.

[110] J. Beck, R. Arvin, S. Lee, A. Khattak, and S. Chakraborty, "Automated Vehicle Data Pipeline for Accident Reconstruction: New Insights from LiDAR, Camera, and Radar Data," *Accident Analysis & Prevention*, vol. 180, p. 106923, 2023.

[111] X. Zhang, J. Tao, K. Tan, M. Törngren, J. M. G. Sánchez, M. R. Ramli, X. Tao, M. Gyllenhammar, F. Wotawa, N. Mohan, *et al.*, "Finding critical scenarios for automated driving systems: A systematic literature review," *arXiv preprint arXiv:2110.08664*, 2021.

[112] M. Abdel-Aty and S. Ding, "A matched case-control analysis of autonomous vs human-driven vehicle accidents," *Nature communications*, vol. 15, no. 1, p. 4931, 2024.

[113] A. D. M. Ibrahum, M. Hussain, and J.-E. Hong, "Deep learning adversarial attacks and defenses in autonomous vehicles: a systematic literature review from a safety perspective," *Artificial Intelligence Review*, vol. 58, no. 1, pp. 1–53, 2025.

[114] X. Lin, L. Chen-Ying, and C. K. Fan, "Exploring the impacts of autonomous vehicles on the insurance industry and strategies for adaptation," *World Electric Vehicle Journal*, vol. 16, no. 3, p. 119, 2025.

[115] A. Budel, R. Alhabib, M. Nicholson, and P. Yadav, "Vincy: A smart-contract based data integrity and validation tooling for automated vehicle incident investigation," 2023.

[116] R. Alhabib and P. Yadav, "Hyperledger fabric platform for secure and efficient data sharing in autonomous vehicles," in *Proceedings of the WINCOM Conference*, IEEE Communications Society, 2024.

[117] R. Alhabib and P. Yadav, "Impact of custom endorsement policies on hyperledger fabric performance in autonomous vehicle data sharing platforms," in *2024 6th International Conference on Blockchain Computing and Applications (BCCA)*, pp. 635–641, IEEE, 2024.

[118] T. V. Doan, Y. Psaras, J. Ott, and V. Bajpai, "Toward decentralized cloud storage with ipfs: opportunities, challenges, and future considerations," *IEEE Internet Computing*, vol. 26, no. 6, pp. 7–15, 2022.

[119] V. Jayadev, N. Moradpoor, and A. Petrovski, "Assessing the performance of ethereum and hyperledger fabric under ddos attacks for cyber-physical systems," in *Proceedings of the 19th International Conference on Availability, Reliability and Security*, pp. 1–6, 2024.

[120] S. Dalla Palma, R. Pareschi, and F. Zappone, "What is your distributed (hyper) ledger?," in *2021 IEEE/ACM 4th International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)*, pp. 27–33, IEEE, 2021.

[121] T. Guggenberger, J. Sedlmeir, G. Fridgen, and A. Luckow, "An in-depth investigation of the performance characteristics of hyperledger fabric," *Computers and Industrial Engineering*, vol. 173, p. 108716, 2022.

[122] S. Shalaby, A. A. Abdellatif, A. Al-Ali, A. Mohamed, A. Erbad, and M. Guizani, "Performance evaluation of hyperledger fabric," in *2020 IEEE international conference on informatics, IoT, and enabling technologies (ICIoT)*, pp. 608–613, IEEE, 2020.

[123] Y. Li, D. Ma, Z. An, Z. Wang, Y. Zhong, S. Chen, and C. Feng, "V2x-sim: Multi-agent collaborative perception dataset and benchmark for autonomous driving," *IEEE Robotics and Automation Letters*, 2022.

[124] P. Thakkar, S. Nathan, and B. Viswanathan, "Performance benchmarking and optimizing hyperledger fabric blockchain platform," in *2018 IEEE 26th international symposium on modeling, analysis, and simulation of computer and telecommunication systems (MASCOTS)*, pp. 264–276, IEEE, 2018.

[125] "Caliper: An open-source performance testing tool for evaluating system scalability." Available online: `https://github.com/hyperledger/caliper/` Accessed: Jan 15, 2023.

[126] T. T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, and K.-L. Tan, "Blockbench: A framework for analyzing private blockchains," in *Proceedings of the 2017 ACM international conference on management of data*, pp. 1085–1100, 2017.

[127] E. H. Halili, *Apache JMeter*. Packt Publishing, 2008.

[128] Grafana Labs, "Load testing for engineering teams — grafana k6." `https://k6.io/`, n.d. Accessed: 2025-06-25.

[129] C. Udokwu, H. Anyanka, and A. Norta, "Evaluation of approaches for designing and developing decentralized applications on blockchain," in *Proceedings of the 4th International Conference on Algorithms, Computing and Systems*, pp. 55–62, 2020.

[130] R. Saraiva, A. A. Araújo, P. Soares, J. C. Pontes, and J. Souza, "Metrics for quality assessment in blockchain-based systems: A systematic mapping study," *Simpósio Brasileiro de Sistemas de Informação (SBSI)*, pp. 545–554, 2025.

[131] P. Schweiger, *Improving Usability of Blockchain-Based Decentralized Applications*. PhD thesis, University of Applied Sciences Technikum Wien Wien, Austria, 2021.

[132] S. S. Gandhi, Y. N. Patil, L. D. Netak, and H. R. Gaikwad, "Usability analysis for blockchain-based applications," in *International Conference on Intelligent Human Computer Interaction*, pp. 349–360, Springer, 2021.

[133] J. Brooke, "Sus: a "quick and dirty'usability," *Usability evaluation in industry*, vol. 189, no. 3, pp. 189–194, 1996.

[134] J. Brooke, "Sus: a retrospective," *Journal of usability studies*, vol. 8, no. 2, pp. 29–40, 2013.

[135] M. Fröhlich, F. Waltenberger, L. Trotter, F. Alt, and A. Schmidt, "Blockchain and cryptocurrency in human computer interaction: a systematic literature review and research agenda," in *Proceedings of the 2022 ACM Designing Interactive Systems Conference*, pp. 155–177, 2022.

[136] J. W. Kim, S. J. Kim, W. C. Cha, and T. Kim, "A blockchain-applied personal health record application: Development and user experience," *Applied Sciences*, vol. 12, no. 4, p. 1847, 2022.

[137] G. Berné Melguizo, "Usability in blockchain: Experimental analysis of platforms for descentralized autonomous organizations," 2023.

[138] H. of Lords, "Automated vehicles bill [hl]." Parliament of the United Kingdom, 2024.

[139] "Data ownership in smart cars," *Harvard Journal of Law and Technology*, vol. 32, p. 299, 2018.

[140] E. Union, "Data protection." `https://ec.europa.eu/info/law/law-topic/data-protection_en`, 2024. Accessed on: May 15, 2023.

[141] C. Melo, G. Gonçalves, F. A. Silva, and A. Soares, "Performance modeling and evaluation of hyperledger fabric: An analysis based on transaction flow and endorsement policies," in *2024 IEEE Symposium on Computers and Communications (ISCC)*, pp. 1–6, IEEE, 2024.

[142] X. Piao, H. Ding, and H. Song, "Performance analysis of endorsement in hyperledger fabric concerning endorsement policies," *Electronics*, vol. 12, no. 20, p. 4322, 2023.

[143] N. Mishra and H. Levkowitz, "Performance evaluation of permissioned-based personal data vault implemented using hyperledger fabric v2. x," in *2022 International Conference on Intelligent Data Science Technologies and Applications (IDSTA)*, pp. 138–145, IEEE, 2022.

[144] P. Gaba and R. S. Raw, "Impact of endorsement policy on the performance of blockchain-based vanet," *Transactions on Emerging Telecommunications Technologies*, vol. 34, no. 9, p. e4822, 2023.

[145] J. A. Chacko, R. Mayer, and H.-A. Jacobsen, "Why do my blockchain transactions fail? a study of hyperledger fabric," in *Proceedings of the 2021 international conference on management of data*, pp. 221–234, 2021.

[146] R. Kawahara, "Verification of customizable blockchain consensus rule using a formal method," in *2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pp. 1–3, IEEE, 2020.

[147] L. De Moura and N. Bjørner, "Z3: An efficient smt solver," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pp. 337–340, Springer, 2008.

[148] D. T. Dang and D. Hwang, "Consensus-based methods for distributed systems, blockchain, and voting: a survey," *Journal of Information and Telecommunication*, pp. 1–24, 2024.

[149] A. Brodovic, *Using Weighted Voting to Accelerate Blockchain Consensus*. PhD thesis, Delft University of Technology, 2024.

[150] M. Açıkkar and S. Tokgöz, "An improved knn classifier based on a novel weighted voting function and adaptive k-value selection," *Neural Computing and Applications*, vol. 36, no. 8, pp. 4027–4045, 2024.

[151] Y. Zhou, Y. Huo, Q. Gao, Y. Wu, T. Jing, and J. Mao, "Securing collaborative authentication: A weighted voting strategy to counter unreliable cooperators," *IEEE Transactions on Information Forensics and Security*, 2024.

[152] Hyperledger, "Endorsement policies," 2024. Accessed: May. 14, 2024.

[153] C. Melo, G. Gonçalves, F. A. Silva, and A. Soares, "A comprehensive hyperledger fabric performance evaluation based on resources capacity planning," *Cluster Computing*, vol. 27, no. 9, pp. 12395–12410, 2024.

[154] H. Javaid, C. Hu, and G. Brebner, "Optimizing validation phase of hyperledger fabric," in *2019 IEEE 27th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pp. 269–275, 2019.

[155] Node.js, "The child_process module," 2023. Available at: `https://nodejs.org/api/child_process.html`, Accessed: 21 September 2023.

[156] E. Androulaki, A. De Caro, M. Neugschwandtner, and A. Sorniotti, "Endorsement in hyperledger fabric," in *2019 IEEE International Conference on Blockchain (Blockchain)*, pp. 510–519, IEEE, 2019.

[157] J. Dharani, K. Sundarakantham, S. Mercy Shalinie, *et al.*, "A privacy-preserving framework for endorsement process in hyperledger fabric," *Computers & Security*, vol. 116, p. 102637, 2022.

[158] M. Soelman, V. Andrikopoulos, J. A. Pérez, V. Theodosiadis, K. Goense, and A. Rutjes, "Hyperledger fabric: Evaluating endorsement policy strategies in supply chains," in *2020 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)*, pp. 145–152, IEEE, 2020.

[159] S. Mazumdar and S. Ruj, "Design of anonymous endorsement system in hyperledger fabric," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 4, pp. 1780–1791, 2019.

[160] M. Kwon and H. Yu, "Performance improvement of ordering and endorsement phase in hyperledger fabric," in *2019 sixth international conference on internet of things: systems, management and security (IOTSMS)*, pp. 428–432, IEEE, 2019.

[161] J. Yu, L. Ge, and M. Wu, "Proposal distribution optimization for endorsement strategy in hyperledger fabric," *The Journal of Supercomputing*, pp. 1–28, 2024.

[162] I. Lotfimahyari and P. Giaccone, "Optimal endorsement for network-wide distributed blockchains," *IEEE Systems Journal*, 2023.

[163] Y. Lee, D. Lee, J. Oh, L. Park, W. Na, and S. Cho, "Endorsement policy for industrial internet of thing with private blockchain," in *2021 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 449–452, IEEE, 2021.

[164] T. Shimosawa, T. Sato, and S. Oshima, "Bcverifier: a tool to verify hyperledger fabric ledgers," in *2020 IEEE International Conference on Blockchain (Blockchain)*, pp. 291–299, IEEE, 2020.

[165] S. Rouhani and R. Deters, "Data trust framework using blockchain technology and adaptive transaction validation," *IEEE Access*, vol. 9, pp. 90379–90391, 2021.

[166] C. Harris, "Performance evaluation of ordering services and endorsement policies in hyperledger fabric," in *2023 33rd Conference of Open Innovations Association (FRUCT)*, pp. 63–69, IEEE, 2023.

[167] C. Wang and X. Chu, "Performance characterization and bottleneck analysis of hyperledger fabric," in *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*, pp. 1281–1286, IEEE, 2020.

[168] R. Saranya and A. Murugan, "A hyperledger fabric-based system framework for healthcare data management," in *2023 7th International Conference on Computing Methodologies and Communication (ICCMC)*, pp. 552–556, IEEE, 2023.