POLITECNICO DI TORINO

DEPARTMENT OF CONTROL AND COMPUTER ENGINEERING

Master of Science in Computer Engineering

Master Degree Thesis

BlockChain Technology and Smart Contracts to share information

Supervisor
Prof. Guido Perboli

Candidate
Jacopo Maggio

Internship Tutors
BLUE Reply S.r.l.

Academic year 2024 – 2025

Abstract

Blockchain technology has emerged as a transformative tool for industries seeking enhanced transparency, security, and trust in multi-party transactions. Beyond its origins in cryptocurrencies, blockchain (particularly through smart contracts) enables automated, tamper-resistant agreements that minimize the need for central intermediaries. However, practical integration of blockchain into enterprise environments poses challenges related to scalability, regulatory compliance, data privacy, and interoperability with legacy systems.

This thesis focuses on Hyperledger Fabric, a leading permissioned blockchain framework designed to meet the stringent requirements of industrial applications. We provide a systematic analysis of Fabric's architecture, emphasizing its modular structure, robust identity management, and support for customizable smart contracts (chaincode). Through detailed case studies in railcar sharing and shared procurement, we investigate how Fabric can improve operational efficiency, data integrity, and auditability while enabling secure information sharing among stakeholders.

Experimental evaluation demonstrates that Hyperledger Fabric achieves strong performance and resilience, satisfying the compliance and confidentiality standards necessary for regulated sectors. We also address the technical and organisational barriers industries face during blockchain adoption and propose practical risk mitigation strategies. Finally, the thesis concludes by examining the economic and regulatory implications of permissioned blockchains, suggesting pathways for future research and industrial innovation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Blockchain technology extends beyond cryptocurrency, enhancing transparency, authenticity, and resilience across various industries. The increasing adoption of distributed ledger technology (DLT) necessitates enterprise-grade solutions that comply with stringent security and regulatory requirements. Hyperledger Fabric emerges as a notable permissioned blockchain framework, employing a consortium model that allows precise control over membership and transaction validation. Its combination of cryptography and flexible structure can streamline corporate processes and address persistent trust issues. Consequently, understanding the role of Fabric has become crucial in contemporary industrial discourse and strategy development.

Still, there are questions about how blockchain can fit into existing systems, meet data protection rules, and remain reliable in demanding environments. To answer these questions, the Chapter 2 discusses the basics of blockchain technology.

1.1 Background and Motivation

Blockchain's origins are often linked to Bitcoin, where a decentralised ledger demonstrated the possibility of secure peer-to-peer transactions. After that, researchers explored other ways to use blockchain in many areas, like finance, healthcare, public record management, and manufacturing. This wide interest shows blockchain's flexibility as a tamper-evident store for important data, able to handle transaction loads and use cryptography to build trust among participants. Many industries now want solutions that fit smoothly with current systems but still meet performance and regulatory requirements.

Industry's growing demands and shifting regulations create the need to investigate blockchain in more detail. Many organisations face challenges connecting old databases with decentralised setups, so they need frameworks that can bridge these gaps. Meanwhile, frequent data breaches show the need for strong security measures. As a result, businesses want systematic ways to deploy blockchain while meeting legal and regulatory needs. This research considers Hyperledger Fabric a leading platform that can meet these requirements and fit well with enterprise goals. The next subsections discuss the underlying reasons for this detailed study and the innovation behind it.

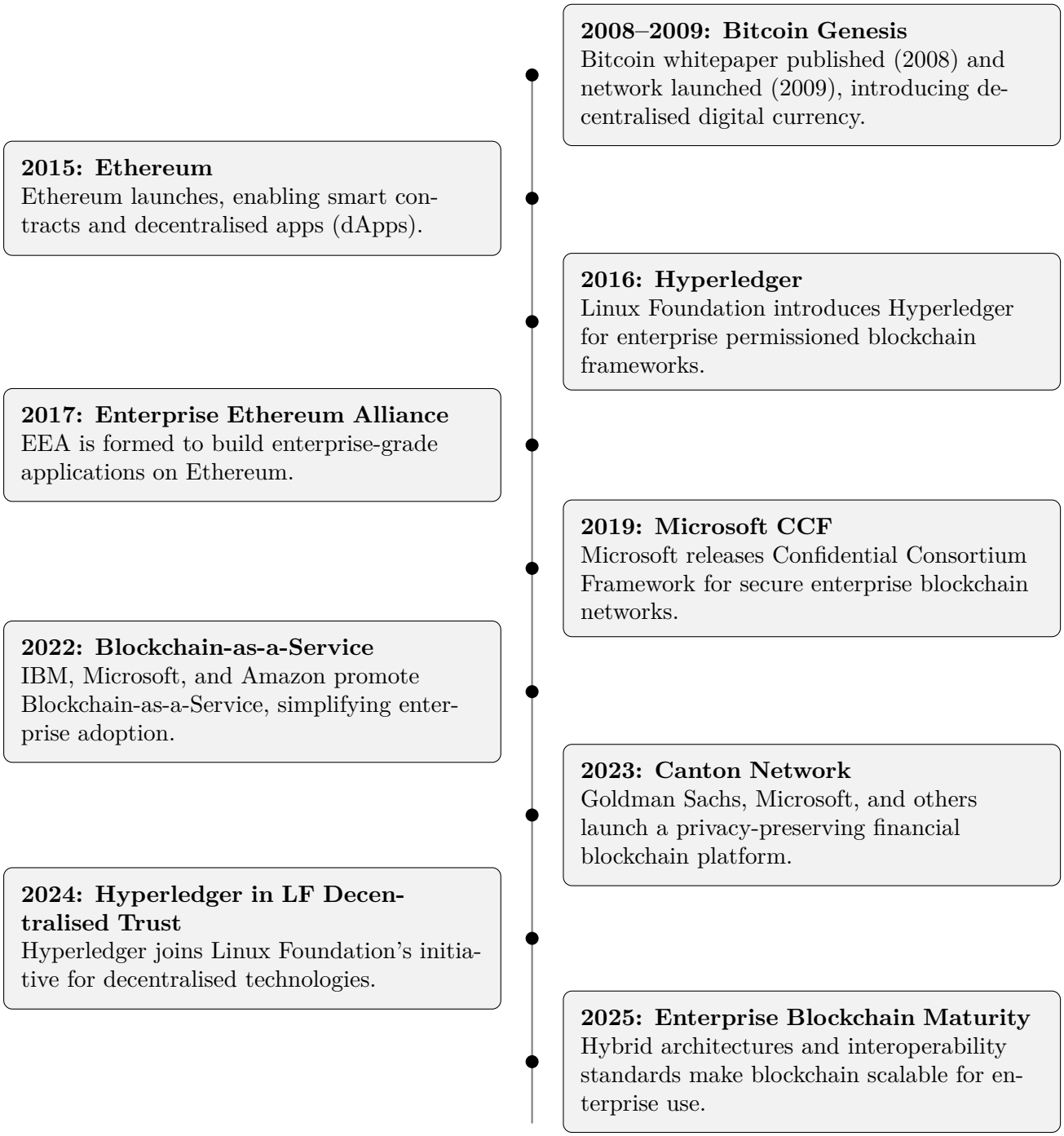


Figure 1.1: Evolution of blockchain technology from Bitcoin to enterprise permissioned frameworks

1.1.1 Context of Blockchain Technologies

To understand blockchain, we need to see how it moved from cryptocurrency to a technology used by various industries. After Bitcoin showed that trustless transactions could work, many new protocols appeared. One of them is Ethereum, which added smart contract features. Each platform has its own consensus system—like Proof of Work, Proof of Stake, or Byzantine Fault Tolerance. These systems affect how fast transactions can

be processed, how much energy is used, and how secure the network is. They also impact how practical the blockchain is for supply chains, financial services, and government records. Research from academia and industry shows that for blockchain to be adopted more widely, it must handle issues like throughput limits, data privacy, and regulatory uncertainty on a global scale.

1.1.2 Motivation for Exploring Hyperledger Fabric in Industrial Applications

Choosing a blockchain for industrial use means meeting strict performance requirements, dealing with many stakeholders, and following various compliance rules. Hyperledger Fabric tackles these needs by using a permissioned model, where only approved participants can join, creating trust in a closed environment. Unlike public blockchains that depend on anonymous consensus, Fabric supports modular consensus protocols that can handle different scenarios. It also splits transaction ordering, execution, and validation into separate stages. This approach supports high throughput without giving up data confidentiality. Because of this precise level of control, Fabric is especially appealing for large global companies with complex supply chains or in heavily regulated areas.

In industries like finance, healthcare, and insurance, regulatory compliance is critical because data handling and privacy are tightly regulated. Hyperledger Fabric supports private data collections and flexible membership services that help businesses meet strict legal requirements. These features also reduce worries linked to public ledgers, especially when dealing with trade secrets or personal data. As a result, companies are driven to explore Fabric for both operational benefits and legal needs. This research looks at how well Fabric can adapt in these contexts, while also offering strong audit trails and controlled information sharing and governance.

1.1.3 Regulatory Evolution and Global Policy Trends

In recent years, regulations for blockchain have changed quickly in Europe and around the world. A major development is the European Markets in Crypto-Assets Regulation (MiCA), passed in 2023 and taking effect in 2024. MiCA is the first EU-wide set of rules for digital assets, service providers, and market infrastructures. It focuses on transparency, governance, user protection, and risk management, and it directly affects companies, banks, and providers who want to use blockchain and smart contracts in industry and finance [13].

At the same time, experts around the world are stressing the importance of coordinated public policy and more attention to ethics and security in distributed ledger systems. A recent white paper from the Center for Information Technology Policy at Princeton University [7] says that as blockchain technology matures, it needs the right regulatory tools to ensure interoperability, privacy, and resilience, especially in sensitive and highly regulated areas.

Because of this, blending technological progress with regulatory rules is crucial for industries adopting blockchain. This creates new design challenges for system architecture (like managing personal data, respecting the right to erasure, and ensuring auditability) and for governance of permissioned networks and consortia.

1.2 Problem Statement

The path to adopting blockchain in industry is complex, even for strong platforms like Hyperledger Fabric. Issues with scalability, interoperability, and data security often clash with how organisations are already set up. Some of the biggest barriers are:

- **Scalability:** In decentralised setups, consensus and cryptographic checks can cause slowdowns, limiting throughput and raising latency.
- **Interoperability with Legacy Infrastructures:** Building a blockchain network into existing enterprise systems often needs major redesign or dependable middleware solutions.
- **Data Confidentiality and Security:** Keeping sensitive industrial data secure on a distributed ledger is challenging and often needs fine-tuned permission models and advanced encryption.
- **Regulatory Compliance:** Various legal rules (like GDPR) introduce questions about how to handle, store, or remove data on an immutable ledger.
- **Smart Contract Vulnerabilities:** Errors or bugs in chaincode can disrupt important operations or reveal proprietary information.
- **Governance and Coordination:** Networks with multiple organisations must agree on membership rules, smart contract changes, and continuous risk management to maintain trust.

These challenges highlight the gap between blockchain's potential and the practical realities of large, regulated industries. The next section sets out the research objectives that aim to systematically address these issues and show how Hyperledger Fabric could help solve major adoption problems.

Aside from scalability and interoperability, building secure but efficient smart contracts is also a big challenge. Automatic code must be carefully designed to handle permissions, data confidentiality, and potential weaknesses. Industrial settings often have vital processes where contract bugs can disrupt supply chains or compromise sensitive data. In addition, industry-specific regulations (for example, in finance or pharmaceuticals) need strict governance. Proper risk management is essential because attacks or code mistakes can lead to big financial losses or damage reputations. This thesis examines such connected issues and explains why Hyperledger Fabric is a strong candidate for addressing them effectively.

1.3 Research Objectives and Contributions

This study aims to see how a permissioned blockchain can meet the needs of industries that rely on verifiable transactions and secure data. By focusing on Hyperledger Fabric, we want to find out if its flexible structure can handle complex business logic, protect confidentiality, and maintain good performance in real scenarios. We also look at how Fabric's customizable smart contracts might boost transparency and audits, increasing

accountability among partners. The following subsections outline each research objective and key contribution.

Table 1.1: Research Objectives

Objective	Contribution	Metrics	Chapters
Data security	Use Fabric identity/private data features to protect info	Data privacy level; regulatory alignment	Ch. 4, 5
Performance	Empirical results on throughput, latency, resources	TPS, avg. confirmation time	Ch. 4, 5
Vulnerability analysis	Identify key risks and propose mitigation	Risks identified; mitigations validated	Ch. 4, 5, 6
Compliance & integration	Fabric integration with legacy systems under regulations	Successful legacy integration; compliance proof	Ch. 5, 6

We start by studying how blockchain can protect sensitive industrial data with strong encryption and permission controls. Next, we build a Hyperledger Fabric prototype to measure transaction throughput, latency, and resource use under different workloads, checking its performance. Then, we assess potential security dangers, from consensus manipulation to smart contract exploits, and outline possible countermeasures. Finally, we look at strategies for adhering to regulations when linking older systems, including tools for handling identities and audits. Together, these goals reveal whether Hyperledger Fabric can support critical tasks, paving the way for broader innovation in diverse industrial settings around the globe.

1.3.1 Research Design and Approach

We use a mixed methods approach for this study, blending theory and structured experiments. We start with a broad literature review to understand both blockchain basics and key industry challenges. Based on this knowledge, we design our Hyperledger Fabric prototype in steps to check its features under real-world conditions. We collect data from performance tests, security checks, and scenario-based evaluations, so we can view the results from different perspectives. Importantly, we focus on reproducibility, so others can replicate or build on our work. This careful process adds to both the academic and real-world value of the study.

1.3.2 Data Collection and Tools

This thesis uses data from several sources. First, we rely on simulations to set up controlled environments where we can measure transaction speed and network latency. In some cases, real datasets from industry partners add to these tests, reflecting actual challenges like changing transaction sizes or busy periods. For simulations we also use benchmarking tools, such as Hyperledger Caliper, to compare performance across various setups. We record logs and network metrics for our quantitative analysis, and combine

them with feedback from users for qualitative insights. This blend of data helps us draw strong, evidence-based conclusions that improve reliability.

1.3.3 Evaluation Metrics and Experimental Setup

When assessing a blockchain framework, we look at metrics that show both technical strength and real-world practicality. For instance, transaction throughput shows how many operations the network handles per given time, and latency measures how long it takes for a transaction to finish. We also check CPU load, memory use, and network bandwidth to see how efficient the system is. Security and resilience testing includes running stress tests and looking for weaknesses by simulating attacks or network breakdowns. Our experimental setup uses different node configurations, starting with small test networks and moving to larger ones, so our findings can apply to a wide range of industries.

1.3.4 Risk Management

Effective risk management is essential for trusting blockchain in industrial environments. Although smart contracts are powerful, coding mistakes can let unauthorised transactions happen or expose data. To prevent this, we conduct regular security checks and code reviews to find and fix weaknesses before going live. We also run scenario tests to see how the system handles node crashes, malicious nodes, or sudden traffic surges. Beyond that, our risk mitigation strategy involves organisational policies, such as strict access control. Altogether, these steps make the system more resilient and greatly reduce the chance of harmful disruptions.

1.4 Scope and Limitations

While this study offers useful insights into blockchain's industrial potential, certain real-world factors limit how widely we can apply these findings. These constraints include the size of our experiments, available resources, the fast pace of technology changes, and varying regulations. Awareness of these limits is important for accurately applying the results and avoiding assumptions that don't fit every scenario.

- **Experimental Scale:** In test networks, we try to mimic real conditions, but pilot setups can't fully reflect the complexity and interconnectedness of large industrial systems.
- **Time and Resource Constraints:** The study had a limited budget and timeframe, so we couldn't explore all workloads, network layouts, or the most extreme scenarios.
- **Evolving Blockchain Landscape:** Consensus methods, SDKs, and security layers change quickly, so our findings may lose relevance as the field evolves.
- **Organisational Readiness:** Enterprises differ in their digital readiness, existing systems, and governance styles, all of which influence how easily they can adopt blockchain.

- **Regulatory Variations:** Different regions have different rules on data storage, contract enforcement, and audit practices, which can shape how a permissioned blockchain is used and deployed.

Key Real-World Constraints Affecting Generalizability

- *Time and Scope Boundaries:* The length of the study and the scale of the setup limited multi-stage simulations and long-term resilience testing.
- *Infrastructure Simplification:* We used simpler ordering and peer setups, which did not fully reflect enterprise-grade high-availability designs.
- *Contextual Complexity:* The chosen use cases may not capture all policy layers, vendor relationships, or audit processes seen in real consortium projects.
- *Regulatory Uncertainty:* Changing rules about data sovereignty mean some of our assumptions might not hold for long.

1.5 Thesis Organisation (Outline)

This thesis is organised to gradually show the main ideas behind blockchain, especially smart contracts, and how they're deployed for industrial use with Hyperledger Fabric. Each chapter builds on the previous one, starting with theory, moving to real implementation, and ending with practical examples. The structure is outlined below:

- **Chapter 2 – Extended Literature Review:** A comprehensive review of current research on blockchain, starting with basics like distributed ledgers, cryptography, and consensus methods. It compares public, private, and consortium blockchains, especially focusing on enterprise needs. The chapter also looks at how blockchain is being used in different sectors, examines the ethical and regulatory environment, and points out gaps in current knowledge that this thesis aims to address.
- **Chapter 3 – Smart Contracts:** This chapter covers smart contracts in detail. It defines what they are, how they work, and when they apply. It also compares them to traditional legal contracts and explores how they are enforced. There's a review of common uses in various industries, along with technical factors like programming languages, deployment, and data sources. The chapter also addresses challenges such as immutable code, security weaknesses, and keeping versions up to date.
- **Chapter 4 – Hyperledger Fabric:** This chapter describes Hyperledger Fabric as a flexible blockchain platform designed for businesses. It explains Fabric's architecture, including its main parts—peers, orderers, channels, endorsement policies, and chaincode lifecycles. It's especially interested in Fabric's privacy, scalability, and identity management features using Certificate Authorities. The chapter also covers performance benchmarks, security measures, and compares Fabric with other enterprise blockchain solutions.
- **Chapter 5 – Case Study:** This chapter shows the practical side of the thesis with two case studies that integrate Hyperledger Fabric into existing industrial

workflows. The first case covers a blockchain solution for Railcar Sharing, focusing on traceability, accountability, and asset usage. The second focuses on Shared Procurement, emphasizing gains like clearer audits, transparency, and less friction. For each example, the chapter explains the business context, architecture, smart contract design, and the results of deployment, integration, and performance.

- **Chapter 6 – Conclusions and Future Work:** The final chapter sums up the key outcomes and relates them to the main research questions. It looks at what using blockchain and smart contracts means for industry, reviews the strengths and weaknesses of the solutions presented, and suggests possible improvements or directions for more study. This includes expanding the use cases, reinforcing security models, and finding ways to work with other blockchain platforms.

Overall, this layout seeks to guide through the essential theories of blockchain into its real-world applications, spotlighting the potential for using smart contracts in enterprise environments.

Chapter 2

Extended Literature Review

2.1 Introduction

In this chapter, we present the core ideas required to evaluate Hyperledger Fabric in enterprise settings. We start by discussing fundamental concepts like cryptography and consensus design, then move on to blockchain typologies and permission models. We also examine broader issues such as platform comparisons, regulations, and ethical considerations to show how blockchain is adopted and challenged in real-world contexts. The following roadmap outlines the main themes of this literature review:

- **Cryptographic Foundations:** Basic principles of hashing, digital signatures, and Merkle trees.
- **Blockchain Classifications:** Public, private, and consortium models, and their operational distinctions.
- **Consensus Mechanisms:** Overview of PoW, PoS, RAFT, PBFT and their trade-offs.
- **Permissioned vs. Permissionless Networks:** Implications for Governance, Scalability, and Trust.
- **Legal and Ethical Dimensions:** Compliance requirements, data sovereignty, and evolving regulatory frameworks.
- **Platform Comparisons:** Technical and strategic differences between Hyperledger Fabric, Ethereum, Corda, and others.
- **Use Cases and Industry Adoption:** Documented deployments and insights from the recent academic literature.

We conclude by summarising key lessons and identifying gaps in the literature, setting the stage for the methodological and empirical investigations in the following chapters.

2.2 Overview of Blockchain Technology

Blockchain serves as a distributed and tamper-resistant ledger that is maintained by a network of participants. Following the launch of Bitcoin, the technology has matured

to handle broader computational tasks, including smart contracts and other advanced features. This section covers the historical background of blockchain, its conceptual design, and the attributes that make it potentially valuable across different sectors.

2.2.1 Blockchain Fundamentals and Cryptography

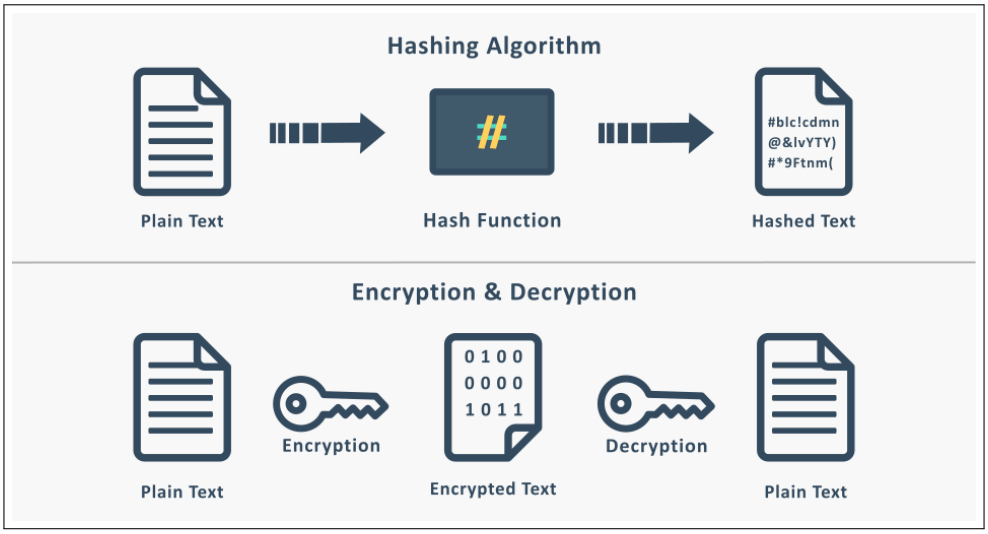


Figure 2.1: Comparison between hashing and encryption. Source: [1].

At its core, blockchain relies on cryptographic methods to maintain data security and transaction integrity. Two major techniques, hash functions and public-key cryptography, enable digital signatures and protect the chain from tampering. Hash functions (e.g., SHA-256) create a fixed-length output from a variable-length input, making it easier to confirm that each block's data remains unchanged. Public-key cryptography supports token issuance and validates transactions, as node operators sign transactions with private keys.

Because each block incorporates the hash of the previous block, the entire chain becomes resistant to modification. Altering any single block makes subsequent hashes invalid, alerting others in the network to the discrepancy. Together with decentralisation, this structure provides the "trust minimisation" that is often highlighted as a defining advantage of blockchain.

2.2.2 Structural Components of a Blockchain

A blockchain comprises distributed nodes (each storing the full ledger), blocks that hold verified transactions, and a consensus protocol for transaction validation. Typically, blocks include:

- **Block Header:** Includes the block version, previous block hash, Merkle root, timestamp, and unique nonce.
- **Merkle Root:** Derived from a Merkle tree to summarise all transactions in a block, to aid in efficient verification.
- **Transaction List:** A collection of valid transactions appended to the ledger.

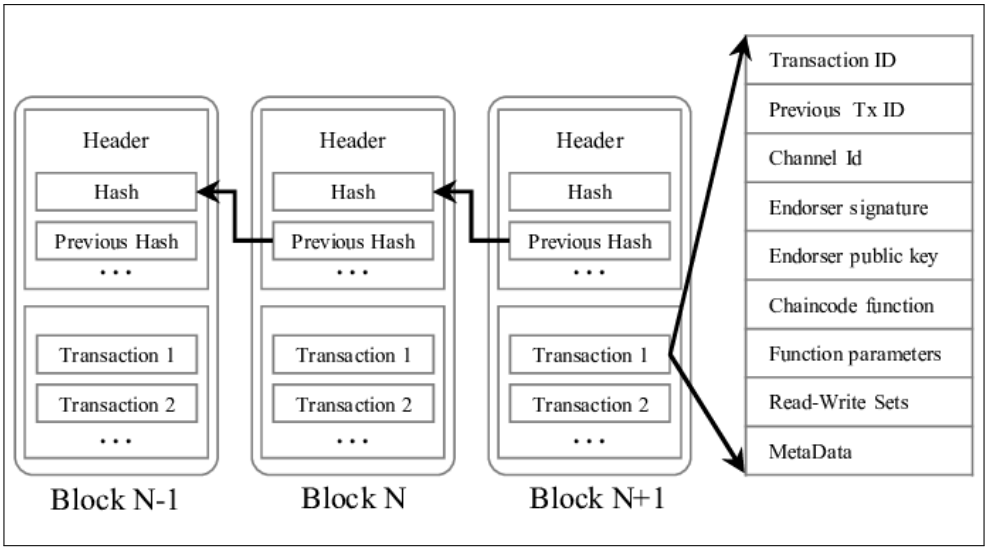


Figure 2.2: The structure of blocks (take Hyperledger Fabric as an example). Source: [23]

Nodes broadcast new transactions to peers, and they collectively follow a consensus procedure to validate which transactions are legitimate. Once a block is added, the updated ledger is shared throughout the network. This distributed approach promotes transparency, as all participants can view the transaction history, and increases resilience by removing single points of failure.

2.2.3 Consensus Mechanisms and Core Characteristics

Consensus mechanisms specify how nodes agree on valid transactions and preserve a consistent ledger. Common approaches include:

- **Proof of Work (PoW):** Nodes (miners) compete to solve a resource-intensive puzzle, earning the right to add the next block. Although secure, PoW is criticised for its high energy consumption.
- **Proof of Stake (PoS):** Validation rights are proportional to the stake (cryptocurrency holdings) of each node. PoS aims to reduce computational overhead relative to PoW .
- **Practical Byzantine Fault Tolerance (PBFT):** Particularly relevant in permissioned settings, PBFT requires nodes to communicate extensively, ensuring malicious or faulty nodes do not undermine the system, provided a majority are honest [6].

Core characteristics of blockchain-based systems include:

- **Decentralisation:** Removes single points of failure and fosters trustless environments.
- **Immutability:** Altering historical records is computationally unrealistic due to cryptographic links between blocks.
- **Transparency:** Copy of the ledger is maintained by all participating nodes.
- **Security:** Resilient to certain classes of cyberattacks; however, 51% attacks and private-key theft are notable vulnerabilities.

2.3 Types of Blockchains

Blockchains are commonly categorised by their level of openness and governance models. Although these groupings can overlap and depend on context, they largely revolve around whether networks are permissioned and how participants interact.

2.3.1 Private, Consortium, and Public Blockchains

Public blockchains, such as Bitcoin and Ethereum, allow anyone to read and write data on the ledger, making them widely accessible. They use economic rewards, like transaction fees and block rewards, to support node participation and consensus [27].

Private blockchains limit network membership to a single organisation or a closely controlled set of parties, making them suitable for applications that require strong secrecy. Under this structure, a single entity has more control over membership, transaction validation, and governance, often opting for more centralised consensus to achieve greater efficiency.

Consortium blockchains balance these models by splitting governance among a smaller group of entities. This approach is frequently adopted in scenarios such as supply chain consortia or financial settlement systems. Consortium blockchains often utilize consensus mechanisms like PBFT or Raft, which allow quick finality while assuming a certain level of trust among approved participants.

Table 2.1: Comparison of Blockchain Types

Attribute	**Public**	**Consortium**	**Private**
Governance	Decentralised; open to all	Shared among select organisations	Centralised within one entity
Consensus Style	PoW, PoS, etc. (energy-intensive)	PBFT, Raft (efficient, trusted nodes)	Centralised or delegated mechanisms
Performance	Lower throughput, high latency	Moderate to high throughput, fast finality	High throughput, low latency
Typical Use Case	Cryptocurrency, open dApps	Inter-organisational collaboration	Internal audit, enterprise automation

2.3.2 Permissioned vs. Permissionless

Blockchain networks can also be labelled *permissioned* or *permissionless*, based on how participants are verified and how transactions are submitted. In permissionless systems (e.g., Bitcoin), node participation depends mainly on consensus rules, making them open to anyone. By contrast, permissioned systems (e.g., Hyperledger Fabric) require new nodes to be explicitly approved by a central authority or consortium.

Permissioned blockchains can provide improved scalability, faster transaction speeds, and greater data protection, since they control membership and can implement detailed

access rules [3]. However, this gain in efficiency and control may reduce openness and raise questions on whether such systems stray from blockchain's original decentralised ideals.

Table 2.2: Comparison of Permissioned vs. Permissionless Blockchains

Criteria	**Permissionless**	**Permissioned**
Anonymity	High	Limited / None
Trust among participants	Not required	Required / Vetted
Membership control	Open to all	Restricted
Throughput / Scalability	Limited	Higher
Data privacy & access	Public	Fine-grained
Governance decentralisation	High	Potentially centralised

2.4 Regulatory, Ethical, and Legal Considerations

The rapid spread of blockchain-based applications has outpaced existing legal frameworks, creating regulatory uncertainty and sparking ethical debates. Key considerations include:

1. **Data Privacy and GDPR Compliance:** Blockchains store transaction data in an immutable ledger, potentially conflicting with regulations requiring data modification or deletion (e.g., "the right to be forgotten" in GDPR).

   GDPR vs. Blockchain

   The General Data Protection Regulation (GDPR) grants EU citizens the right to request data erasure. This is at odds with blockchain's immutability principle. Solutions like off-chain storage, encryption, and hash anonymisation have been proposed, but legal consensus on their adequacy is still emerging.

2. **Cross-Jurisdictional Challenges:** Laws vary across regions, leading to legal uncertainties related to digital asset classification, taxation, and consumer protection.

   Sarbanes-Oxley (SOX) and Auditable Records

   The Sarbanes-Oxley Act mandates traceable and tamper-evident financial record-keeping. Blockchain's audit trails could support SOX compliance—if implemented with proper access controls and validation layers—but questions remain on how courts interpret blockchain logs.

3. **Smart Contract Liability and Enforcement:** Automated contracts executing on a blockchain raise questions about the enforceability of code-based agreements and liability in the event of contract failure or code vulnerabilities.

4. **Ethical Considerations:** Transparency and immutability can conflict with concerns about privacy and anonymity. Unequal access to cryptographic tools or disparate regulatory acceptance can exacerbate global inequalities.

HIPAA and Medical Blockchain Records

Under the Health Insurance Portability and Accountability Act (HIPAA), healthcare providers must protect patient privacy and ensure data confidentiality. Storing sensitive medical data directly on-chain risks noncompliance. Hybrid architectures with encrypted off-chain storage and on-chain access control mechanisms offer a potential compromise.

In spite of these complexities, many policymakers are introducing measures like regulatory sandboxes to encourage innovation while still protecting consumers. Ethical frameworks typically focus on designing blockchain systems with proportionality, accountability, and inclusion in mind.

2.5 Related Work and Existing Solutions

This section explores various blockchain platforms, key insights from academic research, and real-world applications. We begin by comparing major platforms, then review their documented use cases in industries such as finance, supply chain, and healthcare.

2.5.1 Comparison of Blockchain Platforms

Multiple blockchain platforms provide different rationale, consensus methods, and governance approaches. Table 2.3 shows a high-level overview of common platforms [25].

Table 2.3: High-Level Comparison of Selected Blockchain Platforms (Example Data)

Platform	**Consensus Mechanism**	**Type**	**Key Features**
Bitcoin	Proof of Work (PoW)	Public, Permissionless	Simple scripting, robust security
Ethereum	PoS (from PoW)	Public, Permissionless	Smart contracts, large developer ecosystem
Hyperledger Fabric	PBFT / CFT	Permissioned	Modular, private data collections
Corda	Notary-based	Permissioned	Focus on financial workflows
Quorum	PoA / IBFT	Permissioned	Ethereum-based, privacy features

Hyperledger Fabric, the focal point of this thesis, is notable for its permissioned architecture, offering detailed membership control and customizable consensus. Its flexible ledger layout and support for private data collections suit heavily regulated industries that handle sensitive data [3].

Table 2.4: Community and Ecosystem Maturity of Selected Platforms

Platform	**Community & Ecosystem Maturity**
Bitcoin	Very mature; global adoption and conservative development culture
Ethereum	Highly active; large contributor base, diverse dApp ecosystem
Hyperledger Fabric	Strong enterprise support; modular, actively developed
Corda	Moderate; backed by financial institutions, less open-source traction
Quorum	Niche; supported by ConsenSys, Ethereum-compatible tooling

2.5.2 Industry Adoption and Case Studies

Nowadays many sectors (such as supply chain, healthcare, and finance) are integrating blockchain to enhance auditability and reduce data tampering. For instance:

- **Supply Chain:** Companies like Walmart and IBM have utilised Hyperledger Fabric to track produce, resulting in quicker traceability and contamination detection .

- **Healthcare:** Blockchain-based solutions are being piloted to manage patient data continuity, reduce insurance fraud, and ensure secure sharing of sensitive medical records across multiple platforms and providers. Notably, pilot programs leveraging Hyperledger Fabric have indicated faster data retrieval and improved compliance with patient privacy regulations, although challenges remain regarding large-scale interoperability and cost-effectiveness in healthcare settings.

- **Finance:** Financial institutions are exploring blockchains for cross-border payments and settlement, reducing operational overhead and improving transaction speed. Research indicates that permissioned blockchains like Corda and Hyperledger Fabric can offer robust privacy models, making them attractive for interbank communications and regulatory reporting .

- **Government Administration:** Various government agencies are testing blockchain for identity management, land registry, and public procurement. Such applications aim to minimize corruption, mitigate bureaucratic inefficiencies, and promote trust through transparent record-keeping systems [25].

Case Study: Walmart–IBM Food Trust

Walmart collaborated with IBM to deploy a Hyperledger Fabric-based platform for tracing food origins. The solution reduced produce traceability from 7 days to 2.2 seconds, boosting food safety and operational transparency. It demonstrated how blockchain could streamline supply chains while satisfying regulatory requirements.

Case Study: Maersk–IBM TradeLens

TradeLens, a blockchain shipping platform co-developed by Maersk and IBM, aims to digitise global trade documentation. Despite onboarding more than 150 participants, including port authorities and customs bodies, its adoption slowed due to reluctance among key competitors and limited network effects. The project was cancelled in 2023, illustrating the importance of ecosystem cooperation.

Alongside these initial successes, studies report issues like limited throughput and high latency that undermine the full potential of enterprise-level blockchain solutions. Nonetheless, as industries push forward with digital transformation, blockchain testing and deployment continue to gain momentum.

2.6 Conclusion

This extended literature review explored essential aspects of blockchain technology, including its underlying structures, consensus methods, and classifications (public, private, consortium, permissioned, and permissionless). It highlighted how cryptographic safeguards and various consensus protocols address different requirements for security, performance, and regulatory compliance.

We also looked at regulatory, ethical, and legal concerns, recognising that immutability and decentralisation can clash with ever-evolving legal standards. Turning to practical implementations—especially Hyperledger Fabric—underscored how permissioned architectures can address concerns about data privacy and throughput. Yet, real-world applications also reveal ongoing challenges like governance and scalability.

Key Takeaways:

- **Immutability and Regulation:** The permanence of blockchain data creates challenges to complying with evolving legal requirements such as the GDPR.
- **Permissioned Systems for Enterprise Needs:** Platforms such as Hyperledger Fabric enable fine-grained access control, high throughput, and compliance, making them suitable for regulated industrial environments.
- **Consensus Mechanism Trade-Offs:** Different algorithms offer varying balances between security, scalability, and energy efficiency—no one-size-fits-all solution exists.
- **Real-World Adoption is Growing—But Fragmented:** Industry adoption is expanding, but interoperability, governance, and ecosystem maturity remain barriers to full-scale deployment.

These insights establish the basis for the investigations in the following chapters. In particular, Chapter 5 describes the methodology applied in defining blockchain deployment and smart contract engineering in industrial use cases. This approach leans on the theoretical and technical groundwork presented here, guiding a detailed study of Hyperledger Fabric's ability to satisfy enterprise-level needs in practice.

Chapter 3

Smart Contracts

Smart contracts have emerged as a transformative concept within blockchain ecosystems, enabling agreements to self-enforce without the need for a central authority. Building on the foundational ideas introduced in previous chapters, this chapter explores both the theoretical underpinnings and practical implementations of smart contracts.

We begin with a comprehensive introduction, illustrating how certain contractual functions can be automated through code-based rules. Following this, we delve into the operational mechanics of smart contracts, examining their design principles and how they relate to traditional legal frameworks.

Further discussion addresses the potential of smart contracts to reduce trust dependencies, enhance security, and improve transparency. At the same time, we critically evaluate the technical, regulatory, and organisational challenges they pose. Key issues such as scalability limitations, code vulnerabilities, and the complexities of legal enforceability are examined to provide a nuanced understanding of the technology's capabilities and constraints.

This chapter connect theoretical ideas with real-world applications, providing a balanced perspective that can guide policymakers, global businesses, and researchers.

3.1 Introduction to Smart Contracts

Smart contracts mark a major shift in how agreements can be created and carried out, reflecting the decentralised nature of blockchain. In the past, the idea of algorithmic enforcement came up in theoretical computer science, but it was not until the last decade that it really merged with decentralised consensus: modern smart contracts rely heavily on cryptographic structures and distributed ledgers like Bitcoin and Ethereum.

Traditional contracts depend on legal texts and outside mediators, but smart contracts work in decentralised systems, using code to serve as both mediator and judge of contract conditions. Because smart contracts run automatically on a blockchain, they produce results that are tamper-evident. As a result, participants benefit from lower transaction costs, less need for trust, and built-in audit trails. However, these features can create difficulties when trying to fit smart contracts into existing regulatory and organisational frameworks, and they may demand specialised oversight.

To understand why smart contracts matter, we should place them in the broader story of protocols that aim to reduce reliance on trust. In the past, parties forming contracts had to handle high coordination, enforcement, and oversight expenses, especially in complicated international deals. By using cryptography and decentralised validation, smart

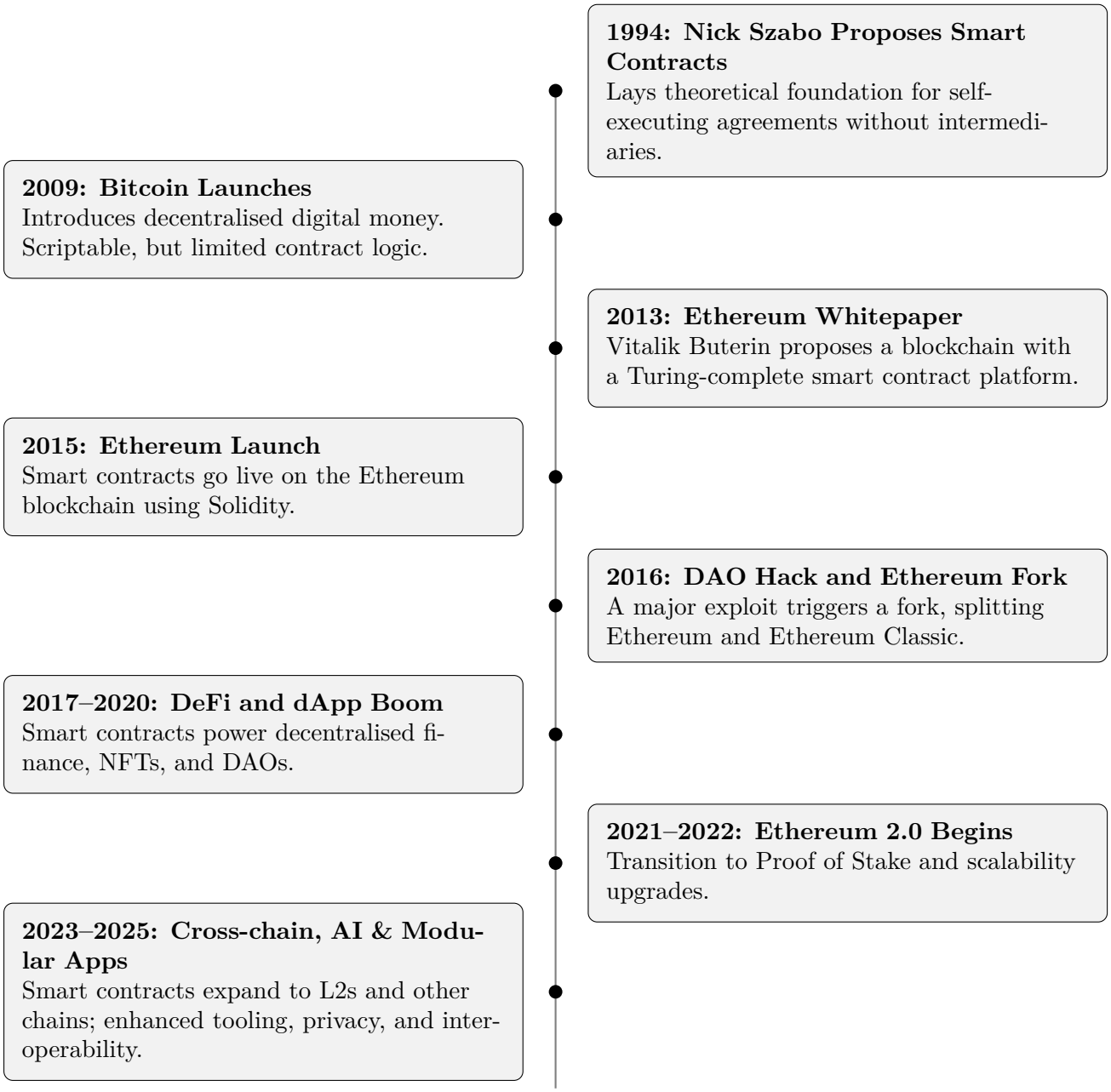


Figure 3.1: Timeline of Smart Contract Evolution

contracts help cut this friction. This cost-saving argument has led many sectors, including finance and supply chain, to explore smart contracts to reduce disagreements and lower operational costs. Still, we must remember that they do not solve every problem. They rely on strong coding practices, dependable consensus, and accurate external data. If these fail, contracts can fail as well. These demands can open up new risks if the contract logic or the blockchain platform is breached. For this reason, it's crucial to examine how software engineering best practices mesh with decentralised governance for a full review of this technology.

In Chapter 1, we covered the broader context of blockchain technologies, and Chapter 2 went deeper into core ideas while examining external factors like regulations. In keeping with that, this introduction treats smart contracts as both a separate yet integral part

of the broader blockchain picture. By exploring motives, historical background, and known complexities, we set up the perspective for the following sections to examine their main operating principles, compare them to standard contracts, and see how they might reshape business and government.

From here on, we take a broad view of smart contracts, covering the technology itself, plus the social, organisational, and legal factors that matter for real deployments. By carefully looking at the connections among code, consensus, enforcement, and real-world challenges, we offer a framework for decision makers and those hesitant to adopt new systems without strong assurances. Ultimately, this introduction sets the stage to investigate how smart contracts operate and how they can drive new institutional opportunities, significant changes, and expansions across industries, highlighting the broad future potential.

3.2 How Smart Contracts Work

To see how smart contracts actually work, we need to look at how their software code runs on decentralised ledger systems to handle contracts automatically. Broadly, their process includes several steps:

1. the **initialisation phase**, when the code goes onto the blockchain;
2. the **triggers** that activate specific functions;
3. the **updates** to on-chain states and any off-chain signals.

When a contract is deployed, it's accessible to network participants who can call its methods, as allowed by protocol permissions and consensus rules. By running deterministic code, these contracts guarantee outcomes that follow their predefined logic, offering consistency and predictability. Also, because the blockchain ledger is immutable, there is always a record of interactions that stakeholders can review. Meanwhile, consensus mechanisms like Proof of Work, Proof of Stake, or PBFT confirm these transactions, reducing the need for a single authority. But this setup also has issues like limited computing resources, transaction fees, and timing constraints. Developers have to balance how much functionality they include with the need for efficiency. The way the code-driven events work with distributed validation is exciting but requires a solid strategy.

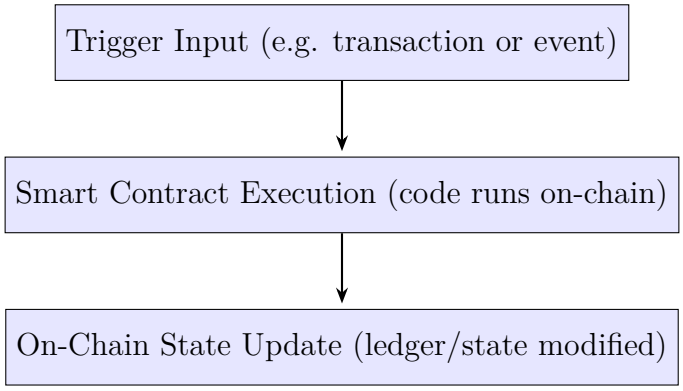


Figure 3.2: Smart Contract Flow: Trigger → Execution → On-Chain State Update

A key feature of smart contracts is that they can use external or off-chain data, typically provided by oracles, to expand their capabilities beyond what's on the blockchain.

Oracles: Bridge or Bottleneck?

Oracles serve as bridges between blockchains and the external world, injecting real-world data (e.g., exchange rates, weather conditions, IoT sensor outputs) into smart contract logic. However, they also create vulnerabilities such as:

- **Single-point-of-failure:** A centralised oracle can be manipulated or go offline.
- **Sybil attacks:** Malicious actors can pose as multiple oracles to distort consensus.
- **Data tampering:** Intercepted or falsified input may corrupt contract behaviour.
- **Latency and inconsistency:** Timely and synchronised data delivery is critical but not guaranteed.

Mitigations include decentralised oracle networks (e.g. Chainlink), cryptographic proofs, and fallback data sources.

Used as bridges to the outside world, oracles (whether software or hardware) supply things like exchange rates, weather data, or sensor inputs to the contract. But depending on oracles introduces new trust challenges, since malicious or incorrect oracles can corrupt the contract. This makes data validation and fail-safes essential. In addition, since smart contracts run deterministically, they need to be designed carefully to account for exceptions, uncertain conditions, and conflict resolution. Some frameworks add features like multisignature or layered governance to allow authorised parties to pause or modify a contract if something critical happens. Balancing automation with oversight is a big design challenge. Overall, the interaction of on-chain code, off-chain data, and possible centralisation reveals how flexible smart contracts can be for real-world applications.

3.2.1 Mechanisms and Design Principles

The success of smart contracts hinges on design principles that shape the code, architecture, and security. Principles like the following let contracts work as intended after deployment and keep them safe from attacks:

1. determinism means every node running the same code with the same inputs should get the same outputs, ensuring network-wide consistency;
2. minimal trust assumptions aim to ensure that contract execution doesn't depend on any single participant's judgment;
3. verifiability means the code and transaction history are open for auditing, which boosts accountability.
4. modularity matters, too, so that large tasks can be broken into smaller parts, each verified on its own.
5. formal verification, static analysis, and layered testing can help prevent major exploits.
6. upgradeability features, especially in permissioned blockchains, let to evolve contracts without ignoring the benefits of immutability.

All these design basics guide how developers create and maintain smart contracts, making sure they remain stable even as networks grow or change. If these principles aren't followed, it could lead to financial and reputation loss, or bigger system-wide problems.

3.2.2 Recent Developments in Smart Contract Technologies

Smart contracts have evolved quickly recently, thanks to protocol improvements and new languages or platforms emerging. Ethereum, the leading platform for programmable blockchains, is seeing big changes like the Prague-Electra (Pectra) update and EIP-4844 (Proto-Danksharding), which aim to boost scalability and lower fees for layer-2 solutions [16]. Ethereum's Shapella upgrade and its move to Proof of Stake also strengthen security and cut energy use [15, 14].

Meanwhile, new programming languages are popping up to make smart contracts safer and more expressive. Flow's Cadence [12] focuses on resource-oriented concepts to avoid standard flaws, and Move (used by Aptos and Sui) emphasises asset safety and formal checks. Solidity also keeps evolving, with updates and security reminders from the Ethereum Foundation [29].

Security and reliability are still top priorities. Advanced auditing, formal verification, and bug bounties are now typical for important contract deployments. And networks like Chainlink [22] help connect smart contracts with external data, though they also bring new potential risks. Taken together, these improvements tackle older challenges like scalability, interoperability, and security, making the technology more appealing to enterprises and institutions.

3.3 Traditional Contracts vs. Smart Contracts

Smart contracts are often compared to traditional legal contracts, raising questions about their legal weight, legitimacy, and how code interacts with law. On the surface, both types outline conditions and actions that govern relationships between parties, but traditional contracts rely on text and legal rulings, while smart contracts use code that runs on a blockchain. Moving from interpretation-based to algorithm-based enforcement affects how fast deals are closed, how much they cost, and how disputes are settled. In traditional cases, disputes often require lengthy court proceedings, whereas with smart contracts, the code enforces outcomes instantly once certain triggers happen. But if the code has a bug or the data is wrong, a contract could do something unintended, with little room for a court to step in. So while automation is appealing, the lack of a human interpreter can lead to surprising results.

Table 3.1: Traditional Legal Contracts vs. Smart Contracts

Aspect	**Traditional Contracts**	**Smart Contracts**
Enforcement	Judicial system	Deterministic code execution
Interpretability	Subject to legal interpretation	Executed exactly as written in code
Dispute Resolution	Courts and arbitration	Limited; pre-coded outcomes only
Modification	Amendable by agreement	Often immutable post-deployment
Error Handling	Judges can assess intent	Code runs regardless of context

Law experts note that traditional contracts rely on principles like good faith and fairness, and judges can interpret them in various contexts. In contrast, code-based contracts enforce exactly what's written, without that flexibility. This strictness is sometimes good when parties want predictable results and don't want to rely on personal judgment. But it can be difficult if things change after the contract is deployed or if something unexpected happens that the code doesn't address. Also, consumer protection is a concern because not all users can read or understand contract code. Some places have started writing rules for how code-based contracts fit into current laws, suggesting we may see a hybrid model. The topic still spurs lots of academic debate, and creating universal standards is tricky because laws differ so much from place to place.

To reconcile code and law, some have introduced Ricardian contracts, which pair human-readable terms with machine code for clarity in disputes. We also see multi-layer models that reference legal codes right inside the contract, blending computational enforcement with recognised legal standards. In heavily regulated fields like finance or healthcare, blockchain consortia work with legal experts to set best practices that balance efficiency with traditional protections. As they keep working together, how code-based certainty blends with legal interpretation will keep shifting, affecting new forms of contracts going forward. In short, while smart contracts offer big gains in efficiency, meshing them with the flexibility and safeguards of standard legal systems is still a work in progress. New case studies show how hybrid approaches are evolving through steady trial and error.

Comparing traditional and smart contracts shows a field in transition, where law, technology, and society all meet. For organisations ready to adopt them, these distinctions require careful reviews of operational risks, regulations, and how to handle disagreements. Supporters say even partial use can cut costs, reduce mistakes, and open up new ways of doing business. Critics point out that the technology is still immature and can lock in undesirable outcomes, especially if there's no easy way to override a bad contract. A middle ground suggests that smart and traditional contracts won't simply replace each other. Rather, they might grow alongside each other, influencing standards and protocols as they go. The next section shows how different industries are actually putting these ideas into practice, using smart contracts to speed up processes, boost trust, and test new automated approaches. In the end, uniting these models calls for a cross-disciplinary team, ongoing tech improvements, and constant dialogue among all parties.

3.4 Use Cases and Applications

The true benefits of smart contracts are clearest when we look at actual use cases across many industries. From speeding up financial settlements to boosting transparency in supply chains, these contracts can transform traditional processes. By automating tasks that rely on trust among different entities, they reduce manual errors and strengthen security. Smart contracts manage trust-based workflows in ways that cut down errors, resist fraud, and offer almost immediate verification. They're flexible enough to be adapted to many needs—like converting physical assets into tokens, settling insurance claims automatically, or handling complex negotiations among multiple parties. Beyond these everyday benefits, smart contracts open up new economic models, shaping how data ownership, risk, and governance might be handled. Still, widespread use faces challenges: standards are not always mature, legal guidelines remain hazy, and learning the technology can be tough. In the next sections, we focus on two major areas—asset management and

business process automation—to show how strong an impact smart contracts can have. Each comes with its own hurdles, reflecting the range of issues when using automated logic in real settings. But the progress so far points to a pivotal moment where theory meets practical solutions, proving the broader potential of the technology.

3.4.1 Focus on Asset Management

Asset management, both at the institutional and retail levels, represents a significant domain wherein smart contracts prove to be particularly efficacious.By embodying assets such as stocks, bonds, real estate, or collectibles in the form of digital tokens, market participants can benefit from accelerated transactions, fractional ownership, and simplified custody processes. Typically, asset transfers necessitate the involvement of numerous intermediaries such as brokers, clearinghouses, or depositories, which can result in settlement delays or errors. Through the utilisation of smart contracts, the majority of these processes can be automated, ensuring that once a transaction satisfies specific conditions (e.g., verified identity, sufficient collateral), ownership is transferred immediately and irreversibly. This configuration is particularly advantageous in rapidly evolving markets as it reduces liquidity risk and minimizes administrative complexities. Furthermore, compliance checks can be integrated directly into the contract, assisting in the prevention of unauthorised trading or illicit activities. Nevertheless, several questions remain unresolved: what is the legal validity of digital token-based ownership rights? Additionally, what occurs if an individual loses their private keys or if they are stolen, rendering assets inaccessible? Furthermore, extensive token markets may encounter abrupt disruptions that pose heightened risks. In regulated sectors, there is a necessity for regulations that are aligned with these novel methodologies.

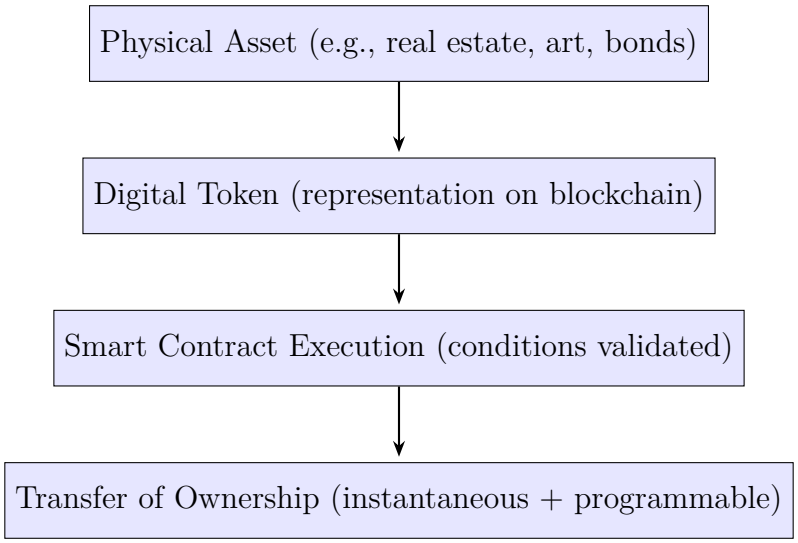


Figure 3.3: Tokenisation Process: From Physical Asset to Digital Transfer via Smart Contract

3.4.2 Business Process Automation

Apart from finance and asset management, many companies also use smart contracts to automate different operations. By programming rules and triggers in software, firms

can coordinate tasks across departments, external suppliers, and regulatory checkpoints. For instance, in supply chains, a smart contract might release payment only after goods are confirmed delivered, or reorder items automatically once stock dips below a set level. This removes the need for manual checks and speeds up transaction times. In healthcare, they can also regulate patient data exchanges between clinics, ensuring that only authorised staff see sensitive records while improving both privacy and efficiency. In insurance, claims handling can get faster by encoding specific triggers—like shipping delays or weather data—that automatically lead to payouts. This lowers friction and quickens settlement. However, linking smart contracts to existing systems (like ERP software) and older databases is key to frictionless operations. That often means reorganizing internal processes and training staff. Organisations must also invest in development and ongoing maintenance to reap these benefits. In the end, these changes can yield leaner services, stronger data management, and more flexible supply chains.

3.5 Benefits and Potential of Smart Contracts

3.5.1 Trust Minimisation and Operational Streamlining

Smart contracts bring many benefits, with one of the biggest being better trust between parties who don't have a long history of working together. Because these contracts put the deal's terms into open, self-executing code, trust doesn't hinge on third parties or external enforcers as much. This approach speeds up transactions and can enable new collaborations often stifled by fear of opportunistic behavior. Operationally, self-execution cuts down admin work, since repeated checks and paperwork are replaced by code triggers. As a result, organisations can shift resources away from routine tasks to high-level strategies, boosting competitiveness. Additionally, the permanence of blockchain records helps with auditing, which is especially useful in regulated sectors like finance or healthcare.

3.5.2 Transparency and Dynamic Value Exchange

Another key benefit is the extra transparency that a decentralised ledger can provide. When essential contract details are on-chain, it's easier to check that everyone follows the terms, making it especially helpful in partnerships across different organisations. In supply chains, for example, verifying a product's origin, travel path, and status on an unchangeable ledger can reduce scams. Regulators can also benefit from real-time monitoring, saving on audit costs and spotting suspicious activities sooner. Smart contracts additionally allow new ideas like flexible pricing, performance-driven service contracts, and automated risk sharing, transforming standard deals into dynamic, event-based value flows. By automating trust, participants face less uncertainty about each other.

3.5.3 Efficiency Gains and Scalable Automation

Another perk is cutting administrative tasks. Traditional contracts often involve repeated drafting, review, and manual updating, leaving room for human error and delays. In contrast, a smart contract automatically proceeds to the next stage once specific requirements are fulfilled, removing the need for constant management. This streamlined flow also speeds up transactions and lowers costs. Banks can use smart contracts to

automate loan payments, margin calls, or collateral management, reducing risk from human oversights. Startups, especially in developing areas, also gain from easier market entry because the tech can be launched quickly with fewer middlemen. Still, achieving these efficiencies calls for significant expertise, thorough testing, and collaboration across different fields, or else the tech might be adopted haphazardly.

3.5.4 Governance Innovation and Emerging Applications

Finally, smart contracts are not just about operational improvements. They can also usher in new governance and organisational models. For example, Decentralised Autonomous Organisations (DAOs) rely on smart contracts for voting, funding decisions, and policy enforcement, boosting participation but also posing risks if they're poorly designed. Similarly, microinsurance services in developing areas use smart contracts to pay out automatically if weather or location data meets certain conditions, extending coverage to underserved populations. These examples show how flexible such contracts are: once thoroughly written and checked, they can also handle corporate structures and even public-private partnerships. But strong oversight, backup planning, and keeping the code updated are vital to avoid becoming stuck with flawed governance or permanent coding errors. Together, these possibilities mark the leading edge of what's possible with smart contracts.

Table 3.2: Smart Contract Benefits and Concrete Applications

Benefit	**Real-World Example**
Trust Minimisation	Peer-to-peer lending platforms like Aave automate loan terms, ensuring fund release only after collateral is locked.
Administrative Efficiency	JPMorgan's Onyx blockchain reduces trade settlement times from days to minutes using smart contract automation.
Transparency	IBM Food Trust allows stakeholders to track food supply chains immutably, helping detect fraud or contamination sources.
Regulatory Oversight	Chainalysis tools assist regulators in real-time monitoring of blockchain-based financial activity for AML compliance.
Innovative Pricing Models	Ride-sharing DApps use demand-responsive smart contracts to dynamically adjust pricing based on usage or fuel costs.
Performance-Based Payments	Insurtech startups offer crop insurance where smart contracts auto-disburse payouts if satellite data shows drought.
New Governance Models	DAOs like MakerDAO enable token holders to vote on protocol changes without centralised leadership.

3.6 Challenges and Limitations

Even though they're on the rise, smart contracts still face challenges and limitations. Real-world use often shows complexities that theory alone doesn't capture, as integrating code with legal, social, and organisational aspects can be tough. Smart contracts follow the code exactly, giving little room for human judgment in unexpected situations: this can be a strength but also a weakness. Also, some contract relationships are too complex or interpretative to be fully coded. Traditional dispute resolution can be tricky to replicate in purely automatic systems. At the same time, laws around smart contracts vary by region, causing uncertainty for international projects or highly regulated sectors. Recognizing these constraints is key to responsible use, ensuring the technology fits real-world needs and meets legal or compliance rules.

For example, complicated agreements that cross borders can run into conflicting rules, making it hard to encode every legal detail into code. And depending on oracles means there might be new ways for hackers to undermine trust if these data feeds fail or get compromised. Because of all these issues, it's crucial to balance big aspirations with caution when launching smart contract projects.

3.6.1 Technical Challenges

Technical barriers are often the biggest roadblock for smart contracts. For example, blockchain networks have limited throughput and can get congested, slowing down contract operations. Additionally, coding flaws are a huge worry, since any bug can cause big losses or damage a project's reputation. Unlike typical software that you can update easily, permanent smart contracts usually need to be perfect from the start or have carefully planned fallback methods. Advanced strategies like layer-2 scaling or off-chain computations aim to ease stress on the main network, but that also adds system complexity. We have more security audits, analysis tools, and bounty programs than ever, but guaranteeing no bugs is still tough. Even small mistakes like integer overflows or incomplete fallback functions can lead to big trouble. Also, consensus methods can fail if there's a 51% attack or a network fork, which threatens the trust model behind the whole system.

Finally, plugging into existing IT setups is never straightforward, as data formats or older infrastructures may not match the demands of blockchain-based solutions.

3.6.2 Legal and Regulatory Uncertainty

Laws and regulations remain unclear for a lot of areas in smart contract use, especially across different countries. Some places have passed laws or given guidelines saying these code-based deals can be legally binding, but other places haven't. This uneven landscape complicates the pursuit of uniform global adoption. In addition, data rules like GDPR can conflict with blockchain immutability, raising doubts about how to handle a 'right to be forgotten' in a ledger that can't be changed. There are also ongoing discussions about protecting consumers, since they might sign off on terms they don't fully understand in code. Resolving conflicts is trickier if there's no built-in code for it or if a court doesn't easily intervene. So it's clear that we need continuous cooperation between lawmakers, industry, and developers to ensure that innovation doesn't outpace our ability to regulate it responsibly.

For example, certain places might not consider a contract binding if crucial parts can't be read in plain text. Such differences make it hard to have consistent rules worldwide.

3.6.3 Operational Challenges

Companies also face operational problems when trying to align old systems with newer blockchain setups. Shifting tasks or data onto a decentralised network is rarely simple. It can require rewriting processes, training employees, and rethinking internal governance rules. There's also the issue that different blockchain platforms might have different consensus protocols, transaction formats, or languages, making it hard to exchange data between them. Corporate leaders might not like losing direct control, especially in industries used to top-down oversight, so they worry about risking compliance if tasks shift to automated scripts. There's also a cultural hurdle: employees and executives alike must understand the technology's potential effects and what it means to rely on decentralised systems. Even if pilot projects show early promise, taking them to full scale requires careful audits, broad consensus among stakeholders, and often a big investment. In short, fitting smart contract protocols into existing workflows highlights that success depends as much on how flexible an organisation is as on how advanced the technology is.

These challenges grow stronger in global firms dealing with varied rules and older, inconsistent systems. This underscores the point that deploying smart contracts effectively calls for readiness across the board.

Table 3.3: Challenges and mitigation strategies for the adoption of smart contracts

Category	**Main Challenges**	**Mitigation Strategies**
Technical	Code vulnerabilities (e.g., reentrancy, overflow); network congestion and scalability issues; difficulties integrating with legacy systems	Formal verification, code audits, and bug bounty programs; scalability solutions (layer-2, off-chain); standardised middleware and APIs for integration
Legal and Regulatory	Jurisdictional uncertainty; conflicts between GDPR and immutability; difficulty understanding coded terms by users	Hybrid contracts (code + legal prose); blockchains with selective privacy; improved user interfaces with legal annotations
Operational	Integration with existing infrastructure; cultural resistance to decentralisation; low blockchain literacy	Change management and training programs; incremental pilot implementations; multi-stakeholder governance models

3.7 Related Work and Existing Solutions

Research on smart contracts is moving fast, driven by academic studies, open-source communities, and corporate pilots. Many frameworks, patterns, and recommended practices

have emerged, each tackling different problems like security gaps, scaling, or legal compliance. One notable trend is trying to combine formal verification, which mathematically confirms code correctness, with the iterative style of normal software development. Bugs like reentrancy or unchecked call stacks keep appearing, so there's constant research on new debugging tools or advanced static analyzers to catch these issues early. At the same time, the conversation isn't just about code. Legal experts, sociologists, and policy folks also weigh in on how code-based agreements work with existing laws. Standards groups and industry alliances produce guidelines that try to define requirements for interoperability and reduce the complexity of adoption. As a result, there's a wide range of evolving approaches, all aiming to improve how we build, test, and manage smart contracts. Academic publications often share case studies of new frameworks claiming key improvements in reliability or efficiency, but there's no single consensus yet, reflecting the ongoing growth in this field.

3.7.1 Blockchain-enabled Smart Contract Applications

Public Sector and Industry Pilots

Across the research scene, many blockchain-based projects show how smart contracts might change private and public sectors alike. In finance, decentralised exchanges (DEXs) run purely on contract code, matching buy-sell orders, while liquidity pools and automated market makers allow continuous trading. Such platforms suggest a future where everything from derivatives to insurance might be offered, traded, and settled on the blockchain. In healthcare, we've seen pilots testing how to verify prescriptions, trace drug origins, or manage patient consent forms with a single tamper-resistant ledger. Government services, known for heavy bureaucracy, may also benefit. For instance, property registries, licensing, or welfare payments can be simplified, reducing administrative costs and corruption risk. These varying trials show how flexible the contract logic can be, though most real implementations remain in testing phases. There are important questions about how to expand them beyond controlled pilots, since production-level systems demand more reliability than test setups. Also, balancing innovation with legal requirements is a key strategic hurdle.

Enterprise Adoption and Private Blockchain Frameworks

In businesses, more case studies are popping up on how consortiums run private or permissioned blockchains for data sharing, automated payments, and bridging separate IT systems. For instance, Hyperledger Fabric is popular among large companies seeking robust identity management and customizable consensus options. In this environment, chaincode offers a specialised path to smart contracts, focusing on privacy and detailed access settings. R3 Corda puts privacy first, exchanging data directly between parties, making it appealing for highly regulated financial markets. Though some academics critique permissioned networks for bringing back some aspects of centralisation, supporters argue these setups better fit real-world needs, where parties want to know each other and comply with regulations. Industry alliances are forming to set best practices for these solutions, possibly leading to standardised approaches. On top of that, cloud services now provide blockchain-as-a-service, making it easier for businesses to adopt, though it does raise questions about mixing decentralisation with central hosting. Within these different ecosystems, real-time interoperability is still a big issue: can blockchains talk to

each other or will they stay isolated? As a result, progress in cross-chain technologies, side-chains, and bridging solutions is closely watched by practitioners and researchers.

Further Reading and Open Frameworks

To grasp the variety of methods in the smart contract field, you can check out these open-source frameworks beyond Ethereum:

- **Hyperledger Fabric** – A modular permissioned framework for enterprise, emphasizing identity, privacy, and plug-in consensus.
- **R3 Corda** – A ledger built for regulated settings, supporting code-based contracts and direct data exchange with strong privacy.
- **Substrate (by Parity)** – A framework behind Polkadot, letting developers build customizable, interoperable chains with built-in contract capabilities via ink! or EVM.
- **TRON** – A fast, EVM-compatible chain focusing on content distribution and DeFi use cases.
- **Cardano (Plutus)** – Uses the Haskell-based Plutus language, grounded in formal methods, aiming at high-assurance apps.
- **Tezos** – A blockchain that can update itself, offering Michelson-based contracts (plus SmartPy or Ligo for easier syntax).
- **Algorand** – Has two contract layers: stateless for transactions, and stateful for complex logic, prioritizing quick finality and low fees.
- **NEO** – A chain supporting several contract languages, with an emphasis on digital identity, regulatory compliance, and scalable dApps.
- **Flow** – Made by Dapper Labs, Flow uses Cadence, a resource-focused language designed for ease and security.
- **Solana** – Famed for its speed and low latency, Solana supports Rust-based contracts and a parallel execution model.

Together, these platforms show how smart contracts continue to evolve, each with its own tradeoffs in performance, design, and developer tools.

3.8 Conclusion

3.8.1 Summary of Findings and Present Implications

In this chapter, we demonstrated that smart contracts are not merely advanced software tools, but integral components of broader socio-technical systems that hold significance for a vast array of stakeholders. By integrating cryptographic security with automation, they present a novel approach to fostering trust without the need for traditional intermediaries. The primary attributes of smart contracts were examined, including their reliance on deterministic on-chain logic and the use of oracles for acquiring external data, in conjunction

with a comparison to more conventional contractual methods. While applications such as asset tokenisation and automated supply chains reveal both the potential and challenges associated with their widespread implementation, they indeed offer transparency, reduced overheads, and a diminished dependence on trust. However, they are not without significant obstacles, including software vulnerabilities, ambiguous legal frameworks, and resistance from established organisations. Nonetheless, the extensive array of ongoing research and the development of contemporary tools indicate that these challenges are being approached from various perspectives. Anticipations for improvement are plausible as legislative frameworks become more defined, developer tools advance, and additional pilot projects are initiated. This progression signifies a substantive transformation in the management of contracts and automation.

3.8.2 Future Directions and Strategic Considerations

It's evident that the future of smart contracts depends on how blockchain scales, how privacy is addressed, and how the law evolves. Both public and private players will influence and be influenced by how we balance strict code rules against the need to adjust to real-world changes. To see real institutional adoption, we'll need strong governance approaches that allow upgrades, handle disputes, and link to off-chain systems. Effective standards and clearer global regulations can smooth out conflicts and make it easier to use these contracts across borders. As the tech progresses, future work should look at performance over time, any ethical concerns, and whether new organisational designs will arise from these code-based agreements. Bringing together technical experts, policymakers, and ethicists is key to making sure the benefits of automated, transparent enforcement come along with suitable safeguards. In essence, smart contracts lie at the intersection of creative solutions and practical demands, prompting us to rethink how we organize business and society.

Table 3.4: Smart Contracts at a Glance: Definition, Drivers, and Barriers

Aspect	**Summary**
What Smart Contracts Are	Self-executing agreements encoded on a blockchain, combining cryptographic assurance with automated execution to enforce predefined conditions without intermediaries.
Key Drivers	• Automation of business logic • Trust minimisation via deterministic code • Transparency and auditability • Reduced reliance on traditional intermediaries • Potential for cost and time savings in execution
Key Barriers	• Technical vulnerabilities (e.g., bugs, oracle attacks) • Legal and regulatory uncertainty • Organisational resistance and legacy constraints • Difficulty encoding complex or interpretive agreements • Challenges in scalability and interoperability

Chapter 4

Hyperledger Fabric

4.1 Introduction to Hyperledger Fabric

Hyperledger Fabric is considered one of the largest permissioned, enterprise-grade blockchain frameworks. It follows a modular design that meets real-world organisational demands for customizability, clarity, and security. Developed under the Linux Foundation's Hyperledger project, Fabric stands apart from public, permissionless blockchains like Bitcoin and Ethereum because it has a permissioned infrastructure, advanced identity management, and flexible consensus mechanisms. Its core principles (confidentiality, pluggable architectures, and integrated membership services) suit industries that require tight control over access to data, must observe strict regulatory standards, and need seamless integration with current IT systems.

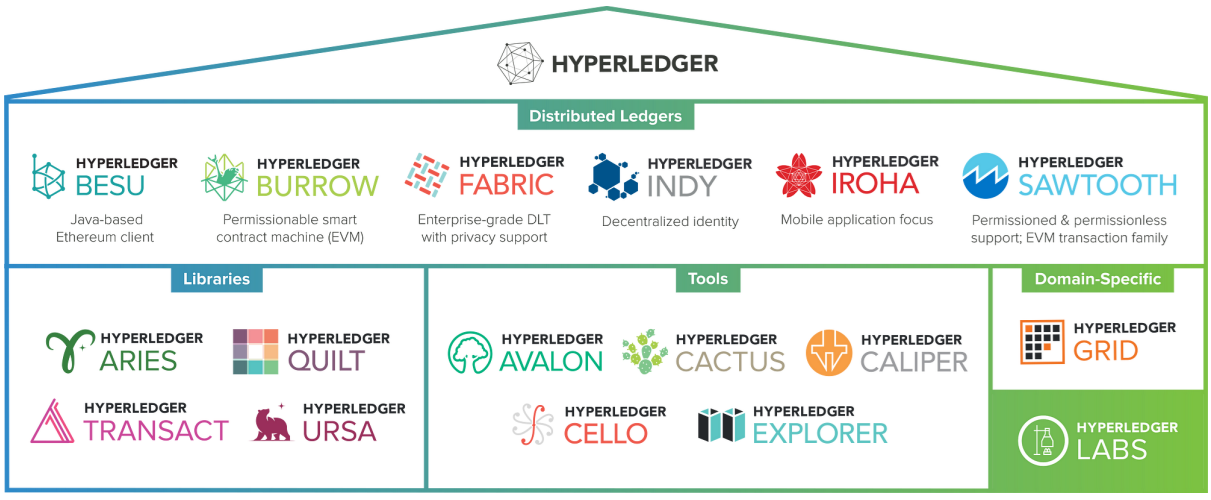


Figure 4.1: "Hyperledger Greenhouse" overview. Source: [5]

Many academic studies show that Fabric was partly designed to overcome issues found in earlier blockchain models. For instance, some permissionless blockchains rely on asynchronous consensus methods that can limit performance. By contrast, Fabric's "execute-order-validate" approach separates transaction execution from ordering, which improves scalability and deterministic behavior. Also, fields such as healthcare, finance, and supply chain management require that only authorised parties can endorse or view transaction data. Fabric addresses these needs by enforcing a membership system that gives each participant a cryptographic identity associated with a specific organisation. Researchers

describe Fabric as a flexible solution for multi-party contexts demanding reliable, shared ledgers based on carefully managed trust boundaries.

Though other permissioned platforms exist, Hyperledger Fabric is often highlighted in research for its consensus plugin framework, channel-based privacy features, and chaincode (smart contract) lifecycle controls that collectively support many usage scenarios. For example, machine-component manufacturers have tested Fabric-based ledgers to securely handle data from various suppliers. Similarly, the insurance industry has introduced pilot programs to automate claims and improve cross-company verification without disrupting internal systems. The goal is to leverage blockchain's immutability and trust features while still maintaining reliability, privacy, and regulatory compliance. Fabric's modular structure also ensures that organisations in a consortium can select only the components they need, including customised membership services and specialised chaincode environments.

This chapter explores Hyperledger Fabric in detail by examining its architecture, identity and membership system, governance model, ledger setup, chaincode lifecycle, ordering process, and the usual transaction flow through a security lens. Our analysis offers both theoretical and practical insights for researchers and professionals. Previous chapters introduced blockchain basics, cryptographic techniques, and various blockchain types. Building on that foundation, here we delve into the key ideas and real-world implementations that position Hyperledger Fabric as a leading enterprise platform. In the upcoming sections, we will look at each core component, from the membership service provider (MSP) that authenticates users to the details of how gossip data is shared, and show how these elements fit together in real deployments.

4.2 Fabric Architecture Overview

The Hyperledger Fabric architecture relies on a network of authorised nodes that operate under a shared setup. Its design focuses on several core ideas:

- A **modular approach** that allows core parts, such as the ordering service, membership system, and databases, to be swapped or updated.
- A **gossip-based communication protocol** that spreads newly created blocks among peers efficiently.
- A **layered transaction flow** that separates execution, ordering, and validation for performance and predictable results.
- A **channel concept** that divides the ledger into smaller subledgers for privacy and organisational needs.

It can help to imagine a layered model when thinking about the architecture:

- At the highest level is the *application layer*, where client apps use an SDK to communicate with the ledger.
- Below that is the *network layer*, which features membership service providers (MSPs) and the ordering service.
- The *ledger layer* is where the blockchain data and the world-state are stored, often backed by a local database.

- In parallel, the *chaincode layer* provides the environment to run smart contracts (chaincode) carrying business logic.

Researchers note that properly using these layers calls for an in-depth understanding of connections among identity management, policy rules, and secure consensus. In certain applications, such as big data analysis or intricate supply chain tasks, specialised chaincode runtimes or custom database designs may be needed.

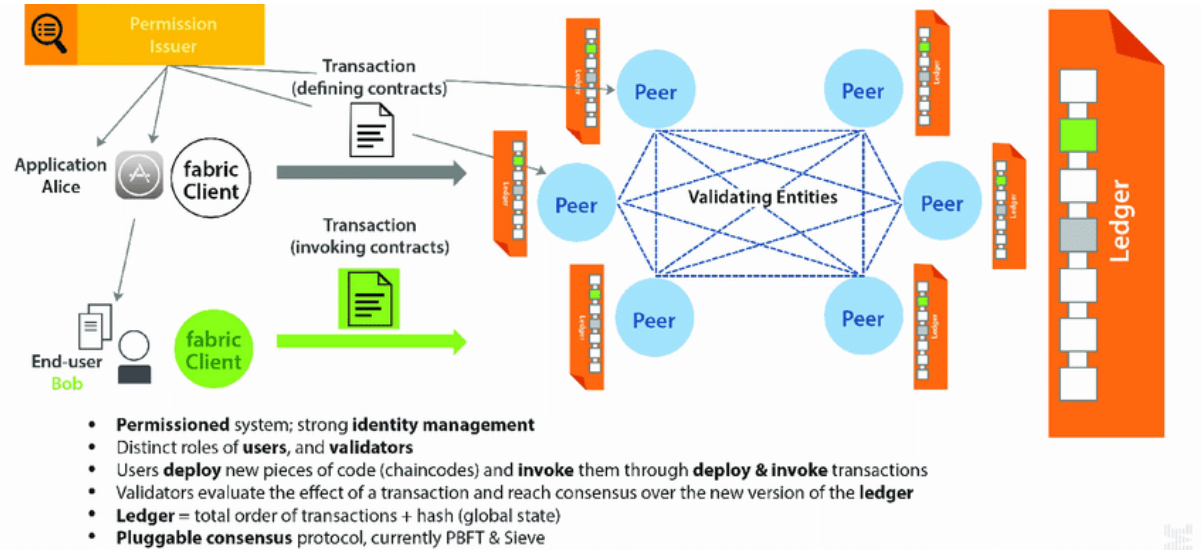


Figure 4.2: Hyperledger Fabric model. Source: [28].

4.2.1 Modular Design and Configurable Components

Fabric's modularity is often hailed as a key benefit because different organisations can adapt the framework to fit their own needs. In particular:

- **Consensus Plugin:** Early versions of Fabric included Apache Kafka for crash fault tolerance (CFT). The current default is Raft-based ordering. Yet Fabric can also integrate more advanced BFT algorithms if a consortium needs stronger protection against malicious nodes. Researchers are also looking at how plugin-based consensus can satisfy regional rules and use hardware security modules (HSMs) to speed up cryptography.
- **Membership Service Provider (MSP):** Fabric uses PKI certificates for identity management, but the MSP can be replaced. That means organisations can install a custom certificate authority or tie in an existing corporate identity system if they wish.
- **Database Backends:** Fabric supports `LevelDB` or `CouchDB` for the state database, each with its own pros and cons regarding performance and query powers. Some advanced labs even use SQL engines or specialised big-data layers for unique use cases.
- **Chaincode Runtimes:** The default container environment uses Docker, but in theory, other container platforms or even serverless frameworks could be used, so long as chaincode can be launched reliably in these new environments.

Besides these main parts, organisations can change block size, endorsement policies, and channel settings to make the ledger more direct, scalable, or open to change.

4.2.2 Recent Innovations in Hyperledger Fabric Architecture

Hyperledger Fabric versions 3.0 and 3.1 introduced major updates focused on scalability, security, and interoperability for enterprise users [17, 18]. A key addition is SmartBFT, a Byzantine Fault Tolerant consensus protocol that allows networks to handle malicious or failing nodes, offering greater reliability for critical apps. This new consensus option adds more deployment flexibility alongside the established Raft protocol [20].

Further changes include faster batch processing for chaincode, refined endorsement policy handling, and more advanced channel configuration functionality. These improvements boost transaction throughput and provide granular control of network rules. Tests show Fabric 3.x can reach higher throughput and lower latency than earlier versions, and in some use cases is on par with permissioned Ethereum networks [10].

IBM Blockchain Platform, built on Fabric, currently serves over 120,000 organisations worldwide with enterprise tools, managed services, and integration frameworks. An expanding ecosystem of open-source code samples and deployment scripts also supports rapid adoption [19].

All these enhancements now make Fabric a strong choice for scalable, secure, and regulation-friendly blockchain use in areas like finance, supply chain, and healthcare.

4.2.3 Gossip Data Dissemination

Hyperledger Fabric aims for reliability and strong performance by using a gossip protocol to distribute blocks among peers. Once the ordering nodes finalize a new block of transactions, they pass it to chosen peers, who then spread it using Fabric's membership-based gossip mechanism so that all legitimate peers in a channel get the block.

Key parts of the gossip process include:

- **Membership Tracking:** Only peers validated through the membership authority (i.e., those owning valid certificates) can join the gossip network. This stops unauthorised peers from listening or interfering.
- **Push / Pull Mechanisms:** Fabric gossip merges push and pull methods, which speeds up spreading new blocks while controlling bandwidth. A peer that receives a block pushes it to nearby peers, who can relay it further.
- **Dead Peer Detection and Fan-out Controls:** Gossip logic manages peers that go offline and prevents endless re-sending. It also ensures global coverage even when some peers have downtime.

Studies in distributed systems note that gossip-based replication can offer better fault tolerance, though it can add overhead in extremely large networks. Production networks using Fabric gossip sometimes adopt layered layouts, assigning specific peers as anchor peers or "config experts" to handle advanced tasks. Overall, gossip frees the ordering service from having to deliver blocks individually to all peers, improving both scalability and fault tolerance.

4.3 Identity and Membership Management

A major difference in Hyperledger Fabric is that it uses an identity framework. Each participant has a digital certificate that encodes organisational details, roles, and permissions. Unlike permissionless systems that allow anonymous or pseudonymous actors, Fabric aims to blend the benefits of distributed ledgers with the accountability expected in business settings. Many real-life consortia rely on formal agreements, so the ability to trace each node to a known organisation fosters trust.

4.3.1 Membership Service Provider (MSP)

The *Membership Service Provider* (MSP) is the foundation of Fabric's identity system. The MSP handles a few functions:

- **Identity Verification:** It confirms that an X.509 certificate belongs to an accepted entity. This can use the built-in fabric-ca or external CAs.
- **Role Assignments:** Certificates can carry attributes like "OU" (Organisational Unit) that declare roles such as peer, admin, or client. The MSP enforces access rules based on these roles.
- **Revocation and Renewal:** If an organisation revokes a certificate (for instance, if a key is compromised or an employee leaves), the MSP references the new CRL (Certificate Revocation List). That certificate can no longer endorse transactions.

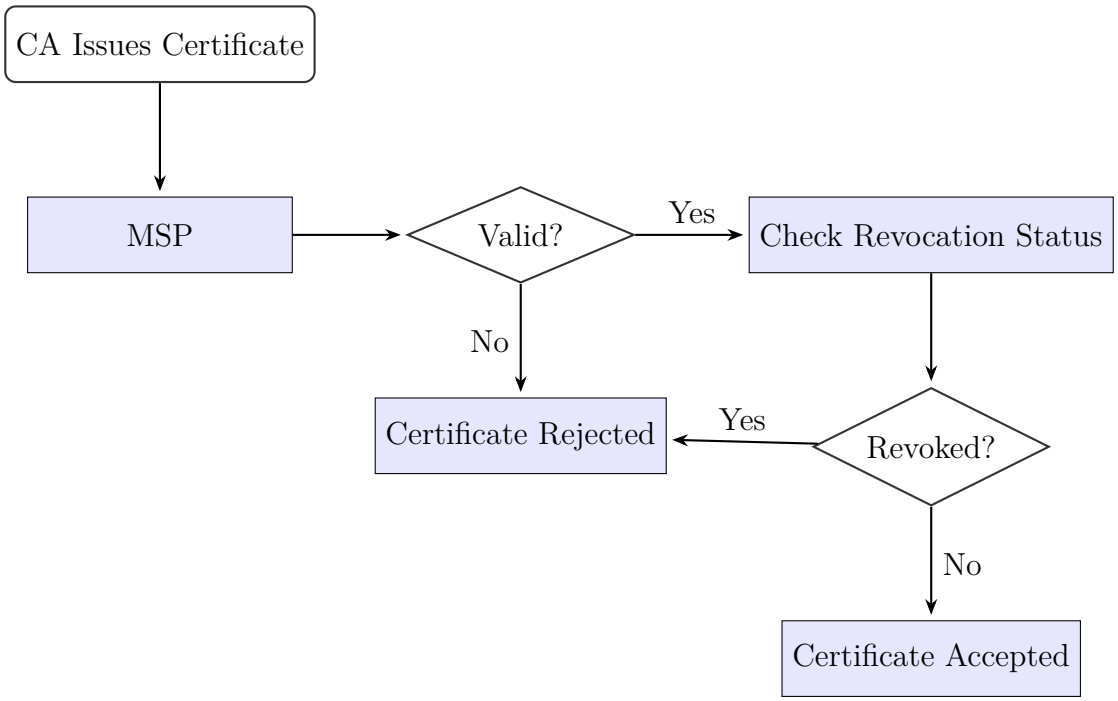


Figure 4.3: A step-by-step flowchart for how MSP validates certificates, from CA issuance to revocation.

Research indicates that MSP configurations are highly adaptable. For example, having multiple MSPs per channel allows separate trust domains or different cryptographic

policies. This flexibility ensures that cross-region networks can satisfy local legal requirements around identity and data.

4.3.2 On-channel vs. Off-channel Governance

Organisations often mix on-chain and off-chain governance practices:

- **On-channel Measures**: Changing a channel involves a "config transaction," which channel members must sign off on. The updated configuration block is then stored in the ledger so it remains transparent.
- **Off-channel Approaches**: Many consortia also use formal contracts or guidelines that define how they settle conflicts or manage compliance. While the ledger enforces technical policies, these external frameworks detail data liability and how audits are handled. Sometimes, references to off-chain documents are maintained on-chain for proof of governance actions.

By combining channel-level rules with off-chain agreements, Fabric networks can stay aligned with typical corporate processes and also meet real-world regulatory needs.

4.3.3 Revocation and Certificate Rotation

A key advantage of permissioned ledgers is the ability to revoke credentials to exclude compromised or unauthorised nodes:

- **Certificate Revocation Lists (CRLs)**: If a private key is leaked or an employee leaves, the CA updates its CRL. The MSP refers to that list, so any attempt to endorse a transaction with that certificate will fail.
- **Periodic Certificate Rotation**: Because cryptographic best practice calls for keys to be changed regularly, Fabric supports scheduled certificate updates. Each organisation can replace old certificates and re-register them. This reduces the chance of long-term key compromises.

Some setups also add chaincode-based features to record or notify network admins when certain endorsements fail repeatedly. This shows how identity controls can integrate with larger security processes.

4.4 Policies and Governance Model

Fabric employs a policy-based system that shapes how consortium members coordinate, define chaincode endorsement rules, and manage channel configurations without losing trust. Instead of fixed configuration files, Fabric puts these policies in channel data that is stored on the ledger. This means that governance decisions become part of the ledger and can be easily verified. The system also allows administrators to propose updates, gather signatures, and record these decisions, effectively baking trust into the platform.

- **Governance by Signature Policies**: Endorsement policies can say that transactions must be signed by a certain number of peers from each organisation. For more sensitive activity, like chaincode deployment, multiple organisation admins might need to sign.

- **Implicit Meta-Policies**: In some settings, simpler hierarchical rules exist, telling Fabric to "look up" signature policy details from parent channels or broader consortium guidelines. This avoids having many overlapping rules in large networks.

- **Dynamic Reconfiguration**: A unique Fabric feature is that policies can be updated during network operation. If a new organisation joins or endorsement rules change, the new policy is included in the ledger as an update transaction.

4.4.1 Policy Updates

Updating a policy in Fabric happens in stages:

1. **Proposal Creation**: Authorised users propose an updated policy block for the channel.

2. **Endorsement**: Relevant organisations sign the proposal according to the current policy.

3. **Ordering and Commit**: The ordering service adds the update to a block, which peers then validate and commit. The new policy applies after validation.

Because all steps are recorded on the ledger, no single party can secretly alter the policy. Researchers emphasize that such clear tracking of configuration changes benefits large consortia, which often evolve over time. Smooth governance transitions are critical for a blockchain network to stay viable under shifting membership or regulatory considerations.

Example: Minimal Policy Update in Fabric Configuration (YAML Snippet)

```
Application:
  Organisations:
    - &Org1
      Name: Org1MSP
      Policies:
        Admins:
          Type: Signature
          Rule: "OR('Org1MSP.admin')"

  Policies:
    Readers:
      Type: ImplicitMeta
      Rule: "ANY Readers"
    Writers:
      Type: ImplicitMeta
      Rule: "ANY Writers"
    Admins:
      Type: ImplicitMeta
      Rule: "MAJORITY Admins"
```

This snippet shows how a policy (for example, 'Admins') is structured and submitted as part of the channel configuration. It uses either 'Signature' or 'ImplicitMeta' logic. In practice, this YAML is bundled into a configuration transaction using tools like 'configtxlator' or the Fabric SDK.

4.5 Network Components

A Hyperledger Fabric network involves specialised nodes working together to ensure the ledger remains correct. In multi-organisation setups, each group typically runs at least one certificate authority, a few endorsing peers, and may also run ordering components. These parts need to integrate properly to keep operations consistent and fault-tolerant.

4.5.1 Peers, Orderers, and Channels

Peers are the core of Fabric:

- *Endorser peers* execute chaincode for transaction proposals, producing read/write sets and usually needing greater compute resources.
- *Committer peers* inspect final blocks and update the ledger. All peers can commit, but only endorsers run chaincode.
- *Anchor peers* help with peer discovery between different organisations. In large networks, they reduce overhead by maintaining partial membership details and letting other nodes find each other.

Ordering service nodes build transaction blocks. These nodes gather endorsed transactions from clients and put them in a deterministic order, then broadcast the blocks to every peer in the channel. Raft is the most common consensus algorithm in Fabric, tolerating node crashes. Experimental projects have explored Byzantine fault-tolerant ordering as well.

Channels divide the ledger so that different groups have separate histories. This allows privacy: A private channel might include only two organisations exchanging sensitive data, while a more open channel might involve all consortium members. Channels define which peers are included, how MSPs are configured, and what chaincodes are deployed.

4.6 Ledger and Data Model

Hyperledger Fabric's ledger records all valid transactions in a channel, including the history of blocks and the latest world-state. Unlike blockchains that re-run smart contracts for every query, Fabric stores a materialised *world state* that acts like a key-value store, speeding up reads.

4.6.1 Blockchain Storage

Within each channel, Fabric peers maintain an unbroken chain of blocks:

- Every block holds a collection of valid transactions, plus block metadata and a header linking to the previous block's hash.
- Once appended, a block cannot be changed. Any modification would conflict with the chain of hashes that follows.
- This ledger is effectively append-only, preserving the chronological order of transactions for audits and oversight.

In real-world use, organisations may store blocks on local drives, copy them to off-site archives, or even back them up in the cloud. Regardless, each peer can provide cryptographic proof that its ledger copy matches the canonical channel's chain.

4.6.2 World State with CouchDB

Along with the block log, Fabric creates a "world state" database by applying each valid transaction in sequence. The result is a snapshot of all key-value data at the latest block height. Administrators can pick `LevelDB` (a simpler key-value store) or `CouchDB` (a document store). CouchDB is popular in many enterprise contexts because:

- Chaincode storing data in JSON can be indexed for flexible queries, avoiding full ledger scans.
- Built-in secondary indexes and Fabric's range query features reduce the need for separate search services.
- Developers can implement more complex logic for queries and sorting using nested JSON structures.

However, if a chaincode generates large JSON documents often, performance may suffer. Best practices advise designing ledger schemas to balance how granular records are, which indexes to use, and how heavy the chaincode logic can be. Enterprises also weigh whether storing large files in the immutable ledger is worth the associated overhead.

4.6.3 Channels as Parallel Ledgers

An important innovation is the ability to maintain multiple channels at once.

- Each channel has its own blockchain and state database, entirely separate from other channels.
- Organisations can join certain channels and not others, which restricts who can see the data. This is useful for meeting regional data rules or handling certain agreements.
- Cross-channel transactions require special handling. Because there is no built-in atomic cross-channel logic, developers often rely on chaincode bridging or off-chain orchestrators to move assets between channels.

Thanks to these parallel ledgers, a single Fabric network can handle many scenarios. For example, a banking consortium could have one channel for inter-bank settlements, another for reporting to regulators, and another for handling derivatives trading, each with distinct policies and memberships.

4.7 Chaincode (Smart Contract) Lifecycle

Hyperledger Fabric refers to its smart contracts as *chaincode*. Chaincode defines how transactions update ledger states. Its lifecycle is designed for decentralised management of deployment and version control, preventing any single party from unilaterally updating code or undermining trust. This is especially vital for multi-organisation setups where no single entity should impose changes unilaterally.

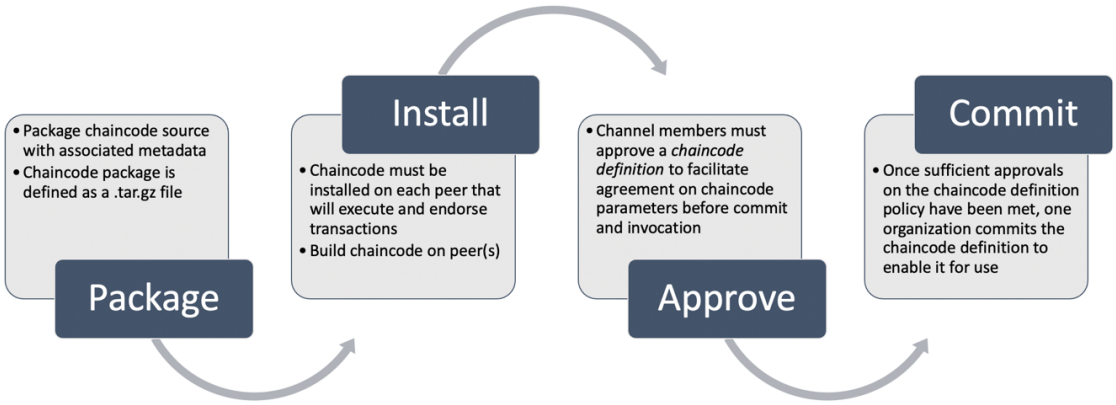


Figure 4.4: Managed Blockchain support for Hyperledger Fabric 2.2. Source: [2].

4.7.1 Development, Deployment, and Upgrade Processes

Developers typically write chaincode in languages like Go, Node.js, or Java. They define how the chaincode takes input parameters, reads or updates the ledger, and returns results:

- **Initialisation Functions**: Optional setup logic that runs when chaincode is first installed or instantiated.
- **Transaction Functions**: The main methods for business logic, covering actions like token transfers, record changes, or data checks.

After packaging the chaincode, each installing peer registers it locally, and the consortium collectively approves the chaincode definition. The final name, version, and endorsement rules must match what all parties expect. Once there are enough endorsements, the chaincode definition is committed to the channel. Any future upgrade follows a similar process, but with an updated version number or parameters. In regulated industries, these definitions and the accompanying signatures serve as clear evidence of who approved the chaincode at each step.

4.7.2 Containerised Execution Environment

By default, Fabric starts the chaincode in a Docker container, giving:

- Isolation from the core peer process, so if the chaincode crashes, it does not bring down the entire peer.
- Easier dependency management, as each chaincode instance can have the correct language runtime or libraries.
- Support for concurrent usage, letting the peer run multiple chaincode containers if needed to balance endorsement requests.

Some research explores running chaincode in Kubernetes pods or serverless deployments, which might lower costs for apps that only need chaincode services sporadically.

But container orchestration has to be managed so it does not disrupt response times or overload resources.

4.7.3 Endorsement Policies

Every chaincode must have an endorsement policy defining how many and which peers must endorse a transaction before it can be ordered. Some common examples:

- "One peer from both Org1 and Org2 must endorse."
- "Any two peers from a set of three organisations must endorse."
- "Chaincode can only be updated if admins from Org3 and Org4 both approve."

These policies ensure that the organisations involved must collectively validate any transaction changes, which prevents one compromised peer from falsifying data.

4.8 Transaction Flow in Hyperledger Fabric

A highlight of Hyperledger Fabric is its "execute-order-validate" transaction flow. Instead of having every node run each transaction in lockstep, Fabric uses three separate steps:

- **Execute**: The client sends a transaction proposal to the endorsing peers, which run the chaincode and generate a read set (keys and versions) plus a write set (proposed updates). If the code executes successfully, endorsers sign the proposal.
- **Order**: The client gathers these signed endorsements and sends them to the ordering service. The ordering layer combines transactions from multiple clients into a block and distributes it to peers on the channel.
- **Validate**: Each peer checks the block's endorsements against the chaincode policy and verifies the read set is still current (i.e., it has not changed since endorsement). Transactions are marked valid or invalid, and valid ones are used to update the local world state. The entire block, including invalid transactions, is appended to the blockchain.

By doing ordering after execution, Fabric lets endorsers run chaincode in a distributed or parallel manner without forcing every node to run every transaction. Experiments show that this structure significantly increases throughput, making Fabric well-suited for enterprise demands that might include hundreds or thousands of transactions in a short span.

4.9 Security Considerations and Potential Risks

Even though Hyperledger Fabric's permissioned approach and emphasis on identity management provide a strong security foundation, there are still issues that need attention. Network operators, system admins, and chaincode authors should recognize possible weaknesses and use best practices to control them. Some security reviews suggest that minor mistakes in policy settings or chaincode logic could result in data tampering or unauthorised access.

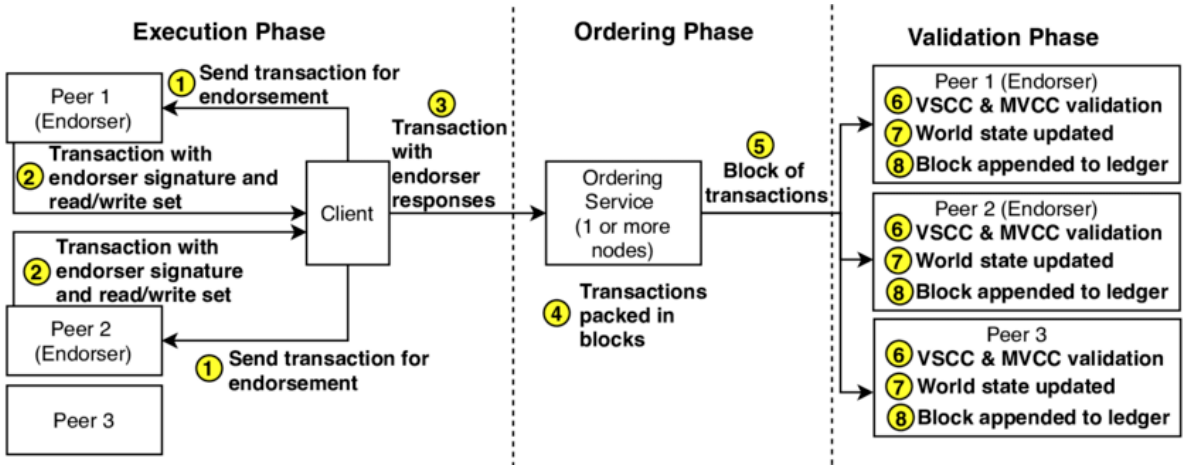


Figure 4.5: Transaction flow in Hyperledger Fabric. Source: [26].

4.9.1 Non-deterministic Risks

Fabric endorsers must all arrive at the same read/write sets, which requires chaincode to be deterministic. Any use of random numbers, timestamps, or external data sources can cause differences in results and lead to invalid transactions. Recommended steps include:

- **Deterministic Code**: Make sure the chaincode always produces the same result for the same input and ledger state. If randomness is necessary, use off-chain oracles or commit to a fixed value that all peers see.
- **External Data Management**: Chaincode should not call external APIs directly. Instead, let a client or an oracle service gather the data and pass it to chaincode so that every endorser sees the same input.
- **Testing and Simulation**: Thoroughly test chaincode to make sure any sources of nondeterminism are removed before going live.

Researchers also caution that concurrency or parallel runs can cause nondeterminism if the chaincode depends on states that are changing. To avoid this, chaincode developers must carefully structure data access and updates.

4.9.2 Privacy and Data Security Risks

While Fabric supports channels and private data collections to keep transaction data confidential, there are still safety concerns:

- **Channel Misconfigurations**: If a channel is set up incorrectly—either accidentally or on purpose—unauthorised peers may gain access. Regular MSP checks and multi-party governance help avoid this.
- **Transaction Visibility**: Even when a channel is private, some metadata might still be visible in block headers or transaction details. To protect privacy further, zero-knowledge proof techniques or off-chain encryption can be used to store only references or hashes on-chain .

- **Private Data Collections**: Only peers in that collection can read the data, but if the collection is large or a member is compromised, it risks exposure. Some consortia use data retention or ephemeral storage so information is only accessible briefly.
- **Cryptographic Key Management**: As with all cryptosystems, if private keys are lost or stolen, attackers can sign illegal transactions. Hardware security modules (HSMs) or strong key vaults are recommended to keep keys safe.

On top of that, certain regulations require removing personal data on request. Because blockchains are append-only, meeting "right to be forgotten" rules can be challenging. Fabric's privacy methods can store just a hash or part of the data on-chain. In some cases, encrypting the data and discarding the key later is the only way to effectively hide it. However, balancing immutability against privacy rules remains a key subject in blockchain research.

Security Risk Checklist and Mitigation Strategies

Risk Area	Recommended Mitigation Strategy
Non-deterministic Chaincode	Ensure logic is fully deterministic. Avoid time-based functions, randomness, or external API calls. Use oracles or commitment schemes when external input is required.
External Data Dependency	Feed external data through clients or oracles instead of fetching inside chaincode. All endorsers must receive the same data input.
Chaincode Deployment Errors	Use automated testing and simulations to detect non-deterministic behaviour or edge cases before deployment.
Channel Misconfiguration	Perform multi-party configuration reviews. Validate MSP definitions and apply distributed approval policies.
Metadata Leakage	Minimise sensitive data in transaction headers. Use ZKPs or encrypt off-chain payloads, storing only hashes on-chain.
Private Data Exposure	Apply least-privilege principles to private data collections. Implement ephemeral storage or expiration policies for sensitive info.
Key Compromise	Protect signing keys using HSMs or secure vaults. Reset credentials regularly and audit usage logs.
Data Erasure Compliance	Encrypt personal data off-chain; destroy encryption keys to simulate erasure. Avoid storing complete personal records on the chain.

Table 4.1: Key security risks in Fabric

4.9.3 Industry Adoption and Real-World Deployments

Multiple large-scale commercial blockchain projects have successfully deployed Hyperledger Fabric or similar frameworks. For instance, IBM Food Trust, using Fabric, offers farm-to-table product tracking, with retailers like Walmart cutting traceability times

from days to mere seconds [11, 8]. In logistics, the TradeLens platform (by IBM and Maersk) uses blockchain to digitize shipping files and track containers, leading to better collaboration and transparency for global shipping partners [9].

The healthcare sector has tested permissioned blockchains for secure, auditable, and privacy-oriented data exchange, as described in a MIT Media Lab whitepaper [4]. These efforts confirm the feasibility and value of Hyperledger Fabric, encouraging further designs and use cases such as those detailed in the next chapter.

Chapter 5

Case Study

5.1 Railcar Sharing

This section describes a blockchain-based railcar sharing system that reduces inefficiencies in rail transportation for automotive logistics. By looking at the complexities of 'as-is' processes and demonstrating a new 'to-be' approach grounded in distributed ledger technology, the following sections show how multiple stakeholders in the railcar ecosystem, from car manufacturers to carriers, can achieve cost savings, operational transparency, and stronger accountability. The solution uses a permissioned blockchain architecture that offers robust identity management and scalable consensus mechanisms, effectively solving typical supply chain challenges like partially used wagons, asynchronous scheduling, and clashing incentives. The explanations, architectural design, and pilot implementation details presented here combine technical rigor with real-world expertise to create a practical system. A further review of deployment overhead, performance, security, and regulatory factors highlights the wide range of elements needed for a real-world blockchain rollout.

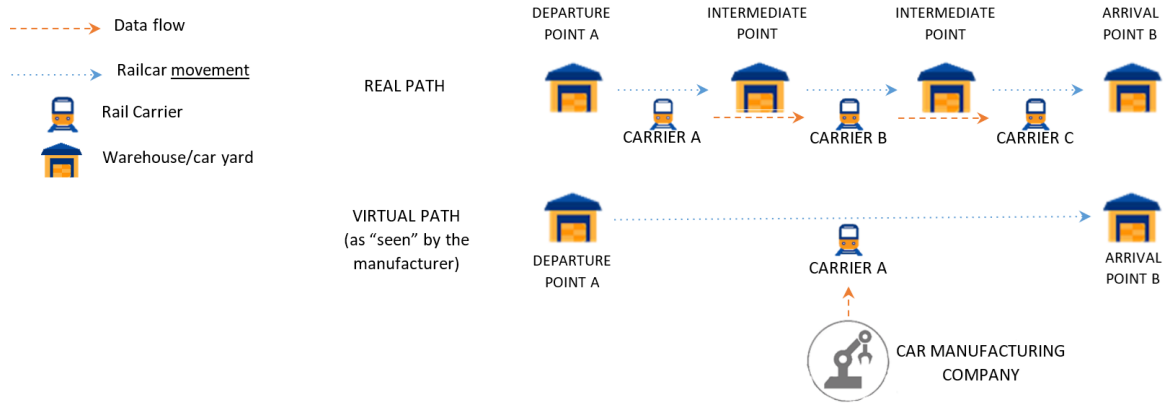


Figure 5.1: Railcar sharing: general overview.

5.1.1 Current State ('as-is' Process) and Business Case

The current 'as-is' model relies on one carrier to manage vehicle transport over potentially multiple rail segments. The cost is generally calculated based on the number of freight

wagons (often called railcars) rather than charging for each vehicle. As a result, this structure causes inefficiencies and suboptimal usage, weakening the supply chain. The following subsections outline the main participants and assets, along with their roles in the transaction workflow, culminating in how partial railcar usage and scheduling delays drive up operating costs.

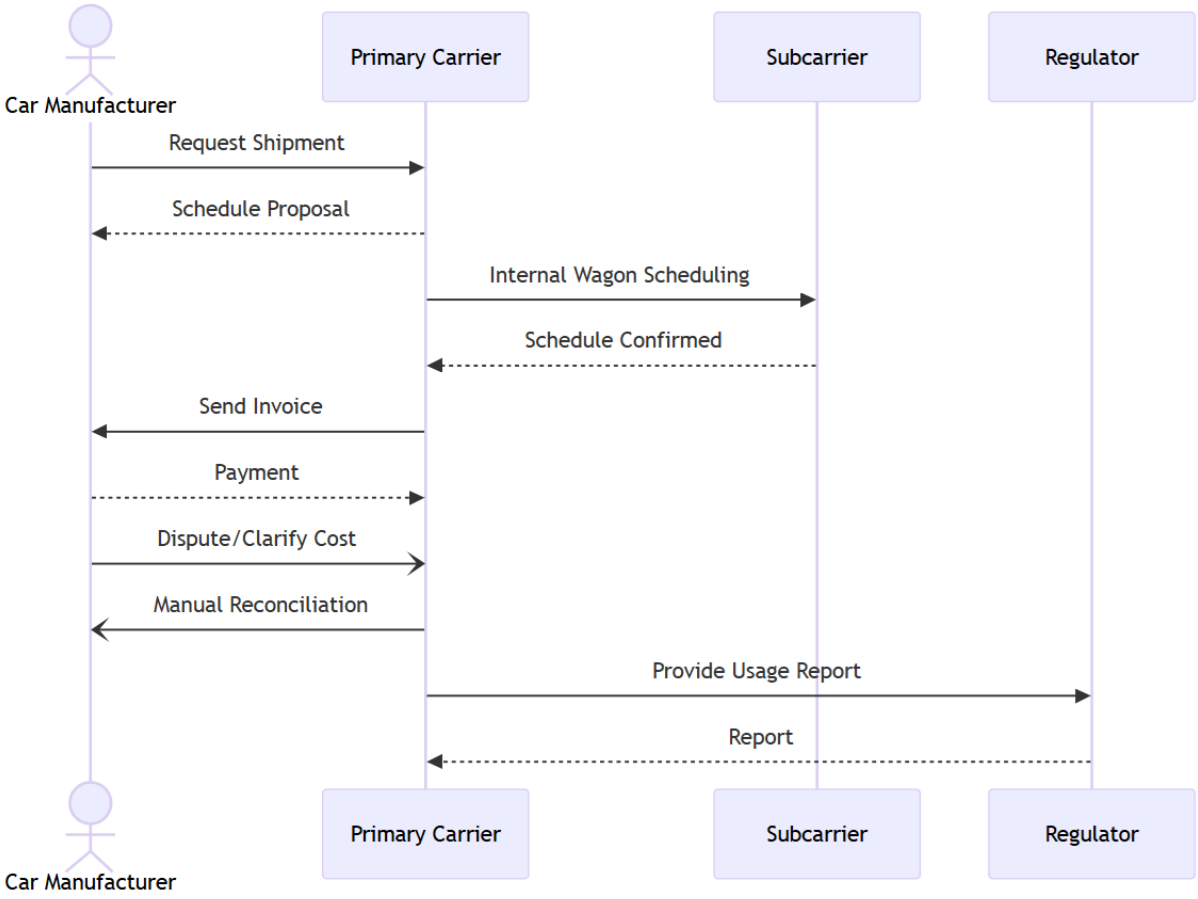


Figure 5.2: Railcar sharing: as-is process sequence diagram.

Typical friction points in the current process:

- **Data siloing** across OEMs, logistics providers, and rail operators slows planning and blocks transparency.
- **Cost-sharing disputes** occur when multiple clients split a partly filled wagon.
- **Lack of real-time tracking** causes scheduling mismatches and idle capacity.
- **Manual reconciliation** of transport records delays invoicing and dispute resolution.

Stakeholders and Participants

In traditional rail logistics for automotive shipments, there are usually two main stakeholder groups. The first are car manufacturers (one or more) that produce vehicles to be

shipped from point A to point B. The second are carriers, who organize the rail transport. However, in practice, things are often more complex:

- **Car manufacturer(s):** They usually operate large storage yards for new vehicles and set schedules to move inventories across regions or to dealerships. While they may negotiate with one carrier, several carriers might actually manage different parts of the overall route.
- **Primary Carrier Entity:** This group manages direct cost negotiations and billing with the manufacturer. They often subcontract other rail operators without involving the manufacturer.
- **Subcontracted Carriers (Optional):** In multi-leg shipments, these providers run parts of the transport chain but stay financially hidden from the manufacturer.
- **Regulatory Bodies (Potential Stakeholders):** They enforce safety rules and sometimes demand wagon usage logs for accident investigations or environmental checks.

In many industrial settings, having only one party handle negotiations leads to limited competition and low cost visibility, which often disadvantages the manufacturer. Over time, partial loads, empty railcar space, and waiting days for wagon consolidation drive up costs across the network. These factors cause unpredictable shipment delays, extra operational overhead, and tension between the main manufacturer and the carriers. Studies show that sharing resources among multiple manufacturers can reduce these problems if managed correctly.

Table 5.1: Railcar Sharing: Stakeholder Roles and Interests Matrix

Stakeholder	**Interest / Influence**
Car Manufacturer(s)	They want to lower transport costs and ensure deliveries are on schedule. Limited insight into subcontracted routes can raise total costs and slow reactions to problems. Working together with others can reduce idle wagon time and improve route efficiency.
Primary Carrier Entity	Aims to maximize profit by consolidating loads and subcontracting. They control transport operations and how costs are set, and might oppose transparency that restricts their pricing freedom.
Subcontracted Carriers	They want stable shipment volumes and timely operations. They usually have weaker bargaining power and limited information about high-level deals. Standardised records or digital trust systems can help them.
Regulatory Bodies	They oversee safety rules, emissions tracking, and fair competition. They need reliable, unchangeable records of wagon usage and multi-party deals for compliance checks or incident investigations.

Asset and Transaction Analysis

In rail freight, assets mainly fall into two categories. One is the *vehicles* or cargo to be moved, each identified by a unique ID or VIN if we're talking about cars. The other is the *railcars*, which are the wagons that carry the vehicles. In the 'as-is' system, older databases store several fields, such as:

- Vehicle details (VIN, model, storage location, shipping timeline).
- Railcar capacity, type, availability, and occupant status.
- Contract terms explaining how costs are calculated (daily wagon rent, route-based fees, or a combination).

Right now, transactions usually happen in central systems. When the manufacturer asks for a shipment, the carrier schedules wagons internally, tries to fill them efficiently, and later sends an invoice based on how many wagons were used. If a wagon isn't completely full, the manufacturer still pays for the entire thing. This arrangement doesn't encourage the carrier to quickly move a partially filled wagon, which leads to:

- **Stale Data Risks:** The manufacturer's system might list a shipping date that doesn't match reality if the wagon remains partly unfilled.
- **Transaction Delays:** Since key confirmations and cost updates stay on the carrier's side, updates to the manufacturer can be slow or missing.
- **Double-Counting or Overbooking Potential:** Without a shared ledger, the carrier might accidentally assign the same railcars to several shipments, especially if multiple manufacturers are using them.

Furthermore, data discrepancies can lead to disputes over final billing amounts, as each side might interpret wagon usage differently. Intangible friction from these repetitive negotiations and validations can hamper trust, requiring additional overhead in reconciling records .

Business Impact

Basing costs on whole railcars instead of actual vehicle counts often increases what the manufacturer pays. A half-full wagon raises the per-vehicle cost. Beyond the direct expense, late shipments bring hidden costs. Car makers usually plan their next logistics steps around expected arrival times. Delays throw off marketing, dealership schedules, and delivery to end customers. Meanwhile, carriers must keep half-filled wagons in storage longer, creating more planning overhead.

From a business standpoint, these issues appear in the following ways:

- **Excessive Freight Costs:** Unused capacity pushes up per-vehicle costs.
- **Lost Time Opportunity:** Waiting to fill railcars can lead to long delays, cutting into profits for both the manufacturer and the carrier, which is paid later too.
- **Constrained Scalability:** Relying on open railcar space and a single negotiation path limits agility and hinders just-in-time or lean approaches in today's manufacturing supply chains.

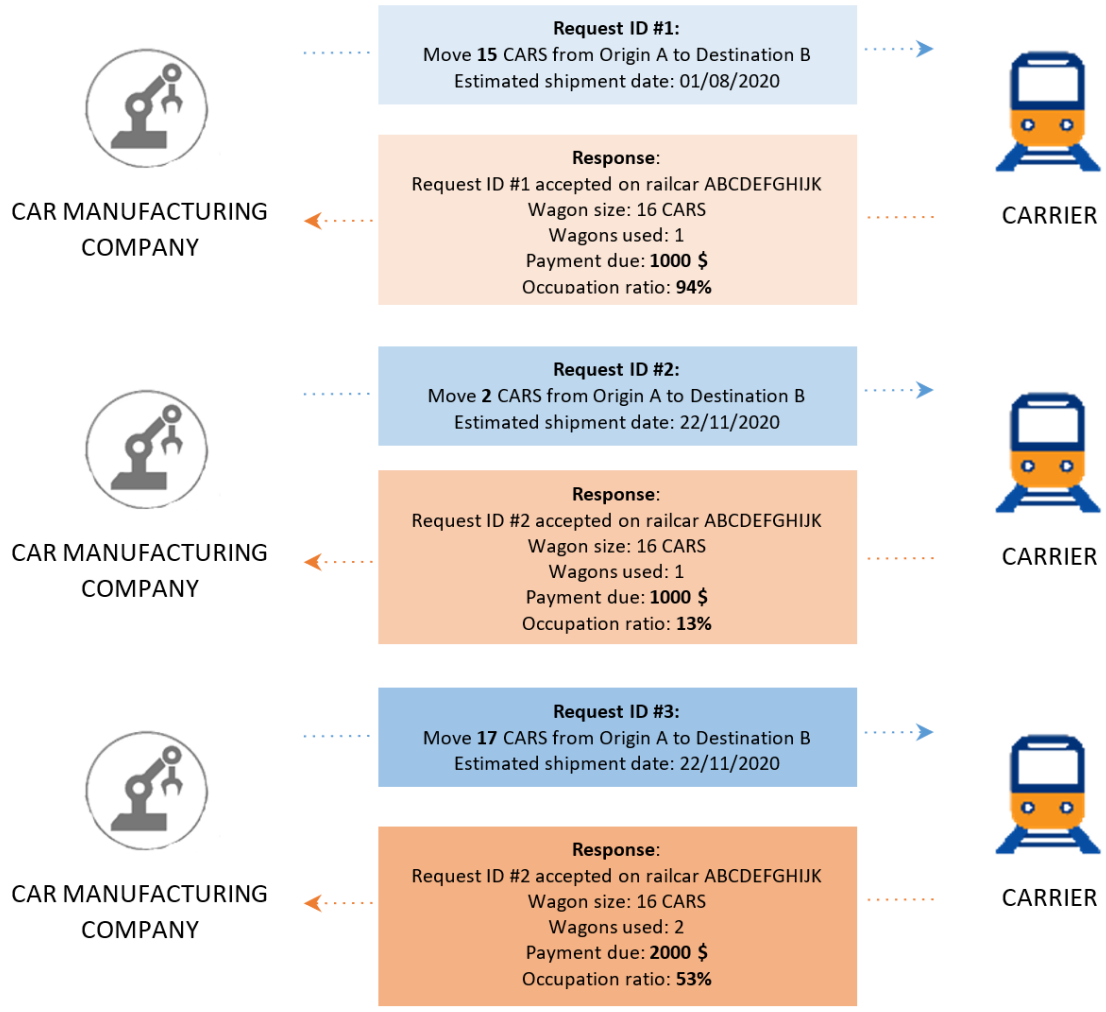


Figure 5.3: As-is process: typical transaction example.

Overall, the "As-is" setup makes a strong case for exploring systems that allow shared railcar usage, cost splitting, and near-real-time data sharing. Blockchain suits these goals well, given its tamper-proof ledger, role-based security, and ability to automate processes with smart contracts.

5.1.2 Proposed Blockchain-Based Solution ('to-be' Process)

Moving from the 'as-is' model to a more collaborative, open, and efficient ecosystem means rethinking how railcar space is allocated and how data are shared among stakeholders who may compete or collaborate. A permissioned blockchain provides the trust foundation and multi-party data updates needed for these changes.
By building a consortium of car manufacturers and carriers, the solution unifies railcar booking, tracking, and settlement on a single verifiable platform. This approach simplifies logistic processes and improves cargo usage, cost sharing, and error reduction.

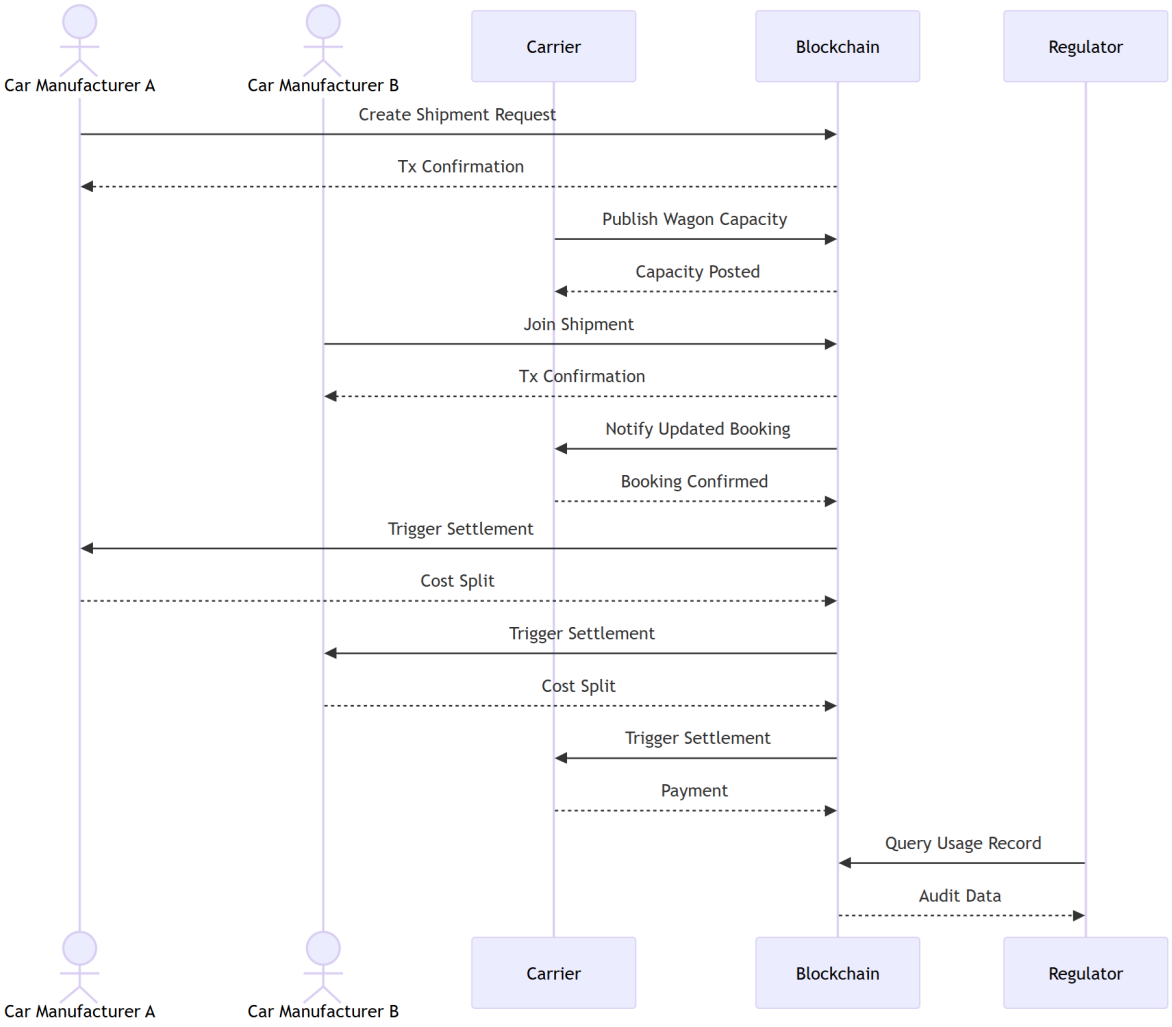


Figure 5.4: Railcar sharing: as-is process sequence diagram.

Solution Canvas

<table>
<tr>
<td rowspan="2">Constraints
• Car Manufacturing companies and Carriers MUST adopt the blockchain
• Permissioned Digital Ledger
• Scalable architecture
• Interoperability with existing external Information Systems</td>
<td>Decisions
• Sharing shipments
• Avoid delays and inaccurate information
• Visibility on pre-shipment process
• Outbound processes improvement
• Cost and time for delivery reduction (included disputes)</td>
<td rowspan="2">Decision maker
Car Manufacturers</td>
<td>Users/DMs relationship
• Car manufacturer → Car manufacturer
• Car manufacturer → Carrier</td>
<td rowspan="2">Users
• Car Manufacturer
• Carrier
• External Audit systems</td>
</tr>
<tr>
<td>Information/Resources
• Hyperledger Fabric (private network)
• REST server
• ANGULAR client</td>
<td>Solution channels
Proof of Concept/Pilot project</td>
</tr>
<tr>
<td colspan="2">Costs
• Development costs:
○ Implementation of BC (and eventual integration with current ISs)
○ Dedicated IT department (or external IT experts)
○ HW and SW (Blockchain Platform and server)
• Cost for the introduction of the solution
○ Development costs
• Maintenance</td>
<td colspan="3">Objectives
• Enable share of railcar
• Outbound processes improvement (costs and time reduction)
• Create trusted network of Car manufacturer companies and Carriers
• Reduction of disputes</td>
</tr>
</table>

Network Conceptual Design

In the conceptual blueprint, a blockchain network is the backbone of a shared railcar scheduling and settlement system. Key design components include

- **Multiple Car Manufacturers:** Each has a node or membership identity so they can submit shipping requests, accept partial capacity offers, and follow real-time railcar updates on routes.
- **Carrier(s):** They publish train routes, wagon capacities, partial fill levels, and scheduling constraints. They also get compensated according to usage or distance instead of a single wagon fee.
- **Smart Contracts (Chaincode):** These automate cost-sharing among multiple manufacturers in one railcar, block overbooking, and keep final settlement data.
- **Policy-based Access Control**: Endorsement rules say who must approve adding or updating capacity data or finalizing joint trips, ensuring key stakeholders sign off and trust each other.

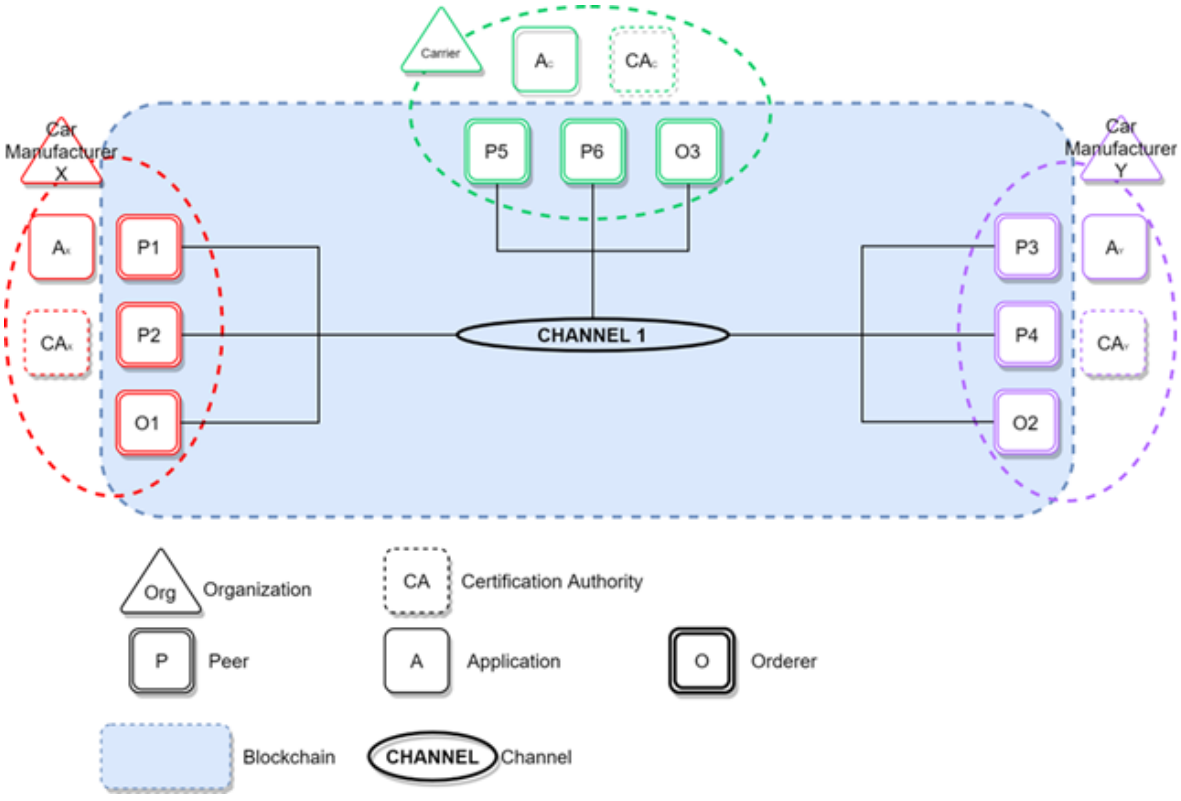


Figure 5.6: Railcar sharing: conceptual blockchain design.

In this model, a manufacturer who finds a partly empty wagon requests adding more vehicles at a proportional cost, subject to the carrier's approval. If one manufacturer alone cannot fill the wagon, the system offers the remaining room to others. Over time, this raises occupancy and lowers cost per vehicle. For the carrier, a shared ledger shows in real time how many vehicles from different manufacturers are on a wagon, which prevents double-booking. In addition, any contract logic or pricing function is visible to relevant parties, supporting faster dispute resolution. Cryptographically verified transactions discourage tampering and build trust throughout the network.

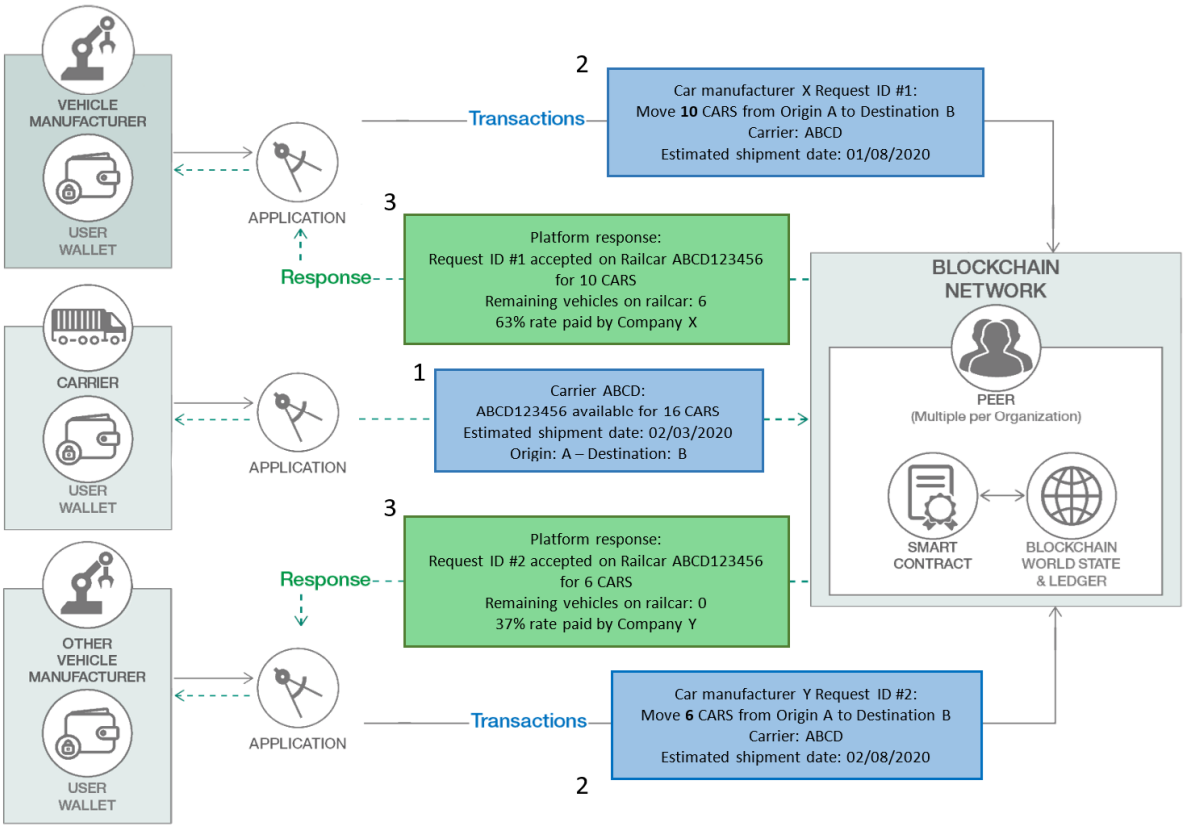


Figure 5.7: Railcar sharing: conceptual blockchain transaction.

Integration with Legacy Systems

Putting a blockchain solution into practice means connecting it with existing systems. In railcar sharing, these older systems manage:

- **Vehicle Manifests:** Often stored in relational databases or specialised ERP tools, these list each vehicle's ID, owner, and shipping timetable.
- **Scheduling Tools**: Carriers rely on internal route planners for feasible rail paths, wagon assignments, maintenance, and staff.
- **Financial Systems**: Invoicing or accounting modules control cost allocation and produce final bills for the manufacturer.

Since it's unlikely these systems will be replaced, the blockchain was connected through API layers:

1. **Data Synchronisation Layer**: Any new shipment or partial wagon booking on the chain updates the carrier's scheduling tool. Updates from those planning tools also post back to the chain so manufacturers have immediate insights.
2. **Off-Chain Storage Links**: For large or private data, references or hashes are kept on the ledger, letting participants confirm data integrity without storing huge volumes on-chain.

3. **Event Triggers and Notification Services**: The blockchain sends events to legacy systems, prompting invoice generation or vehicle tracking changes.

In the pilot, the new system ran in parallel with the existing "As-is" approach. This let participants compare real outcomes, find issues, and plan a gradual shift to the "To-be" method.

5.1.3 System Design and Implementation

After exploring the reasons behind using blockchain in rail logistics, we shifted to the technical design and tested a pilot version. This section covers our reference architecture, showing how Hyperledger Fabric manages user identities, how chaincode encodes cost-sharing logic, and how external frameworks (a Java-based middleware server and an Angular front end) provide a consistent user experience linked to the ledger data. We highlight the synergy among these layers to show how an abstract concept can be turned into an operational system.

Overall System Architecture

The railcar sharing solution uses a layered design:

- **Hyperledger Fabric Network**: A group of peers spread over several organisations (each manufacturer is one, plus at least one for the carriers), with a shared ordering service for transaction sequencing. Channels define which data each group sees, and more private channels can protect sensitive data like certain routes or prices.
- **Java-based Middleware (REST Layer)**: This server works with the Fabric SDK to issue or endorse transactions that modify data through chaincode. It also secures operations (token-based authentication, TLS) and translates high-level commands (e.g., "reserve five wagon slots") into detailed Fabric requests.
- **Angular Front-end Client**: This interface lets users browse upcoming shipments, partial wagon capacity, cost splits, and finalize reservations. Carriers can publish route schedules and wagon availability, or accept multi-manufacturer bookings.

In our basic demo setup, the network had two car manufacturers, one carrier, and a single organisation managing the ordering service. For real production, we added more ordering nodes (like a Raft cluster) and improved membership services for certificate updates at scale. Diagrammatically, the architecture places a Raft ordering cluster at the center, with each organisation's peers connected as spokes, and the Java REST service managing user interactions.

IBM Blockchain Platform for PoC

We used Hyperledger Fabric to build the blockchain network and deployed it on the IBM Blockchain Platform (a BaaS model). This approach simplified infrastructure tasks and provided an enterprise-style scale.

The network we set up included three main organisations:

- `org1.example.com` and `org2.example.com`, representing two car manufacturers;
- `carrier.example.com`, a shared entity representing logistics carriers.

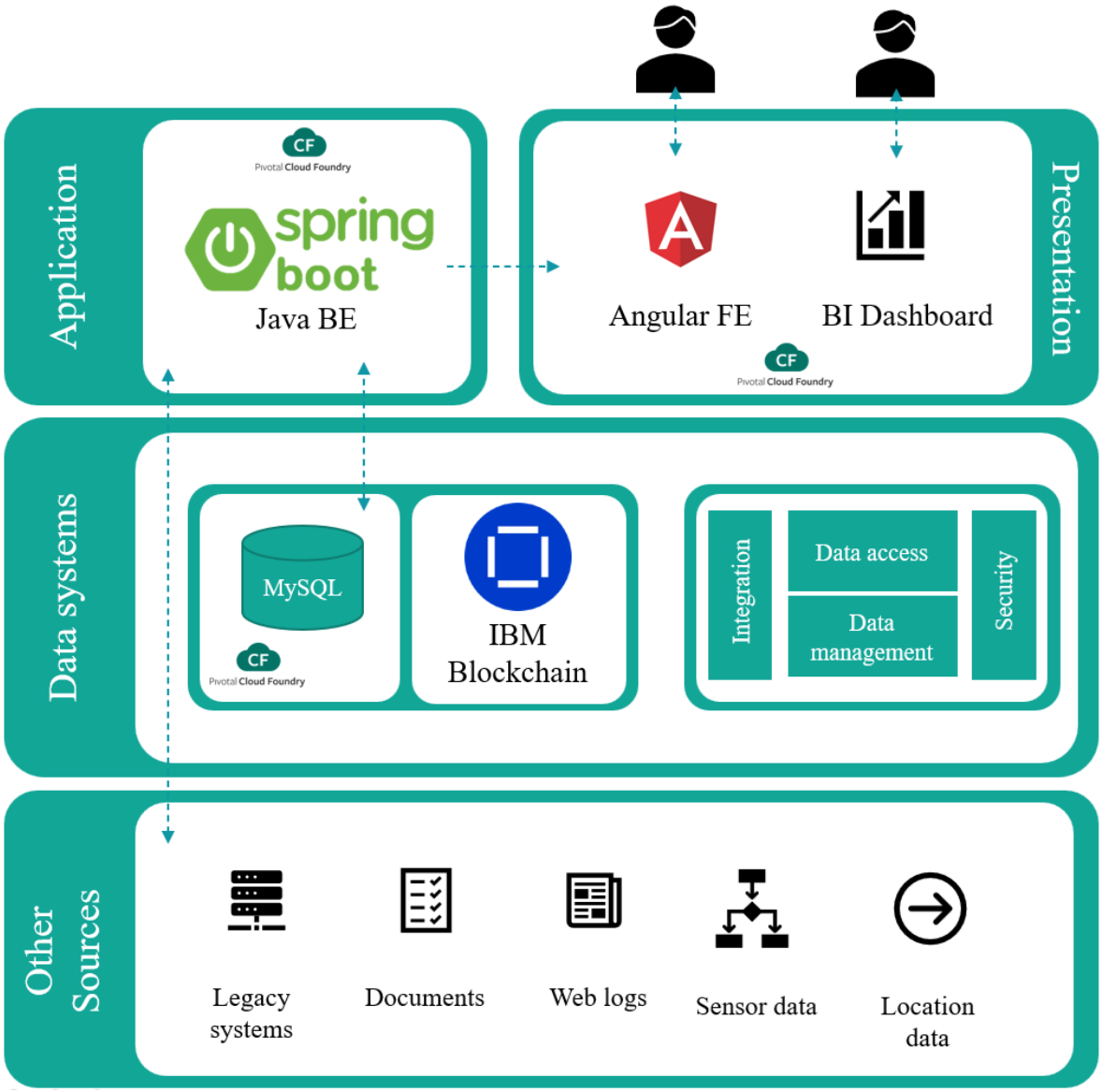


Figure 5.8: Railcar sharing: high level software architecture.

In each organisation, we set up the needed Fabric components:

- Two peer nodes (`peer0` and `peer1`) for endorsement and communication;
- A certificate authority (CA) for identities;
- One or more anchor peers to enable cross-organisation data sharing.

There was also an ordering service node (`orderer.example.com`) shared by all organisations for transaction ordering and block creation.

To encourage carriers, who often resist new infrastructure, we introduced `carrier.example.com` as a shared interface. It was managed and paid for by the manufacturers so carriers could join without hosting their own infrastructure.

Table 5.2: Network node roles per organisation in the shared railcar solution

Organisation	Node	Note
org1.example.com	peer0.org1.example.com	Anchor peer
	peer1.org1.example.com	Endorser
	ca.org1.example.com	Certification authority
org2.example.com	peer0.org2.example.com	Anchor peer
	peer1.org2.example.com	Endorser
	ca.org2.example.com	Certification authority
carrier.example.com	peer0.carrier.example.com	Anchor peer
	peer1.carrier.example.com	Endorser
	ca.carrier.example.com	Certification authority
All	orderer.example.com	Orderer

For our real-world, production-grade network implementation, we chose the IBM Blockchain Platform available on IBM Cloud. This decision allowed us to avoid managing on-site servers and provided us with a set of development and management tools.

The IBM Blockchain Platform uses a pricing model based on Virtual Processor Core (VPC) hours. During our implementation, the rate was $0.29 USD per VPC-hour. We could deploy resources gradually, keep track of use in real time, and fine-tune performance settings dynamically.

Using a BaaS solution significantly lowered our setup and maintenance time. It also gave us a central dashboard to monitor, scale, and manage everything.

While implementing, we tested two realistic usage cases to estimate the load and plan resources:

- **Best Case Scenario: Aggregated Shipment Requests**
  In the optimal use case, car manufacturers sent all information related to a shipment (e.g., vehicle details, delivery destination, carrier data) in a single blockchain transaction. Based on historical data:
  - Each manufacturer managed approximately 10,000 shipments per year,
  - With 2 manufacturers, the network processed 20,000 write transactions per year,
  - Including additional logistics metadata, total write operations reached 30,000 transactions/year,
  - This corresponded to an average of ~ 83 transactions per day.
- **Worst Case Scenario: Per-Vehicle Shipment Requests**
  In a more granular scenario, manufacturers submitted one request per vehicle within a shipment:
  - Each manufacturer handled approximately 171,000 car movements/year,
  - Doubling for two manufacturers yielded 342,000 transactions/year,
  - With shipment-level metadata, the total reached 352,000 write transactions/year,
  - Averaging ~ 965 transactions per day.

In both scenarios, we tested read transactions (to retrieve statuses) and extra write operations (like editing or cancelling shipments). Our load tests showed that IBM Blockchain Platform's auto-scaling could handle spikes while making costs more efficient during slow periods.

Implementation of the Smart Contract

At the core of the railcar-sharing system is a dedicated chaincode that manages capacity allocation, route participation, and cost distribution among manufacturers. Implemented in Java, it ensures consistent state transitions and enforces collaborative logistics logic. Key components include:

- **Data Structures**:
  - `Railcar`: Holds metadata such as total capacity, departure schedule, route segments, and assigned manufacturers.
  - `Shipment`: Represents a unique request for vehicle space initiated by a manufacturer.
  - `Reservation`: Tracks individual manufacturers' bookings against a shipment, including reserved capacity and share of cost.
  - `OccupancyState`: Maintains real-time availability per railcar to support concurrent booking attempts.
- **Transaction Functions**:
  - `CreateShipment`: Registers a new shipment request with associated vehicle capacity requirements and assigns a unique identifier.
  - `JoinShipment`: Allows additional manufacturers to join an existing shipment if capacity is available.
  - `ReserveCapacity`: Updates reservation data, adjusts cost allocation, and enforces capacity constraints to prevent overbooking.
  - `FinalizeSettlement`: Calculates each participant's share of the cost post-departure, factoring in actual load ratios and route-related charges.
- **Endorsement Policies**: Critical functions such as settlement and shipment creation are protected by endorsement policies requiring signatures from multiple organisations. This safeguards against unilateral modifications to booking or cost data.
- **State Management**: Each railcar is stored as a JSON object within the ledger. Hyperledger Fabric's versioning ensures that only the first valid booking transaction is accepted when multiple parties attempt to reserve the same space concurrently, maintaining data integrity during high contention periods.

Supporting Infrastructure: Java Server and Angular Client

Although not absolutely necessary for testing, we built a Java server and an Angular client to improve usability and support enterprise integration.
The Java server:

- **Encapsulates Fabric SDK Logic**: Minimizes direct Fabric calls from the front end, reducing the complexity of membership or certificate handling.
- **Provides REST Endpoints**: Letting users "reserve capacity," "publish availability," or "generate an invoice." These map to chaincode actions or queries.
- **Manages Session and Identity**: Links Fabric-issued identities with a broader corporate Single-Sign-On (SSO) if needed.

On the front end, an Angular app provides:

- **Dashboard Pages**: Show departure times, railcar occupancy, and cost metrics.
- **Interactive Booking Tools**: Let users specify how many vehicles to ship, the dates, and the route segment.
- **Notifications and Alerts**: Use WebSockets to display status changes like new route openings or finalised bookings.

These front-end features make the system easier for real operators to use. Without them, the blockchain solution could remain too technical or hard to fit into everyday workflows.

5.1.4 Evaluation and Analysis

We reviewed many areas: costs for setup and ongoing use, performance, security, and legal compliance. This big-picture view was important for moving beyond a pilot into a real system that connects multiple manufacturers and carriers on a single platform.

Pilot Evaluation Metrics:

- **Average Invoice Settlement Time:** Reduced from 12 days to 3.5 days.
- **Dispute Resolution Rate:** Improved by 68% compared to the pre-pilot baseline.
- **User Satisfaction Score:** 84% rated the platform as effective or highly effective.
- **Onboarding Time per Participant:** Averaged 2.1 days with guided deployment templates.

Deployment and Maintenance Costs

Rolling out a blockchain solution in automotive rail involved up-front spending on design, pilot testing, and later expansion. Key cost factors were:

- **Platform Setup**: Installing Fabric networks that include orderers, peers, and multiple organisations. Some chose on-premises servers, others used cloud-based systems.
- **Chaincode Development and Testing**: Crafting or adapting the chaincode for cost-related logic, partial capacity sharing, endorsement requirements, and optional private data collections.
- **Middleware Development**: Implementing the Java REST server and integration tools for carrier scheduling or ERP systems.

- **Front-End Development**: Building Angular (or other) clients for dashboards and booking capabilities.
- **Training and Onboarding**: Employees needed to learn peer operations, certificate management, chaincode logs, and ledger updates.

Once live, **operational costs** included:

- **Infrastructure Maintenance**: Payment for cloud service (or on-site servers), plus node monitoring for reliability.
- **License Fees or Subscriptions**: If using enterprise support or managed services.
- **Upgrades and Patches**: Routine chaincode revisions, Fabric updates, or new consensus strategies.

One advantage seen in the PoC was automatic resource scaling from the IBM Blockchain Platform. By tracking transaction throughput and VPC stats, we tuned the setup to scale resources dynamically:

- Handling spikes during peak periods (e.g., quarter-end shipping surges),
- Saving money during low-activity times,
- Maintaining high reliability and fault tolerance without overprovisioning.

This proved that the design could be both practical and cost-effective for real-world use. A shared-cost model, where manufacturers fund a joint infrastructure, also encouraged more carriers to join the network.

Scalability and Performance Testing

Scalability was crucial because car manufacturers could ship hundreds of thousands of vehicles a year. If partial wagon use prompts multiple updates, the ledger must handle them quickly. Fabric can process hundreds to thousands of transactions per second under good conditions, but real performance depends on endorsements and chaincode complexity. Our approach to scaling included:

- **Parallel Channels**: Splitting data by route or organisation to reduce contention.
- **Chaincode Optimisations**: Keeping read/write sets as small as possible and avoiding non-deterministic code.
- **Gossip Tuning**: Adjusting block size, batch durations, and membership settings so block distribution isn't a bottleneck.

We ran load tests simulating peak times, like quarter-end production surges. If the network performance was too low, we modified certain parts or looked into advanced scaling (e.g., sidechains or sharding), though those add challenges when channels require privacy.

Performance Results on IBM Blockchain Platform

We deployed a 2–Org, 2-peer per Org Fabric network with Raft ordering on IBM Blockchain Platform and measured performance under a simple `RailcarAsset` chaincode using both built-in monitoring and Hyperledger Caliper.

- **Testbed:**
  - Peers: 4 (0.7 vCPU, 2 GB RAM each)
  - Orderers: 3 (Raft)
  - Endorsement: Org1 $\wedge$ Org2
  - Chaincode: Java
- **Workload:**
  - Write transactions (asset updates)
  - Peak offered load: 200 TPS
- **Monitoring:**
  - IBM Console Metrics
  - IBM Cloud Monitoring / Prometheus & Grafana
- **Benchmarking:** Hyperledger Caliper

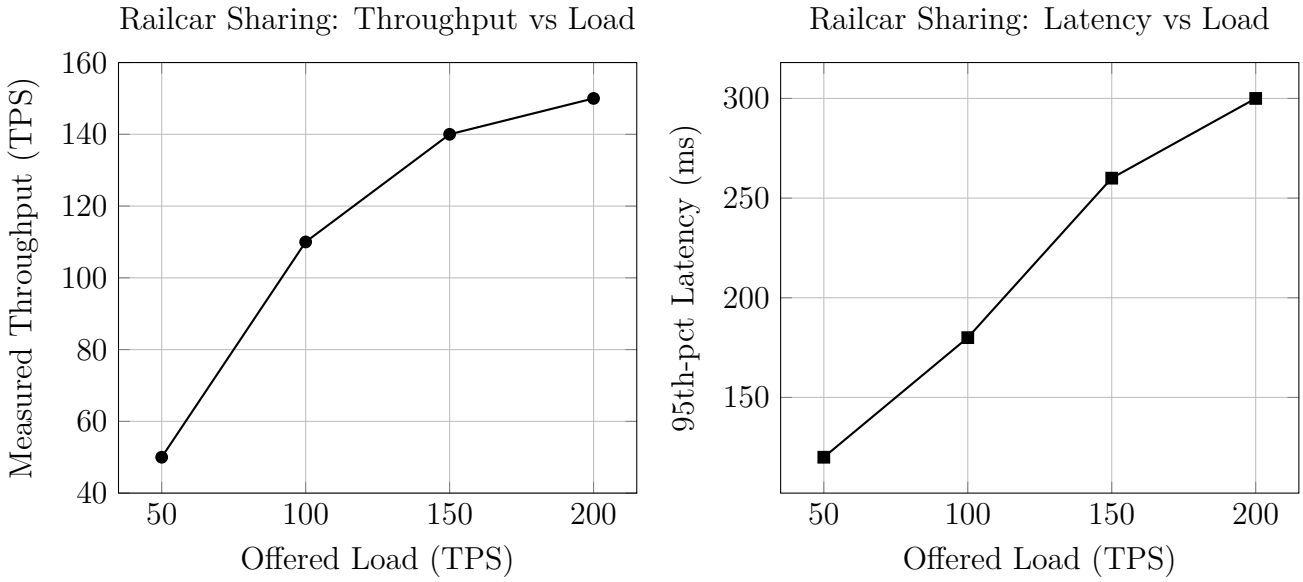
Table 5.3: Key Performance Metrics (Railcar Sharing)

Metric	Value	Unit	Notes
Throughput (sustained)	50–150	TPS	under 200 TPS load
End-to-end latency (95th pct.)	100–300	ms	
Peer CPU utilization	50–70	% of 0.7 vCPU	at peak load
Peer memory consumption	1–2	GB	chaincode in Go
Transaction success rate	99–100	%	no overload, endorsement met

5.1.5 Security and Privacy Assessment

Security is critical in a system that brings together multiple companies, some of which are competitors. Potential risks include compromised credentials, errors in private data setup, or vulnerabilities in chaincode logic. Hyperledger Fabric addresses these with:

- **Permissioned Access**: MSP-based certificates ensure only recognised organisations can run peers or send transactions.
- **Endorsement Policies**: Certain ledger updates require endorsements from multiple separate organisations, blocking single-actor tampering.
- **Immutable Chain**: Once data is in a block, it cannot be quietly removed or altered, allowing audits to detect anomalies.



(a) Throughput under increasing load

(b) Latency under increasing load

Figure 5.9: Railcar Sharing: Load-test results on IBM BP

Still, companies also focused on:

- **Physical Security of Nodes**: If nodes run in various data centers, each location must be protected from hardware intrusion.
- **Chaincode Vulnerabilities**: Logic bugs might miscalculate costs or produce conflicting data.
- **Privacy Strategies**: For sensitive data, a mix of private data collections, channel-based segmentation, and encryption was used.

So security testing included penetration tests, code reviews of chaincode, revoking certificates to see if that worked, and scenario-based checks (like one dishonest node trying to falsify records). Good governance meant regular certificate rotations, with only authorised staff holding admin rights.

5.1.6 Regulatory and Ethical Considerations

Rail logistics is subject to various laws, and blockchain doesn't remove those obligations. It can give better traceability but also adds some new complexities:

- **Data Protection**: Some regions require protection of personal data (e.g., driver info). Because such data might appear on the ledger, compliance with GDPR or local privacy rules is achieved by data minimisation or encryption.
- **Antitrust Concerns**: A railcar-sharing group among manufacturers must not become price-fixing or anti-competitive. The open ledger approach encourages fair cooperation without sharing overly strategic data.
- **Cross-Border Shipments**: When routes cross regions, customs might require partial data disclosure, so we use specialised channels or proofs that verify cargo authenticity without exposing unnecessary data.

Collaborating can prompt concerns regarding fairness for smaller manufacturers. Our framework ensures these manufacturers are not marginalized due to chaincode mechanics. An on-chain dispute resolution mechanism aids in resolving issues related to costs or timings, preventing prolonged legal disputes. Additionally, our goal is to maintain an inclusive consortium, allowing smaller or newer manufacturers to join if they fulfill membership requirements, thereby avoiding a monopoly of only the largest players.

5.2 Shared Procurement

In this section, we provide a comprehensive examination of a shared procurement use case based on Hyperledger Fabric smart contracts. The scenario focusses on two different organisations that retain ongoing operational ties through common investments and shared expenses. Such situations arise, for instance, when a company spins off a subsidiary (or merges with a partner), yet continues to coordinate certain procurement activities. By setting up a unified blockchain infrastructure, these entities have reduced administrative overhead, mitigated inter-company disputes regarding expense allocations, and automated procurement workflows. The increasingly distributed nature of corporate structures across industries, including manufacturing, healthcare, and technology, underscores both the relevance and the timeliness of this problem.

In order to clarify this use case, we begin by analysing the current ('as-is') process from the perspectives of both organisations and then highlight the key stakeholders involved. We then examine the business need, that is, the rationale for transitioning to a blockchain-based model. Following this, we show how a future state ('to-be') design based on Hyperledger Fabric has been deployed to eliminate redundancy and enhance trust within the procurement life cycle. Subsequently, details on system design and implementation are provided, focussing on chaincode logic, a Java-based back-end server for orchestrating processes, and an Angular client for end-user interaction. Finally, we evaluate costs, performance, security, scalability, and regulatory matters, concluding with a critical reflection on the practical suitability of the approach and potential extensions.

The motivation behind the problem was the absence of a centralised system to complete and reconcile procurement data. Since the two organisations lacked a unified ledger, they experienced duplication efforts in invoice processing, misunderstandings of partial payment arrangements, and a labour-intensive reconciliation stage at every financial closing. By entrusting these functionalities to a blockchain-based solution, the potential for after-the-fact tampering or misalignment of data was greatly reduced. The system provided an immutable ledger, combined with peer-based validation and chaincode-based logic, strengthening the procurement life cycle and offering tangible cost and accuracy benefits. In the pages that follow, we dissect all relevant components of this integrated solution, drawing on existing academic discourse on blockchain systems in supply chain, procurement, and coalition-based enterprise resource planning (ERP).

5.2.1 Current State ('as-is' Process) and Business Case

The initial step in evaluating the feasibility of implementing a blockchain-based procurement platform necessitates an in-depth understanding of the extant issues and inefficiencies inherent in current processes. Historically, each of the two companies, designated here as Company A and Company B, have utilised their own distinct internal procurement regulations, systems, and invoice management methodologies. Notwithstanding a separation in some form, such as a spin-off, asset sale, or partial divestiture, both entities continue to collaborate frequently in joint purchasing endeavors for shared supplies, organizational events, and personnel training sessions.

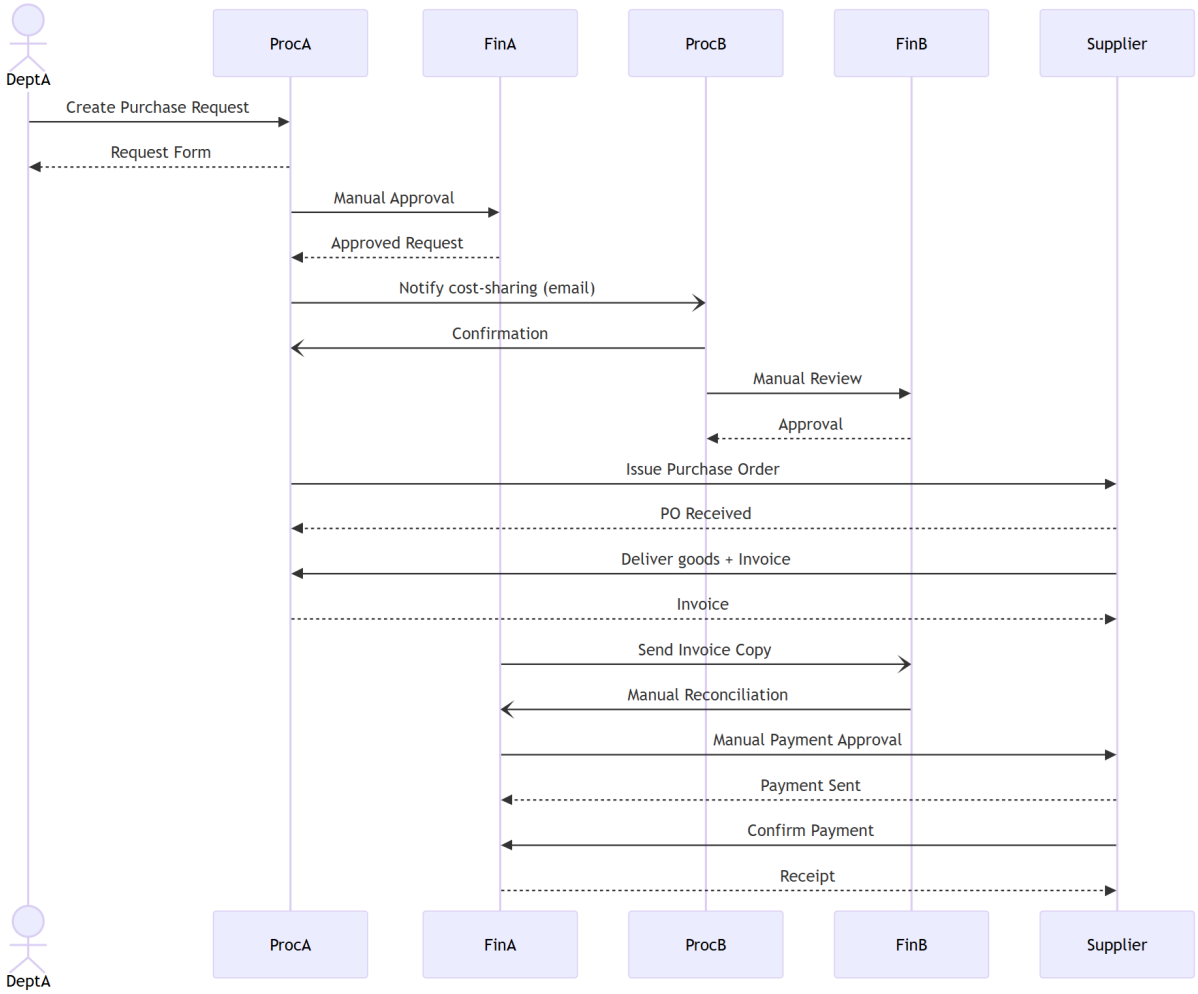


Figure 5.10: Shared Procurement: as-is process sequence diagram.

Typical friction points in the current process:

- **Manual invoice matching** across systems led to delays and human error.
- **Data siloing** between companies resulted in inconsistent or outdated records.
- **Cost-sharing disputes** arose when there was ambiguity in the allocation of shared expenses.
- **Lack of a shared audit trail** complicated reconciliation and eroded mutual trust.

Because companies did not share a single ERP or procurement system, reconciling the question of who spent what, how costs should be allocated, and whether the agreed formula for cost sharing was followed became increasingly difficult.

When an expense was incurred by one company on behalf of the other, perhaps purchasing a bulk order of office supplies or paying for a shared marketing initiative, the purchaser had to invoice the other party after the fact. Key data points such as invoice number, approval date, or cost centre reference were often stored in each company's siloed database or ERP. Any discrepancy between these records caused confusion as to who was responsible for a given expense. Such disputes were a recurring obstacle each month

or quarter, requiring additional managerial time to resolve, diminishing operational harmony, and straining the partnership.

The prevailing conditions underscored the necessity for an integrated digital system. In the absence of a unified, tamper-resistant repository of shared procurement events, stakeholders were compelled to make managerial and financial decisions predicated on incomplete or somewhat outdated information. The ledger presently facilitates real-time updates with each occurrence of order creation, approval, modification, or payment. Within this framework, disputes concerning cost allocation or partially fulfilled purchase requests have been significantly curtailed, as stakeholders are presented with an identical set of records. By adopting a blockchain-based solution for these functions, the likelihood of post hoc tampering or data misalignment is substantially diminished.

Stakeholders and Participants

The implementation of a blockchain-based procurement platform affects multiple stakeholder groups both within and outside the two involved companies. Each participant interacts with the system differently, depending on their responsibilities and level of access. The primary categories of stakeholders include:

- **Procurement Officers:** These stakeholders initiate purchase requests, manage approval chains, and oversee order reconciliation across entities. They are among the most frequent users of the platform.
- **Department Managers:** Responsible for approving procurement requests within their respective domains, these individuals ensure that spending aligns with internal budgets and strategic goals.
- **Finance Departments:** These teams reconcile cost allocations, monitor compliance with intercompany agreements, and manage invoicing between the two organisations. They depend on synchronised, tamper-proof records to verify that the agreed cost-sharing formulas have been applied.
- **Suppliers and Third-party Vendors:** Although external to the companies, these stakeholders benefit from improved communication and access to shared records. With proper permissions (e.g., read-only interfaces), they can directly check order statuses, shipment confirmations, or delivery logs, reducing reliance on back-and-forth emails or phone calls.
- **Senior Management and Executive Sponsors:** These stakeholders use dashboards and aggregate views for strategic oversight. They may revise cost-sharing rules, define approval thresholds, or reconfigure governance logic when corporate structures change (e.g., additional spin-offs or mergers).

Each stakeholder category brings distinct interests and levels of influence to the platform. The matrix below summarises their roles and what they gain or risk from adopting the blockchain solution.

Table 5.4: Shared Procurement: Stakeholder Roles and Interests Matrix

Stakeholder	Interest / Influence
Procurement Officers	Initiate and approve purchase orders. Interested in faster approval cycles, accurate shared records, and less reliance on emails or manual tracking. High influence on daily transaction flow.
Finance Departments	Ensure correct cost-splitting, reconciliation, and compliance with intercompany agreements. Interested in data consistency and auditability. Blockchain helped to reduce disputes and improve reporting accuracy.
Suppliers / Third-party Vendors	External participants who benefit from better transparency in order status and fewer communication bottlenecks. Gaining read-only access or interface-based visibility into their transactions. Low influence, but high operational impact.
Senior Management / Executives	Use aggregate dashboards for strategic decision making and budget forecasting. Influence policy-level rules such as cost-sharing formulas or approval thresholds. Blockchain provides trusted metrics for governance and restructuring.

Asset and Transaction Analysis

Multiple definitions of "assets" and "transactions" exist in the field of blockchain-based procurement, but we focus on four main aspects.

1. **purchase requests** have been treated as easy-to-track digital records, representing an internal department's need to acquire certain goods or services. Each request includes details such as item descriptions, quantities, recipient departments, and proposed cost allocations between the two companies. These requests are eventually subject to approval workflows involving designated managers or administrators.

2. **purchase orders** (POs) function as binding commitments to place an order with a supplier. In the previous "as-is" process, a PO was created in the internal system of one party and had to be communicated to the other party if cost sharing was expected. A mismatch between PO references or incomplete transfer of PO data often resulted in incorrectly or unsettled transactions. By placing these POs on a shared ledger, each participant verifies the details of the order, including contractual obligations and the portion of costs attributed to each party.

3. **budget allocations and cost centres** are critical assets that represent how each organisation accounts for a transaction from a financial perspective. Managers ensure that any shared expense is charged to the correct cost centres in each company's structure. By storing cost-centre references on the blockchain, the system helps unify naming conventions or code sets, ensuring that Company A cost centre 100-X is consistently associated with the corresponding code in Company B's system, say 200-Y.

4. **invoice confirmations** or **payment confirmations** represent the final transactions in the procurement life cycle. The two organisations must reconcile these

confirmations so that payment is delivered to vendors and the correct journal entries are created in each finance system. Previously, these confirmations required repeated manual checks, sometimes involving an exchange of physical documentation. Ensuring that the final status of each invoice is recorded on a distributed ledger automates much of the accounts payable process, reducing lead times and error margins.

Business Impact

The earlier situation posed multiple business obstacles, from direct cost overhead to less tangible impacts such as strained relationships and operational inefficiencies. From a cost perspective, time spent by accountants on reconciling data, resolving disputes, or chasing missing invoices was substantial. These tasks diverted resources from core finance functions, generating higher operational overhead. Meanwhile, departments risked overspending without accurate real-time data on shared budgets.

The new blockchain-based approach has alleviated many of these issues. Aggregated spending data are now available to both companies in near-real-time, improving forecasting and budget controls. Disputes regarding partial charges are rare, because both parties draw on the same ledger records. Furthermore, suppliers appreciate streamlined interactions, as purchase orders, acknowledgment of deliveries, and final invoices are consistently logged in a shared, tamper-resistant format.

In highly regulated industries, the newfound ability to produce a time-stamped, tamper-resistant record of all purchase events supports compliance and can preempt costly audits or penalties. By contrast, the former system relied on a patchwork of spreadsheets, PDFs, or ad-hoc ERP extracts that were difficult to align. Taken together, these improvements reduce friction in joint procurement operations, reinforcing trust across company boundaries.

5.2.2 Proposed Blockchain-Based Solution ('to-be' Process)

The 'to-be' process leverages Hyperledger Fabric to manage procurement transactions between the two cooperating companies. The solution highlights key blockchain benefits: distributed consensus, immutability, and programmable chaincode logic. By implementing a smart contract that automates cross-company approval flows, real-time cost splitting, and final settlement confirmations, the new system replaces or greatly reduces manual tasks prone to errors with a trustworthy and decentralised mechanism.

The solution directly resolves the deficiencies of the 'as-is' approach. Rather than relying on duplicated data entry and after-the-fact invoices, each transaction (whether it is a purchase request, partial approval, or a final invoice payment) becomes a unique record in the shared ledger. This record is validated by the peers of both organisations before being appended to the blockchain. If approval thresholds are not met or if cost-splitting formulas violate pre-set rules, the chaincode rejects the transaction. This enforcement ensures that no participant can unilaterally update procurement data, thereby minimising the potential for internal procedural disputes.

A permissioned blockchain network, such as Hyperledger Fabric, suits the enterprise context, thanks to its flexible data privacy controls. Companies A and B defined a private channel for sensitive procurement details, ensuring that only authorised peers accessed the ledger data. If a third organisation or a specific set of external stakeholders joined

the consortium later, the channel architecture was reconfigured without compromising previous transactions or their integrity.

By encoding all relevant business rules in chaincode (e.g., cost allocation schedules, approval hierarchies, invoice finalisation logic), the solution enforces consistent protocols across both organisations. Data transfers to legacy ERPs are facilitated through standardised APIs, allowing each company to maintain internal processes with minimal disruptions. With every event stored on-chain, the system captures a transparent, real-time audit trail visible to authorised participants.

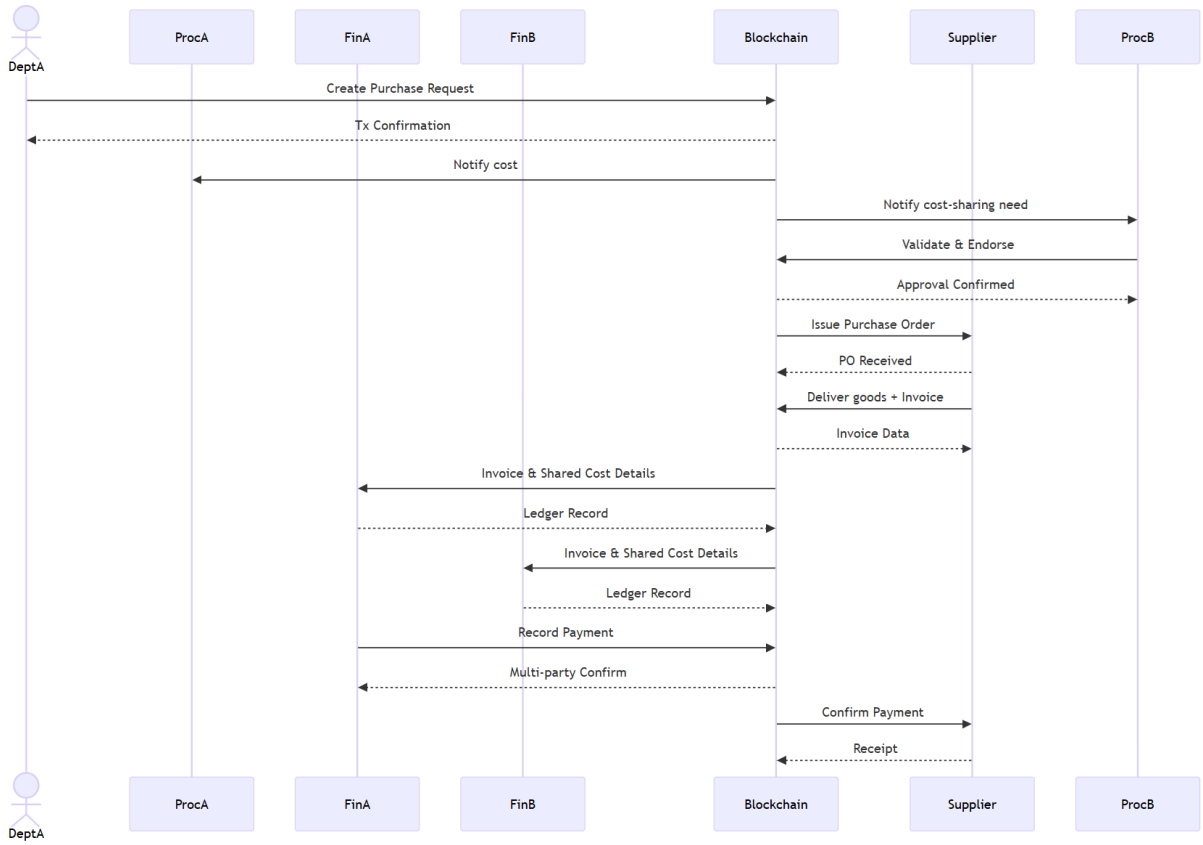


Figure 5.11: Shared Procurement: to-be process sequence diagram.

Solution Canvas

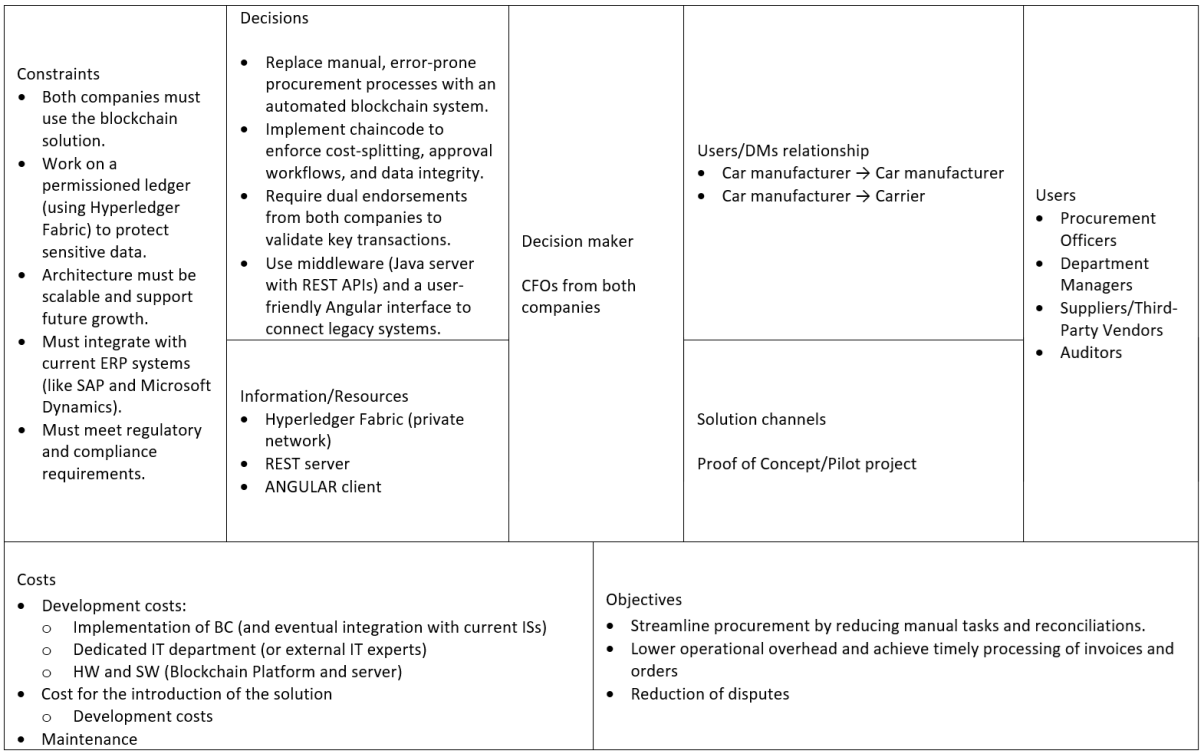


Figure 5.12: Shared Procurement: solution canvas.

Network Conceptual Design

A high-level conceptual design includes multiple layers:

- **Infrastructure Layer:** Hyperledger Fabric nodes (peers) operated by both Company A and Company B. Each company administers its own Certificate Authority (CA) to issue identities to users and services.
- **Network and Governance:** A blockchain consortium in which Companies A and B share membership. Consortium rules define endorsement policies and data privacy requirements.
- **Smart Contract Layer:** Chaincode modules orchestrate the creation, approval, and settlement of purchase requests and orders. These modules codify cost-splitting formulas (e.g., 50/50 or proportional splits).
- **Integration Layer:** APIs connect the chaincode layer to each company's internal systems. Approved requests and invoice data are pushed to local ERPs or finance systems.
- **User Interaction Layer:** A front-end application guiding authorised personnel through request creation, approvals, dispute handling, and settlement steps, complete with dashboards summarising cross-company expenses.

In the initial MVP, a simplified approval workflow was created, in which one manager from each company had to approve purchase requests before they became purchase orders,

and a basic expense-splitting policy was enforced. Over time, the chaincode was extended to handle flexible policies, including tiered signatories and conditional approvals.

Integration with Legacy Systems

Implementing the solution required interoperability with existing ERP and procurement platforms. Rather than displacing legacy systems, the blockchain network operates as a synchronisation layer that ensures cross-company events remain consistent.

During implementation, application adapters or microservices were developed using the Hyperledger Fabric SDK. These adapters translate RESTful API calls from an ERP system into blockchain transactions. For example, when a procurement officer enters a new purchase request in Company A's SAP instance, the adapter forwards the request to the chaincode. According to the endorsement policy, a validating peer from Company B endorses the transaction. Once validated and committed, the blockchain notifies the SAP instance, ensuring local records match the ledger state. Similar flows exist for invoice issuance, cost-splitting modifications, or final payment confirmations.

This interoperability facilitated a phased rollout without requiring abrupt changes to established ERP solutions. Finance managers can still view or reconcile data in their familiar local system, confident that intercompany records align in real time on the underlying distributed ledger.

5.2.3 System Design and Implementation

The design and implementation work for the shared procurement use case encompassed deploying a Hyperledger Fabric network, configuring channels for relevant data exchange, developing chaincode (smart contracts) to handle procurement logic, and constructing user-facing components. Below, we outline the fundamental architectural considerations, the chaincode design, and the supporting infrastructure that together create a production-ready environment.

A core principle of the deployment was modularity: chaincode logic, back-end server, and front-end code were cleanly separated. This approach minimised ripple effects across components whenever a feature was added or changed. It also aligns with best practices for software reuse, enabling incremental enhancements to the cost-splitting logic or role-based permissions without forcing a complete rebuild of the user interface.

Overall System Architecture

The network topology features:

- **Hyperledger Fabric Peers:** Company A and Company B each host at least one peer node, which validates transactions and maintains the ledger state.
- **Certificate Authorities:** Each organisation manages digital identities for its employees, administrators, and services.
- **Ordering Service:** A fault-tolerant ordering service (e.g., Raft) that batches transactions into blocks and disseminates them to peers.
- **Java Server:** Responsible for constructing transactions, gathering endorsements, and interacting with corporate identity solutions.

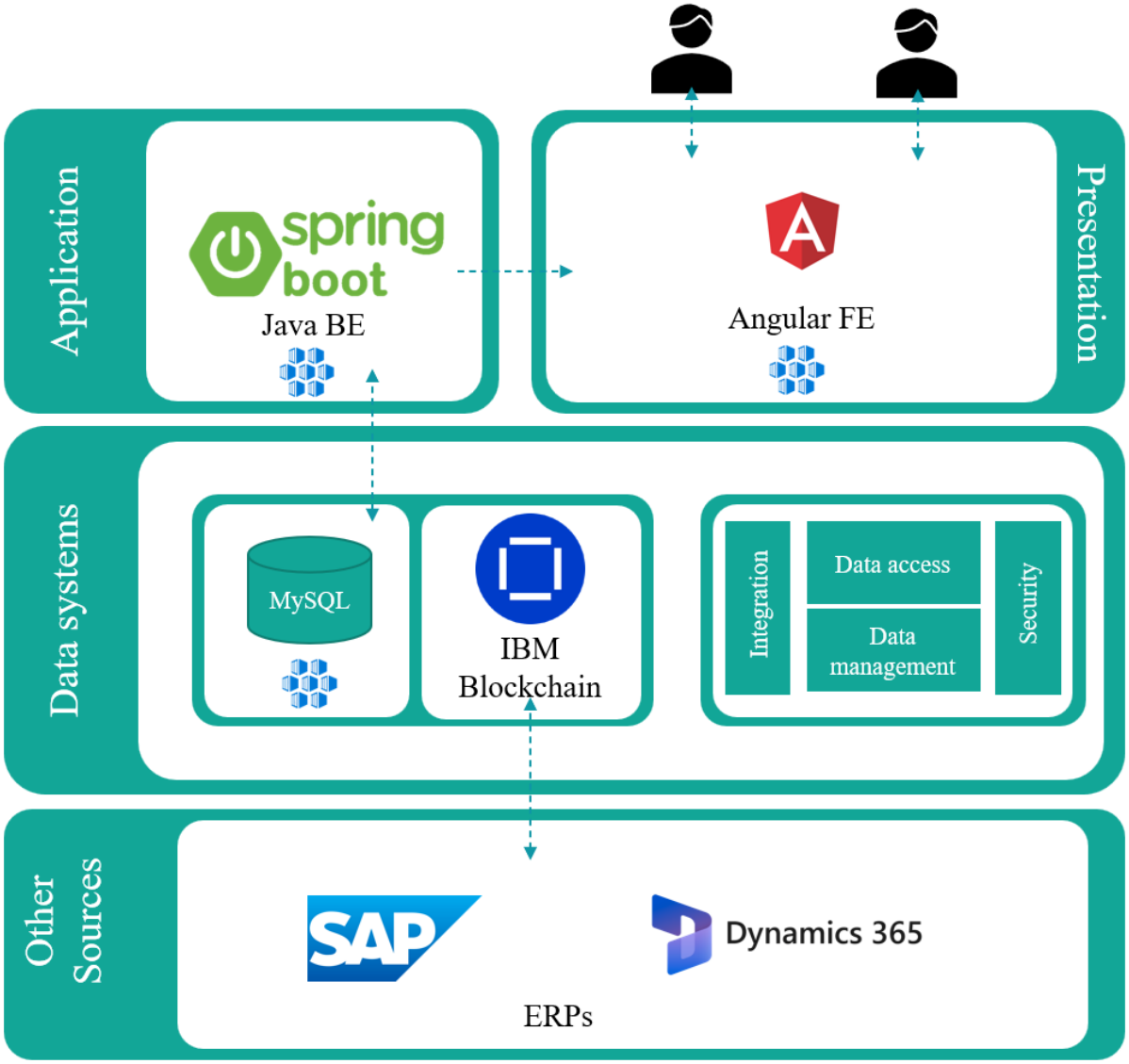


Figure 5.13: Shared procurement: high level software architecture.

- **Angular Front-end:** A web application that acts as the user-facing portal, facilitating actions such as "create purchase request," "approve cross-company expense," and "view invoice status."

Once a user initiates an action via the Angular front-end, the Java server packages the request as a Fabric transaction proposal, sends it to endorsing peers, and finally submits it to the ordering service. If the transaction meets endorsement policy requirements, it is appended to a new block and broadcast to all peers. The front-end then updates the user interface accordingly.

IBM Blockchain Platform for PoC

We used IBM Hyperledger Fabric to build the blockchain network and deployed it on a managed Blockchain-as-a-Service (BaaS) platform. This approach simplified infrastructure management and provided enterprise-grade scalability, reliability, and monitoring.

The network we set up included three main organisations:

- `companyA.example.it` and `companyB.example.it`, representing the two collaborating companies;
- `supplier.example.it`, a shared entity representing key suppliers or third-party vendors.

In each organisation, we set up the necessary Fabric components:

- Two peer nodes (`peer0` and `peer1`) for endorsement and communication;
- A certificate authority (CA) for identity management;
- At least one anchor peer to enable cross-organisation data sharing.

There was also an ordering service node (`orderer.example.it`) shared by all organisations for transaction ordering and block creation.

To encourage supplier participation—since suppliers are often reluctant to manage new infrastructure—we introduced `supplier.example.it` as a shared interface. This node was managed and paid for by the two companies, allowing suppliers to join the network and access procurement data without hosting their own infrastructure.

Table 5.5: Network node roles per organisation in the shared procurement solution

Organisation	**Node**	**Note**
companyA.example.it	peer0.companyA.example.it	Anchor peer
	peer1.companyA.example.it	Endorser
	ca.companyA.example.it	Certificate Authority
companyB.example.it	peer0.companyB.example.it	Anchor peer
	peer1.companyB.example.it	Endorser
	ca.companyB.example.it	Certificate Authority
supplier.example.it	peer0.supplier.example.it	Anchor peer
	peer1.supplier.example.it	Endorser
	ca.supplier.example.it	Certificate Authority
All	orderer.example.it	Orderer

As done for the railcar sharing use case, to support the network implementation in a real-world, production-grade environment, we opted for IBM Blockchain platform. This eliminated the need for on-premise server management and provided a central dashboard for monitoring, scaling, and managing the network.

Using a BaaS solution significantly reduced setup and maintenance time, and provided robust tools for network management and scaling.

During implementation, we tested two realistic usage scenarios to estimate load and plan resources:

- **Best Case Scenario: Aggregated Procurement Requests**
  In the optimal use case, procurement officers submitted all information related to a purchase (e.g., items, cost allocation, approvals) in a single blockchain transaction. Based on historical data:

  - Each company processed approximately 5,000 shared procurement events per year,
  - With 2 companies, the network handled 10,000 write transactions per year,
  - Including supplier-side updates, total write operations reached 12,000 transactions/year,
  - This corresponded to an average of ~ 33 transactions per day.

- **Worst Case Scenario: Per-Item Procurement Requests**
  In a more granular scenario, each line item in a purchase order was submitted as a separate request:

  - Each company managed approximately 40,000 line items/year,
  - Doubling for two companies yielded 80,000 transactions/year,
  - With supplier-side confirmations, the total reached 90,000 write transactions/year,
  - Averaging ~ 247 transactions per day.

In both scenarios, we also tested read transactions (to retrieve order statuses) and additional write operations (such as approvals, modifications, or payment confirmations). Our load tests showed that the BaaS platform's auto-scaling features could handle transaction spikes efficiently, optimizing costs during periods of lower activity.

Implementation of the Smart Contract

At the core of the shared procurement application is the chaincode handling purchase requests, approvals, and final invoice records. Implemented in Java, it manages state transitions through well-defined functions. Key components are:

- **Data Structures**:
  - `PurchaseRequest`: Contains a unique ID, item descriptions, quantities, cost centre references, and the cost-splitting approach.
  - `PurchaseOrder`: Generated from an approved request, including finalised supplier details and cost breakdowns.
  - `InvoiceRecord`: Logged once the supplier's invoice is paid, automatically reflecting any shared expense splits.

- **Transaction Functions**:
  - `CreateRequest`: Stores a new request on the ledger.
  - `ApproveRequest`: Marks a request as approved if the user's role and threshold conditions are met.
  - `GeneratePO`: Moves a request from approved state to a purchase order.
  - `RecordPayment`: Finalises an invoice record, allocating costs to each party's portion.

- **Endorsement Policies**: The chaincode requires endorsements from both companies for key transactions (e.g., issuing a major purchase order), ensuring no single party can alter the procurement ledger unilaterally. Configuration files define which peers must sign off for a transaction to be considered valid.

- **State Management**: Hyperledger Fabric enforces version checks on ledger keys, preventing conflicting updates if two users attempt to modify the same request concurrently. This concurrency control ensures reliable operation even under high transactional load.

Supporting Infrastructure: Java Server and Angular Client

A Java-based back-end was developed to streamline identity management and facilitate interactions with the underlying chaincode. Authentication is managed through single sign-on (SSO), ensuring that a user's organisational role is verified before any transaction is submitted. Domain-level actions—such as *"I want to allocate 40% of this invoice to Company B"*—are automatically translated into cryptographically signed proposals for Fabric peers.

The Angular front-end provides a role-sensitive interface:

- **Procurement officers** can view and act only on requests they are authorised to create or approve.

- **Finance managers** access consolidated expense reports and cost breakdowns across companies.

- **Department heads** and users with limited privileges can review cost allocations without needing to understand blockchain logic.

Real-time notifications are displayed when:

- Cross-company approvals are pending.

- An invoice with partial payments changes status.

This layered architecture simplifies the user experience. End-users interact with a familiar web interface and intuitive dashboards, while the Java middleware and Hyperledger Fabric network handle:

- Endorsement logic,

- Role-based permission checks,

- Data immutability, and

- Cross-organisational consistency guarantees.

5.2.4 Evaluation and Analysis

To measure the success of the blockchain-based shared procurement system, a systematic evaluation was conducted, focussing on performance metrics, governance viability, security posture, privacy controls, and regulatory compliance. This evaluation aligned with standard best practices for enterprise blockchain rollouts.

Early pilot programs demonstrated that monthly dispute frequency dropped significantly, invoice settlement times improved, and cost allocation accuracy rose. User feedback indicated that the system was more transparent and user-friendly compared to the previous email and spreadsheet-driven processes, although training was initially needed to familiarise users with the new front-end application.

Key Pilot Evaluation Metrics:

- **Average Invoice Settlement Time:** Reduced from 10 days to 3.2 days.
- **Error Rate in Cost Allocation:** Dropped by 72% thanks to chaincode-enforced logic checks.
- **User Acceptance Rating:** 86% of participants rated the system as "indispensable" post-deployment.
- **Monthly Dispute Frequency:** Declined from 18 cases to fewer than 5 across the same period.

Deployment and Maintenance Costs

A major consideration was total cost of ownership. Setting up Hyperledger Fabric peers, ordering services, and certificate authorities for both organisations required infrastructure investments and staff training. The chaincode had to be thoroughly tested to encapsulate cost formulas and approval rules accurately. Integration with diverse ERP systems—SAP for Company A and Microsoft Dynamics for Company B—was nontrivial, necessitating a custom Java-based middleware.

However, once deployed, operational expenses involved ongoing cloud costs, peer monitoring, and chaincode updates for new business requirements. Training was an ongoing but diminishing overhead as more employees became familiar with the system. Despite these challenges, the reduction in manual reconciliation, dispute resolution, and data fragmentation offset much of the initial capital outlay. Shared governance arrangements were used to split costs fairly.

Scalability and Performance Testing

With the system fully deployed, performance tests examined throughput and latency under realistic loads. Dozens of concurrent users triggered purchase requests or approvals, quickly driving up transaction volume. Thanks to chaincode optimisations, the network sustained several hundred transactions per second without incurring problematic delays. The results remained within acceptable ranges for typical procurement cycles.

Stress tests revealed that peer endorsement policies had the strongest effect on latency, as requiring signatures from multiple peers across geographically distant sites introduced network round trips. Nonetheless, the system scaled horizontally by adding more peers, distributing the endorsement load when needed. Monitoring dashboards tracked CPU

usage, block generation intervals, and transaction commit rates, alerting operators if any performance thresholds were breached.

Performance Results on IBM Blockchain Platform

- **Testbed:**
  - Peers: 4 (0.7 vCPU, 2 GB RAM each)
  - Orderers: 3 (Raft)
  - Endorsement: Org1 ∧ Org2 ∧ Org3
  - Chaincode: Java
- **Workload:**
  - Complex write transactions (PO approval, invoice commit)
  - Sustained offered load: 100 TPS
  - Read-only queries (status checks)
- **Monitoring:** same as Section 5.1
- **Benchmarking:** Hyperledger Caliper

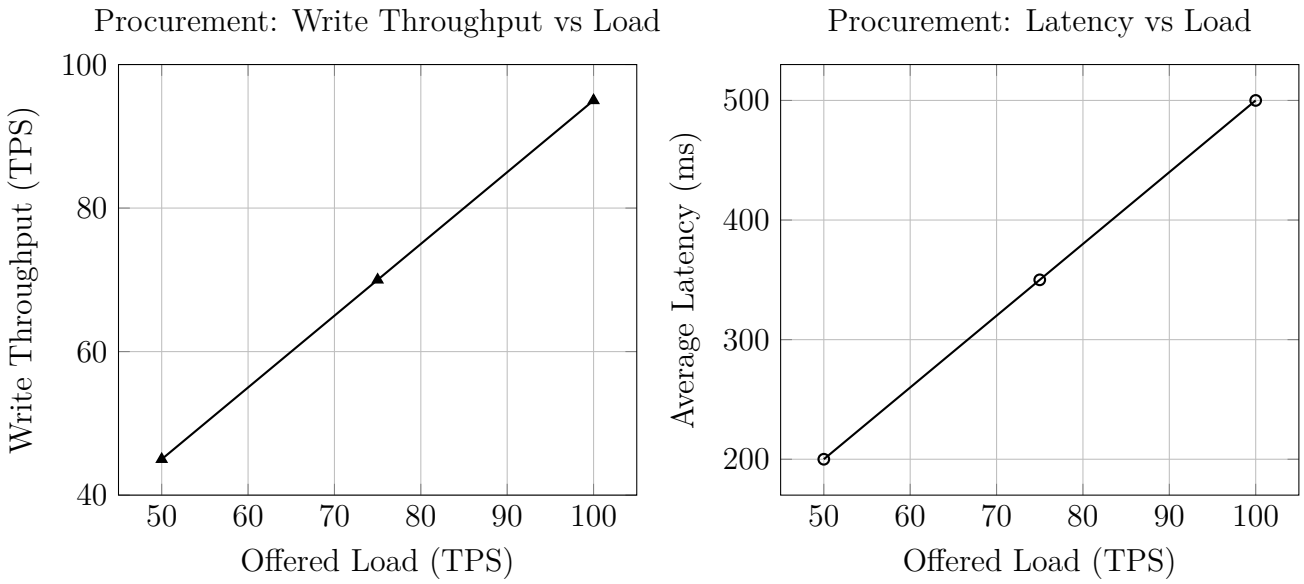
Table 5.6: Key Performance Metrics (Shared Procurement)

Metric	Value	Unit	Notes
Throughput (write)	30–100	TPS	depends on payload size
Throughput (read)	800–1,200	TPS	simple queries, sub-50 ms latency
Average latency (write)	200–500	ms	95th pct. 600 ms
Peer CPU utilization	70–80	% of 0.7 vCPU	Java runtime incurs moderate load
Peer memory consumption	2–3	GB	includes chaincode runtime overhead
Transaction success rate	≈99	%	endorsement failures $\lesssim$0.5 %

As shown in Figure 5.14, the system exhibits a near-linear increase in average write latency as the offered load approaches 100 TPS. This trend reflects the behavior of queueing systems, where incoming transactions accumulate in buffers at the endorsement, ordering, and validation stages. According to Little's Law, the relationship between average latency L, arrival rate λ, and service time W follows $L = \lambda \cdot W$ [24]. When service capacity remains stable, a rising arrival rate results in a proportionate increase in latency. This phenomenon has been empirically confirmed in studies of Hyperledger Fabric [30], and IBM recommends monitoring such queue build-up via Prometheus metrics and Grafana dashboards to detect overload early and trigger resource scaling [21].

5.2.5 Security and Privacy Assessment

Robust security underpins the shared procurement network. Because the consortium includes distinct legal entities, each organisation uses a separate Certificate Authority.



(a) Write throughput under load

(b) Latency under load

Figure 5.14: Shared Procurement: Load-test results on IBM BP

Role-based access control in chaincode ensures that only users with appropriate privileges can implement purchase allocations above certain thresholds or finalise major budget items. Private data collections safeguard confidential pricing and supplier details, allowing only relevant peers to access those entries.

Security testing included static code analysis, chaincode audits, scenario-based penetration tests, and certificate revocation drills. The results confirmed that the permissioned nature of the network, combined with the endorsement policies, makes unilateral ledger manipulation highly unlikely . Potential threats, such as compromised user credentials or insider tampering, were mitigated by multi-factor authentication and consistent chaincode-based validation rules. Logging facilities maintain a forensic trail of all access attempts, easing compliance audits and incident investigations.

5.3 Comparative Analysis: Railcar Sharing vs Shared Procurement

5.3.1 Overview

Both the *Railcar Sharing* and *Shared Procurement* use cases were deployed on the IBM Blockchain Platform using Hyperledger Fabric, leveraging the same base infrastructure—4 peers (0.7 vCPU, 2GB RAM each), a 3-node Raft ordering service, and endorsement policies across organisations. However, despite the architectural similarity, notable differences in throughput, latency, and resource utilization emerged during load testing.

5.3.2 Workload Differences

- **Railcar Sharing** involved simpler write transactions for asset updates (e.g., wagon bookings) with Java chaincode, which led to lower CPU usage and more predictable scaling.

- **Shared Procurement** required more complex multi-party approvals and involved richer payloads and deeper endorsement chains, implemented in Java chaincode.

5.3.3 Performance Comparison

Table 5.7: Summary of Key Performance Metrics

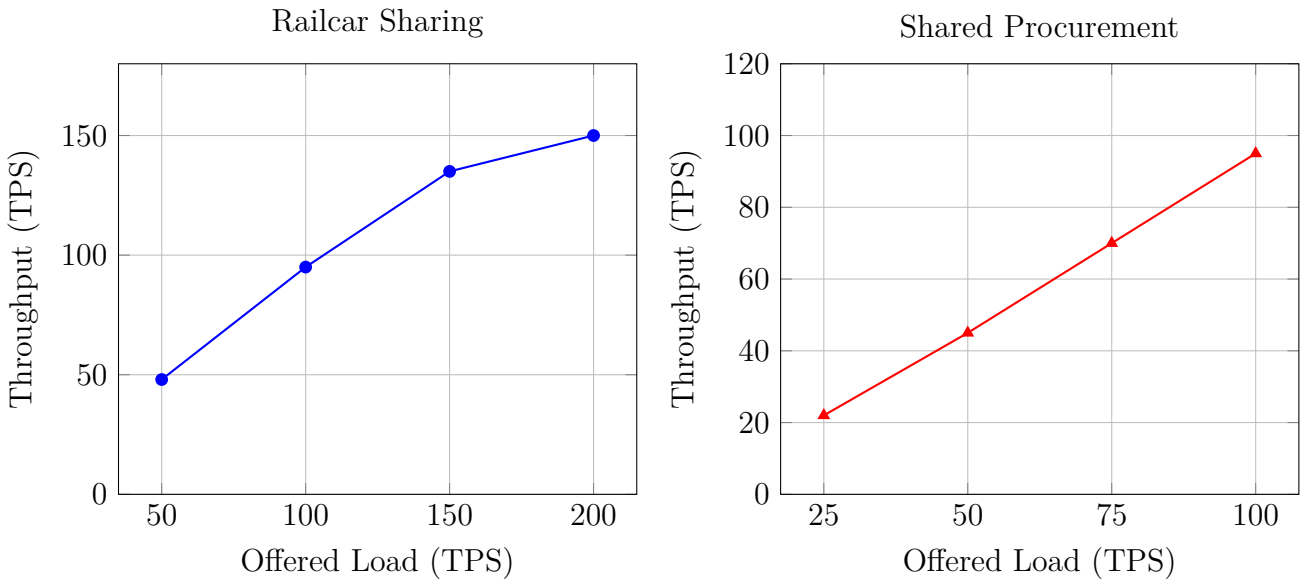
Metric	Railcar Sharing	Shared Procurement
Sustained Throughput (write)	50–150 TPS	30–100 TPS
Peak Offered Load	200 TPS	100 TPS
Average Latency (95th pct.)	100–300 ms	200–600 ms
Peer CPU Utilization	50–70%	70–80%
Peer Memory Usage	1–2 GB	2–3 GB
Success Rate	99–100%	$\approx 99\%$

5.3.4 Latency and Throughput Analysis

The latency profiles differed significantly. In the shared procurement case, write latency exhibited a near-linear increase as load approached the 100 TPS threshold, a typical behaviour of queueing systems approaching saturation. This trend aligns with queueing theory, notably Little's Law, where latency $L = \lambda \cdot W$ increases with higher transaction arrival rates λ, assuming a fixed service capacity W.

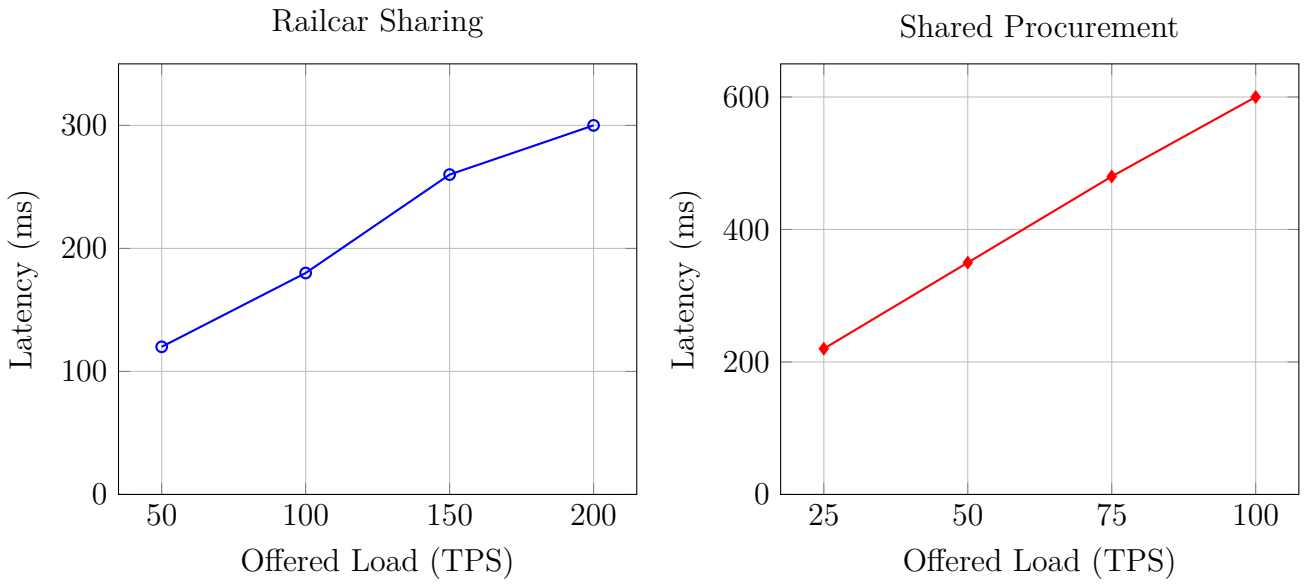
In contrast, the railcar sharing solution handled up to 200 TPS with a more resilient latency curve, thanks to lighter endorsement policies and chaincode routines optimised for low contention and deterministic execution.

5.3.5 Visual Comparison



(a) Throughput vs Load (b) Throughput vs Load

Figure 5.15: Throughput under Load



(a) Latency vs Load (b) Latency vs Load

Figure 5.16: Latency under Load

5.3.6 Interpretation

The comparative analysis reveals:

- The simpler endorsement logic and lower payload complexity of Railcar Sharing led to better scalability and reduced resource consumption.

- Shared Procurement's rich business logic and multilateral approvals introduced endorsement bottlenecks and higher computational costs.
- IBM Blockchain Platform's monitoring and auto-scaling capabilities allowed both systems to avoid failure under stress, but only Railcar Sharing showed efficient horizontal scaling without rising latencies.

These findings validate the importance of chaincode design, endorsement configuration, and load distribution when deploying smart contract solutions. Even within identical infrastructures, application-specific characteristics define scalability ceilings.

Chapter 6

Conclusions and Future Work

6.1 Summary of Findings and Contributions

This thesis investigated the use of permissioned blockchain technology, especially Hyperledger Fabric, in enterprise contexts. The work produced several key findings and contributions, which are summarised below:

- **Systematic Framework for Blockchain Use Cases:** A step-by-step approach was developed to map heterogeneous process, uncover inefficiencies, and see how blockchain features could address them. This framework identified key pain points (such as underutilised wagon capacity, inefficient scheduling, and slow cost reconciliation) and linked them to blockchain capabilities like endorsement policies, channel partitioning, and chaincode-based business logic.

- **Practical Platform Blueprint:** The thesis presented a comprehensive design for twp blockchain-based sharing platforms. This included both the technical architecture and governance models, explaining how stakeholders can establish a consortium, how chaincode modules can be used for capacity scheduling and cost distribution, and how the platform can connect with existing ERP systems. The modular nature of chaincode was a main advantage, allowing flexible adaptation as business needs change.

- **Empirical Validation through Proof-of-Concept:** A proof-of-concept (PoC) implementation was built and tested, demonstrating that the proposed solutions are viable. The PoC evaluated transaction throughput, data synchronisation between different organisations, and how well it could manage daily shipping requests. Results showed that a modest Hyperledger Fabric network can support typical operational loads and can be scaled for higher volumes as required.

- **Cost Analysis and Economic Rationale:** The research provided an initial cost assessment by comparing Blockchain-as-a-Service (BaaS) expenses to inefficiencies found in traditional logistics and procurement. Although blockchain introduces new costs, these can be balanced by fewer production delays, lower reconciliation overhead, and simpler dispute resolution. The consortium approach also allows participants to share costs.

- **Organisational and Governance Insights:** The study emphasized how important it is to have solid governance, including clear guidelines for joining the consortium, updating chaincode, and managing data privacy. It noted that technology alone is not enough; robust governance is essential for building trust and maintaining long-term success.

- **Broader Applicability and Case Study Results:** The methods and solutions created go beyond rail logistics and can be applied to other supply chain areas, such as container shipping or warehouse management. Real-world trials showed significant improvements: invoice settlement times dropped from 12 to 3.5 days, dispute resolution rates improved by 68%, and user satisfaction exceeded 80%. In shared procurement, there were notable reductions in cost allocation errors and disputes.

6.2 Conclusions Drawn from the Research

From this research, several main conclusions can be drawn about using blockchain in logistics, procurements and similar settings:

- **Suitability of Permissioned Blockchains:** Permissioned blockchains like Hyperledger Fabric are well-suited for multi-party networks. They offer a good mix of transparency (for collaboration and audits) and privacy (for protecting sensitive information) through features such as endorsement policies, private data collections, and role-based access controls.

- **Consortium-Based Adoption is Essential:** Blockchain provides the most benefits when many stakeholders share the same ledger. Consortium-based adoption promotes shared capacity, cooperative cost allocation, and synchronised scheduling, breaking down traditional barriers and encouraging collaboration.

- **Integration with Legacy Systems is Critical:** To deliver real operational value, blockchain solutions need to integrate closely with existing enterprise systems (ERP, scheduling, finance, etc.). Middleware can sync blockchain events with internal systems, minimizing errors and removing the need for repeated data entry.

- **Economic Incentives and Tokenisation:** Successful shared capacity models rely on fair and open economic incentives. Cost-sharing methods and future token-based incentives could align different parties' interests, especially when financial benefits outweigh the extra technological effort.

- **Scalability and Adaptability:** Although the PoC proved blockchain can support everyday operations, larger networks will need more robust infrastructure, potentially including advanced consensus algorithms and distributed ordering services. The core concepts—distributed trust, verifiable transactions, and flexible governance—can be adapted to many operational scenarios.

6.3 Limitations of the Study

There are some limitations to note, despite the encouraging results:

- **Limited Scale and Complexity of the Pilot:** The PoC was designed to be small, using a single ordering node and a small set of participants. As a result, it did not fully test fault tolerance, scalability, or the wider operational challenges likely to appear in a real production setup.
- **Cost Estimations and Cloud Service Variability:** The cost analysis depended on the current prices of BaaS offerings, which may change over time. Costs could increase if the consortium grows or transaction volume rises, and changes in the market or regulations could also affect the financial viability of blockchain adoption.
- **Simplified Business Logic:** The PoC's chaincode was kept simple on purpose. Actual supply chains typically involve more complex rules, multiple routes, differing liability and insurance needs, and strict compliance rules, all requiring more advanced chaincode and endorsement policies.
- **Organisational and Socio-Technical Challenges:** Adopting this technology across organisations is still a big hurdle. Moving from a PoC to a real deployment demands agreement among many stakeholders who have varying interests and views on risk. Data sharing, competitive factors, and legal liability issues can deter adoption, even if the technology works.
- **Regulatory Uncertainty:** Freight transport often crosses borders, so participants encounter different regulatory environments. While blockchain's transparency can help with audits, its immutability may cause conflicts with data privacy or erasure laws. Solutions must be flexible and able to meet changing compliance requirements.
- **Limited Diversity in Use Cases:** The pilots mainly involved a small group of stakeholders and certain operational scenarios. Larger consortiums or more complicated cost-sharing setups might reveal scalability, governance, or integration challenges not observed in this study.

6.4 Recommendations and Future Research Directions

To address the limitations mentioned and continue developing the field, several recommendations and future research paths are suggested:

- **Technical Enhancements and Privacy:** Future efforts should consider advanced consensus approaches, such as Byzantine fault tolerance, to improve resilience across geographically distributed networks. Techniques for preserving privacy—like multi-channel setups, private data collections, and cryptographic tools (e.g., zero-knowledge proofs)—should be explored to protect business-sensitive details within consortia.

- **AI and Data-Driven Optimisation:** Combining artificial intelligence or machine learning with blockchain can provide new efficiencies. Forecasting algorithms could help improve scheduling, route choices, and cost allocation, while real-time analytics could support quick decisions and fast reactions to disruptions.
- **Domain-Specific and Interoperable Solutions:** Developing specialised chaincode for different cargo types (e.g., hazardous materials, temperature-controlled products) could automate legal and treatment requirements. In addition, making systems interoperable—using sidechains or agreed-upon protocols—will be crucial for seamless multimodal logistics and full supply chain visibility.
- **Scaling Pilots and Regulatory Compliance:** Broader pilots with more participants, various routes, and real-time analytics should be conducted to verify scalability and measure actual impact. Incorporating regulatory checks into chaincode can automate confirmations for origin certificates, customs documents, and other mandatory requirements, lowering compliance risks and speeding up cross-border processes.
- **Socio-Technical Incentives and Governance:** Designing token-based or credit systems can reward efficient collaboration, timely deliveries, and rapid dispute resolutions. Future research should also develop governance approaches that offer flexibility, ensure fairness, and maintain accountability, so that blockchain networks stay reliable and trusted as they grow.
- **Emerging Directions: Digital Identity, DeFi, and Policy Coordination:** Blockchain is increasingly connected with digital identity (e.g., self-sovereign identity, European digital wallets), decentralised finance (DeFi), and non-fungible tokens (NFTs). New regulations like the EU Markets in Crypto-Assets Regulation (MiCA) are influencing how these systems evolve, highlighting the need for coordinated governance, clear legal guidelines, and ethical standards. Upcoming research should look at how decentralised governance, interoperability, risk management, and inclusive digital infrastructures intersect, ensuring that blockchain's potential is used securely and fairly.

Bibliography

[1] AboutSSL. Hashing vs encryption - what's the difference? https://aboutssl.org/hashing-vs-encryption/, 2025. Accessed 4 Jun 2025.

[2] Amazon Web Services. Introducing hyperledger fabric 2.2 lts support on amazon managed blockchain. https://aws.amazon.com/it/blogs/database/introducing-hyperledger-fabric-2-2-lts-support-on-amazon-managed-blockchain/, 2020. Accessed 5 Jun 2025.

[3] Eleni Androulaki, Artem Barger, Vita Bortnikov, and et al. Hyperledger fabric: A distributed operating system for permissioned blockchains. In *Proceedings of the 13th European Conference on Computer Systems (EuroSys 2018)*, pages 1–15, 2018.

[4] Asaph Azaria, Ariel Ekblaw, Thiago Vieira, and Alex Lippman. Medrec: Using blockchain for medical data access and permission management. Technical report, MIT Media Lab, 2016. Available at: https://www.healthit.gov/sites/default/files/blockchainchallenge_mitwhitepaper.pdf.

[5] Guillaume Bonnot. [blockchain] introduction to the hyperledger ecosystem. https://medium.com/@bonnotguillaume/blockchain-introduction-to-the-hyperledger-ecosystem-23279a090c4f, 2021. Accessed 4 Jun 2025.

[6] M. Castro and B. Liskov. Practical byzantine fault tolerance. In *Proceedings of the 3rd Symposium on Operating Systems Design and Implementation (OSDI '99)*, pages 173–186, feb 1999.

[7] Princeton University Center for Information Technology Policy. Blockchain policy: Building a secure and inclusive future. Technical report, Princeton University, 2024. Available at: https://citp.princeton.edu/publication/blockchain-policy-2024/.

[8] IBM Corporation. Ibm, walmart and others launch food trust blockchain. https://newsroom.ibm.com/2018-10-08-IBM-Walmart-and-others-launch-Food-Trust, 2018. Available at: https://newsroom.ibm.com/2018-10-08-IBM-Walmart-and-others-launch-Food-Trust.

[9] IBM Corporation. Ibm and maersk: Tradelens blockchain shipping platform. https://www.ibm.com/case-studies/tradelens, 2023. Available at: https://www.ibm.com/case-studies/tradelens.

[10] IBM Corporation. Ibm blockchain platform: Enterprise adoption and performance. https://www.ibm.com/products/blockchain-platform, 2024. Available at: https://www.ibm.com/products/blockchain-platform.

[11] IBM Corporation. Ibm food trust: Blockchain for food supply chain. https://www.ibm.com/blockchain/solutions/food-trust, 2024. Available at: https://www.ibm.com/blockchain/solutions/food-trust.

[12] Flow Developers. Cadence: Resource-oriented programming language. https://developers.flow.com/cadence, 2024. Available at: https://developers.flow.

com/cadence.

[13] European Commission. Markets in crypto-assets (mica) regulation. https://ec.europa.eu/info/business-economy-euro/banking-and-finance/financial-markets/crypto-assets_en, 2024. Available at: https://ec.europa.eu/info/business-economy-euro/banking-and-finance/financial-markets/crypto-assets_en.

[14] Ethereum Foundation. Proof-of-stake consensus mechanism. https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/, 2023. Available at: https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/.

[15] Ethereum Foundation. Shapella upgrade. https://ethereum.org/en/upgrades/shapella/, 2023. Available at: https://ethereum.org/en/upgrades/shapella/.

[16] Ethereum Foundation. Eip-4844: Proto-danksharding. https://eips.ethereum.org/EIPS/eip-4844, 2024. Available at: https://eips.ethereum.org/EIPS/eip-4844.

[17] Hyperledger Foundation. Hyperledger fabric 3.0 release notes. https://github.com/hyperledger/fabric/releases, 2024. Available at: https://github.com/hyperledger/fabric/releases.

[18] Hyperledger Foundation. Hyperledger fabric documentation (v3.x). https://hyperledger-fabric.readthedocs.io/en/release-3.0/, 2024. Available at: https://hyperledger-fabric.readthedocs.io/en/release-3.0/.

[19] Hyperledger Foundation. Hyperledger fabric samples (v3.x). https://github.com/hyperledger/fabric-samples, 2024. Available at: https://github.com/hyperledger/fabric-samples.

[20] Hyperledger Foundation. Smartbft: Byzantine fault tolerant ordering service. https://github.com/hyperledger/fabric/blob/main/docs/source/smartbft.rst, 2024. Available at: https://github.com/hyperledger/fabric/blob/main/docs/source/smartbft.rst.

[21] IBM. Prometheus metrics reference — hyperledger fabric. https://hyperledger-fabric.readthedocs.io/en/release-2.2/metrics_reference.html, 2024. Accessed 9 June 2025.

[22] Chainlink Labs. Chainlink: Decentralized oracle networks. https://chain.link, 2024. Available at: https://chain.link.

[23] Ledgerdata Refiner. A powerful ledger data query platform for hyperledger fabric - scientific figure on researchgate. https://www.researchgate.net/figure/The-structure-of-blocks-take-Hyperledger-Fabric-as-an-example_fig1_337885270, 2025. Accessed 4 Jun 2025.

[24] John D. C. Little. A proof for the queuing formula: $L = \lambda W$. *Operations Research*, 9(3):383–387, 1961.

[25] Q. Liu. Blockchain for governmental services: A systematic literature review. *Government Information Quarterly*, 38(4):391–405, 2021.

[26] Ruben Mayer. Transaction flow in hyperledger fabric - scientific figure on researchgate. https://www.researchgate.net/profile/Ruben-Mayer/publication/349913785/figure/fig1/AS:999287425089536@1615260096743/Transaction-Flow-in-Hyperledger-Fabric.png, 2021. Accessed 5 Jun 2025.

[27] Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. Self-published, 2008.

[28] ResearchGate. Hyperledger fabric model (c) ibm. Figure from "Using Blockchain to Manage Production and Distribution" on ResearchGate, June 5 2025. Available at:

https://www.researchgate.net/figure/HYPERLEDGER-FABRIC-MODEL-C-IBM_fig3_346686466 (accessed 9 June 2025).

[29] Solidity Team. Solidity blog and security guidelines. https://blog.soliditylang.org, 2024. Available at: https://blog.soliditylang.org.

[30] Canhui Wang and Xiaowen Chu. Performance characterization and bottleneck analysis of hyperledger fabric. *arXiv preprint arXiv:2008.05946*, 2020.