

Article

A Blockchain-Based Tokenize–Track–Trace (3T) Architecture for Oracle-Anchored Supply Chain Traceability Using Dynamic NFTs and Smart Contracts

Yassine Idel Mahjoub, Tarik Chargui, Abdelghani Bekrar and Damien Trentesaux * 

LAMIH, CNRS, UMR 8201, Univ. Polytechnique Hauts-de-France, F-59313 Valenciennes, France; yassine.idelmahjoub@uphf.fr (Y.I.M.); tarik.chargui@uphf.fr (T.C.); abdelghani.bekrar@uphf.fr (A.B.)

* Correspondence: damien.trentesaux@uphf.fr

Abstract

This paper introduces an approach based on blockchain and dynamic Non-Fungible Tokens (dNFTs) for transparent supply chain management (SCM) through end-to-end traceability and automation. The approach addresses three recurring limitations of blockchain-enabled SCM: physical-to-digital anchoring, explicit event semantics and authorization, and reproducible performance evaluation. Each physical product is represented as a dynamic on-chain asset whose tokenization is gated by a threshold oracle-attestation rule, and whose lifecycle is updated through role-authorized, auditable events, supporting accountability-oriented practices such as enhanced transparency, improved provenance, stronger recall capability, and responsible sourcing. We operationalize the proposed approach through the 3T (Tokenize–Track–Trace) architecture and instantiate it on a permissioned blockchain to demonstrate oracle-anchored product tokenization, end-to-end traceability, and workflow automation via smart contracts. Performance is evaluated using standard blockchain metrics (throughput, latency, load-scaling and saturation behavior under a fixed topology, resource utilization, success rate, and traceability-query latency) under controlled load-generation settings. The paper also discusses security assumptions, implementation limitations, and potential future research directions for blockchain-enabled SCM.

Keywords: blockchain; smart contracts; supply chain; transparency; tokenization; traceability; security; automation; Hyperledger

1. Introduction

Subject to the trends of economic globalization, market expansion, and escalating competition, companies from various industries are facing increasingly complex management and control challenges [1]. Recent disruptions, such as the COVID-19 outbreak, have highlighted the necessity for sustainable and transparent supply chain systems. These disruptions exposed vulnerabilities including lack of end-to-end visibility, insufficient information traceability and transparency, cash flow issues, inadequate collaboration among stakeholders, and security concerns [2]. Such issues underscore the critical need for industries to adopt digital transformation, embracing sustainable practices through advanced information systems and digital technologies [3].

The world landscape has witnessed an exponential growth of new digital technologies and innovations such as artificial intelligence, big data, internet of things, blockchain, and distributed ledger technologies. Blockchain is considered as an emerging



Academic Editors: Jesús Rodríguez-Molina and Bimal Ghimire

Received: 12 June 2026

Revised: 19 July 2026

Accepted: 29 July 2026

Published: 4 August 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

and potentially disruptive technology that could transform societies, social organizations, and economies by providing cost-efficient and transparent solutions to different industries [2,4–6]. Satoshi Nakamoto first implemented it in 2009 during the global financial crisis [7], which laid out the mathematical foundation for the first cryptographic currency, Bitcoin. Blockchain was mainly used in the financial sector and digital assets such as digital currencies (e.g., Bitcoin, Ethereum). However, its benefits were soon recognized by other domain applications, such as supply chain [8], healthcare [9], transportation [10], energy [11], and entertainment industry (e.g., metaverse and non-fungible tokens (NFTs)) [12], as industry and academia acknowledged that it could improve efficiency. Peer-to-peer lending and decentralized finance further illustrate blockchain's broader role in digitally mediated financial services [13,14]. Blockchain is also being explored for sustainable and resilient Industry 5.0 applications [15,16]. These examples provide broader context while the present study remains focused on enterprise SCM.

Despite rapid experimentation beyond cryptocurrencies, blockchain-enabled SCM systems still face persistent limitations related to trustworthy physical-to-digital linkage, well-defined event semantics, and reproducible performance evaluation [17–19]. In this context, the present paper focuses on traceability and automation mechanisms that can be instantiated on enterprise-grade ledgers.

1.1. Research Gap

Despite extensive adoption of blockchain for traceability, practical SCM deployments still face three unresolved issues: (i) *physical-to-digital* anchoring (preventing the digitization of non-existing or counterfeit items), (ii) *event semantics and authorization* (defining who may update which part of the lifecycle, under which rules), and (iii) *reproducible evaluation* (reporting performance under clearly stated experimental parameters). This work addresses these gaps through a unified architecture and an implementation-level validation on a permissioned blockchain.

As synthesized in Section 3.5, prior approaches commonly emphasize one or two of these requirements rather than specifying them jointly within a single auditable lifecycle model. The resulting gap is therefore not the absence of blockchain, NFTs, or oracles individually, but the absence of an integrated specification that couples threshold-gated product creation, role authorization, state-transition preconditions, event-specific evidence validation, dynamic product-state evolution, and a reproducible traceability evaluation protocol.

1.2. Contributions

This paper makes the following contributions:

- We propose a unified architecture (3T: Tokenize–Track–Trace) for supply chain traceability and automation that combines dynamic non-fungible representations (*dNFTs*), oracle-validated anchoring of physical events, and event-driven smart-contract workflows.
- We formalize the product state, oracle-attestation rule, authorization conditions, and event-triggered transition model as a platform-independent specification, and demonstrate one concrete realization on Hyperledger Fabric.
- We implement the proposed design on Hyperledger Fabric, including smart-contract interfaces for tokenization, custody transfer, and delivery confirmation, with role-based authorization.
- We evaluate the implementation under controlled workload, reporting throughput, latency, load-scaling and saturation behavior under a fixed topology, and resource efficiency with explicitly stated experimental parameters.

1.3. Novelty Boundary

The scientific contribution does not reside in a new consensus mechanism, cryptographic primitive, oracle protocol, or token standard. Rather, it resides in the integrated 3T lifecycle design that combines: (i) an explicit oracle-threshold condition for physical-product tokenization, instantiated with $\tau = 2$ in the case study; (ii) formal identity- and role-based authorization through $\text{Auth}(u, e, x)$ together with the business-state precondition $\text{Pre}(e, x)$ for typed lifecycle updates; (iii) a dynamic non-fungible product state linked to an auditable event history; and (iv) an implementation and controlled performance benchmark on Hyperledger Fabric. The value $\tau = 2$ is the case-study tokenization policy.

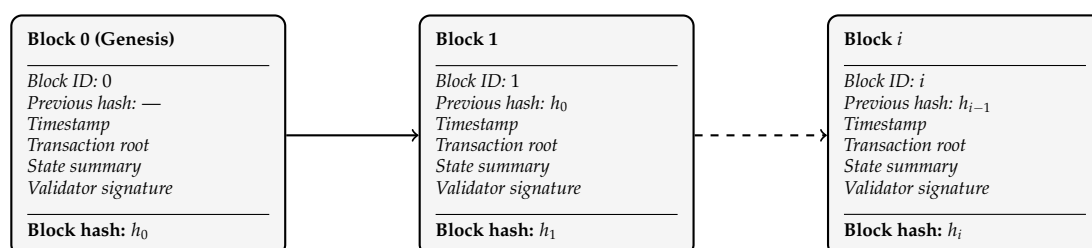
The product-state, event, oracle-threshold, authorization, and transition semantics are platform-independent abstractions; their implementation is necessarily platform-specific. We instantiate these abstractions on Hyperledger Fabric to demonstrate feasibility and to report reproducible performance results, and Section 4.4.1 identifies the adapters required for an EVM-compatible deployment.

Section 2 presents the background and basic concepts. Section 3 reviews related work and adds an analytical study-level comparison. Section 4 details the proposed 3T architecture, its formal acceptance semantics, and its computational characteristics. Section 5 instantiates the model on Hyperledger Fabric and presents the case study and experimental results. Section 6 discusses the scientific and industrial implications and the evaluation boundary. Section 7 concludes the paper, consolidates the limitations, and outlines future research directions.

2. Background and Basic Concepts

2.1. Blockchain Technology

The standard ledger, authentication, smart-contract, and governance concepts reviewed below are included to keep the interdisciplinary presentation self-contained and to define mechanisms referenced later in the 3T model. They are not presented as scientific contributions. The contribution boundary is stated in the Introduction and examined comparatively in Section 3.5. Blockchain is a Distributed Ledger Technology (DLT) that stores transactions in blocks chained together with cryptographic mechanisms in a peer-to-peer network [7] (Figure 1). Blockchain was designed to operate without a central authority (i.e., with no bank or regulator controlling the network), while still ensuring that transactions are authenticated and tamper-evident.



Each block commits a cryptographic summary of transactions and state. Block hashes are computed as $h_i = H(B_i \parallel h_{i-1})$, ensuring immutability and verifiable traceability across the ledger.

Figure 1. Basic blockchain structure. Each block encapsulates a cryptographic summary of transactions and system state, and includes the hash of the previous block. This chained construction ensures immutability, auditability, and tamper evidence, and serves as the foundation for traceability mechanisms used throughout the paper.

2.1.1. Hash-Chained Data Structure

Let B_i denote the i -th block and let h_i be its block hash. The cryptographic linkage between consecutive blocks is typically modeled as:

$$h_i = H(B_i \parallel h_{i-1}), \quad (1)$$

where $H(\cdot)$ is a cryptographic hash function and $\parallel$ denotes concatenation. This hash-pointer structure implies that any modification of a past block changes its hash and breaks the linkage to all subsequent blocks, providing strong tamper evidence.

2.1.2. Authentication and Digital Signatures

Transactions are authenticated using asymmetric cryptography through a key pair (sk, pk) : a private signing key sk and a public verification key pk [7]. The public key can be viewed as an address to which value or messages are directed, while the private key is used to authorize actions. Formally, for a transaction payload m , the sender computes a digital signature

$$\sigma = \text{Sign}(sk, m), \quad (2)$$

and any node can verify authenticity via:

$$\text{Verify}(pk, m, \sigma) = 1. \quad (3)$$

This mechanism provides non-repudiation and prevents unauthorized transaction issuance.

Common blockchain properties and their deployment-dependent interpretations are summarized in Table 1.

Table 1. Common blockchain properties under deployment-specific trust and governance assumptions.

Tamper-evident	Committed records resist undetected alteration under the ledger's stated trust and validation assumptions.
Shared visibility	Participants maintain full or partial ledger views according to the network's access and privacy model.
Distributed governance	Control is distributed to a degree determined by the public, private, or consortium architecture.
Authenticated or pseudonymous identities	Identity treatment depends on the network model; the evaluated Fabric network uses authenticated X.509 identities.
Data integrity, not data truth	The ledger preserves submitted data and attribution but does not guarantee that a physical-world observation is accurate.
Fault tolerance	Tolerance of node, network, or ordering failures depends on replication, consensus, and deployment configuration.
Programmable intermediation	Smart contracts can automate selected intermediary and approval functions without eliminating governance responsibilities.
Settlement characteristics	Latency and finality depend on the platform, ordering or consensus mechanism, and workload configuration.

These properties concern ledger operation and governance; they should not be interpreted as guarantees of physical-data accuracy, anonymity, decentralization, or faster settlement in every deployment.

Once a transaction is agreed upon between peers, it must be validated and ordered before being appended to the ledger. The decision to commit a transaction is made through a consensus protocol [20], i.e., a set of nodes must agree on transaction validity and ordering. Consensus reduces the risk of divergent histories (e.g., forks) and ensures that all honest nodes converge to the same ledger state. Nodes maintain a full or partial copy of

the blockchain; increasing the number of nodes generally improves decentralization and resilience to single-point failures [21]. Different consensus mechanisms have been proposed, including Proof-of-Work (PoW), Proof-of-Stake (PoS), and Proof-of-Authority (PoA) [20]. Once a transaction is committed, altering historical data requires recomputing and replacing the cryptographic links across multiple blocks, which is computationally and operationally infeasible under standard threat models and sufficient honest participation assumptions.

Blockchain technology is not only limited to simple value transactions between peers; it also provides a programmable execution layer enabling conditional transactions upon fulfillment of specified rules [22]. This capability is realized through smart contracts [23], i.e., self-executing programs stored and executed according to deterministic logic (often expressed as *if/when-then* statements). Smart contracts permit trusted transactions and agreements to be carried out among peers without requiring a central authority or a traditional trusted intermediary. First implemented at scale by Ethereum, smart contracts enabled applications such as decentralized finance (DeFi), decentralized applications (DApps), decentralized autonomous organizations (DAOs), and tokenized assets including non-fungible tokens (NFTs) [12]. In enterprise contexts, smart contracts can streamline business processes by reducing intermediary overhead and automating verifiable workflows [24,25].

Depending on the use case and requirements, blockchains have been categorized into three types of architectures: public, private, and consortium. Public blockchains are permissionless and non-restrictive, where all participants connected to the internet can read, write, and verify the network transactions and contribute to the consensus process. Examples of such types are Bitcoin and Ethereum [24]. Unlike the public, a private blockchain is a permissioned, restrictive, and more centralized blockchain that operates in a closed network. It is mainly owned, operated, and controlled by a single entity that makes its own rules for access and use. For these reasons, it may be suited for enterprises and businesses as they require significant control over the participants who can access data [26]. Hyperledger Fabric is an example of such a blockchain [21,27]. Finally, a consortium blockchain is best suited for organizations with a need for both types of blockchains, i.e., public and private. Copies of the blockchain are only distributed among the consortium. IBM Food Trust and Corda are examples of such blockchain.

2.2. Oracles and Physical-to-Digital Anchoring

A core limitation of purely digital ledgers in SCM is that they cannot directly verify real-world conditions (e.g., whether a physical product exists, whether a handover occurred, or whether a temperature threshold was violated). An *oracle* is an off-chain mechanism (human, IoT device, information system, or a trusted service) that observes real-world facts and produces a verifiable statement that can be consumed by on-chain logic.

Definition 1 (Oracle attestation). Let $p \in \mathcal{P}$ be a physical product and let $\mathcal{V}$ denote a set of validators (or oracle agents). Each validator $v \in \mathcal{V}$ issues an attestation

$$\sigma_v(p) \in \{0, 1\}, \quad (4)$$

where $\sigma_v(p) = 1$ indicates that v confirms the required real-world condition for p (e.g., product existence at tokenization time, or delivery confirmation).

Definition 2 (Threshold acceptance rule). Given a threshold $\tau \in \mathbb{N}$, the oracle-validated condition holds if:

$$\text{Valid}(p) = \mathbb{I}\left(\sum_{v \in \mathcal{V}} \sigma_v(p) \geq \tau\right), \quad (5)$$

where $\mathbb{I}(\cdot)$ is the indicator function. Only if $\text{Valid}(p) = 1$ is the corresponding on-chain transition authorized (e.g., creation of the product traceability object).

Threshold Safety and Liveness

Let $n = |\mathcal{V}|$. Assuming that honest validators correctly assess the targeted condition, do not approve a false statement, and retain uncompromised credentials, fewer than τ malicious or compromised validators cannot independently satisfy the threshold; false acceptance therefore requires at least τ such validators. Liveness is a separate property: a valid statement can be accepted only when at least τ validators are available and approve it, so at most $n - \tau$ unavailable or withholding validators can be tolerated. In the case-study policy $n = \tau = 2$, one unavailable validator blocks tokenization, while collusion or compromise of both validators can satisfy the rule for a false statement. The ledger verifies that the configured approvals were recorded; it does not establish the physical truth of those approvals.

2.3. Performance Metrics Used in This Paper

To enable reproducible evaluation and comparison across configurations, we use standard blockchain performance indicators. For an experiment committing N transactions during a measurement window $\Delta t = t_{\text{end}} - t_{\text{start}}$:

$$\text{TPS} = \frac{N}{\Delta t}. \quad (6)$$

For each transaction i , end-to-end latency is defined as:

$$L_i = t_{\text{commit}}^{(i)} - t_{\text{submit}}^{(i)} \quad (7)$$

The next section explores the potential impact of blockchain technology on SCM. This part of the paper highlights the significant advantages blockchain offers, such as improved transparency, enhanced security, and better product tracking in supply chains. It also addresses the challenges, including scalability issues, technical complexities, and regulatory uncertainties. The review evaluates how prior SCM systems address physical anchoring, product representation, authorization, privacy, and empirical validation, thereby establishing the comparison basis for the 3T architecture.

3. Literature Review: Blockchain Opportunities and Challenges for SCM

3.1. Supply Chain Management Issues

Supply chain management (SCM) is the systematic coordination and management of activities involved in sourcing, procurement, conversion, logistics, and collaboration with suppliers, intermediaries, third-party service providers, and customers [28]. Its objective is to coordinate material, information, and financial flows from the point of origin to the point of consumption while improving operational performance, resilience, and sustainability.

Modern supply chains are expansive, multi-actor, and frequently fragmented networks involving manufacturers, suppliers, logistics providers, warehouses, retailers, and final consumers. These networks are often supported by paper-based processes and disconnected information systems, which create data silos and make it difficult to track product status, verify provenance, and reconstruct the chain of custody. This lack of traceability and transparency can lead to delays, errors, disputes, counterfeiting risks, inefficient recalls, and higher operational costs [29]. Figure 2 summarizes these traditional supply-chain issues.

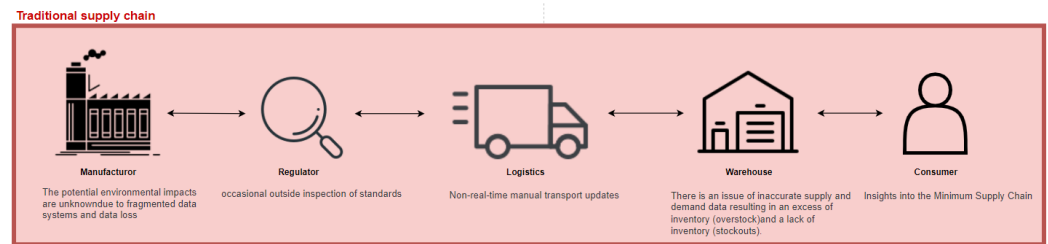


Figure 2. Traditional supply chain issues.

From a traceability perspective, three requirements are particularly important. First, physical products must be reliably linked to their digital representations. Second, lifecycle events such as production, pickup, warehouse check-in, transfer, delivery, and recall must be represented with clear semantics. Third, each event must be attributable to an authorized actor. Blockchain and distributed ledger technologies can contribute to these requirements by providing a shared, tamper-evident, and auditable transaction history. However, blockchain alone does not solve all SCM problems: it cannot directly verify physical reality, it may expose sensitive business relationships if privacy is not properly managed, and its performance depends strongly on platform design and deployment configuration.

3.2. Blockchain Opportunities and Limitations for SCM

Blockchain technology offers several opportunities for SCM, especially in contexts where multiple organizations need to share trusted information without relying on a single central database [4]. In supply chain traceability, its main value lies in tamper-evident record keeping, shared visibility, authenticated transactions, and programmable automation through smart contracts.

3.2.1. Traceability, Transparency, and Privacy

Blockchain can improve supply chain traceability by recording product-related events in an append-only ledger shared across participating actors. Once a lifecycle event is committed, it becomes auditable and difficult to alter without detection. This property is useful for provenance verification, custody tracking, recall management, and compliance audits [25,30]. Compared with fragmented databases, a distributed ledger can reduce inconsistencies between actors by providing a common history of committed events.

However, transparency also creates a privacy challenge. In SCM, transaction metadata may reveal commercially sensitive information such as supplier relationships, transaction frequency, routes, volumes, or customer identities. Therefore, public transparency is not always desirable for enterprise supply chains. Permissioned frameworks, such as Hyperledger Fabric, address this issue through identity management, access-control policies, channels, and private data mechanisms [21]. This trade-off between auditability and confidentiality is central when selecting a blockchain platform for SCM.

3.2.2. Automation Through Smart Contracts

Smart contracts enable business rules to be encoded as executable logic. In SCM, they can automate operations such as product registration, custody transfer, delivery confirmation, compliance checks, and event-triggered notifications [31]. Automation reduces manual coordination and can improve process consistency, especially when multiple parties must agree on the state of a product.

Nevertheless, smart contracts are only as reliable as the data they receive. In supply chains, many relevant events occur in the physical world: a product is manufactured, inspected, transported, stored, delivered, or recalled. A blockchain cannot observe these

events directly. As a result, smart-contract automation must be combined with oracles, sensors, enterprise systems, or human validators to anchor physical events to digital records.

3.2.3. Tokenization and Dynamic NFTs

Tokenization provides a digital representation of a physical or logical asset on a blockchain. Non-fungible tokens (NFTs) are especially relevant for traceability because each token can represent a unique product, batch, vehicle, certificate, or asset record [32]. In SCM, such uniqueness is useful for provenance, anti-counterfeiting, product history, and recall management. However, many NFT-based models are static: they mainly certify existence or ownership at a given point in time. Supply chain products, by contrast, change state throughout their lifecycle. Their location, custodian, condition, quality status, and delivery state may evolve several times before reaching the final consumer. This motivates the use of dynamic NFTs (dNFTs), where the asset keeps a persistent unique identity while selected attributes are updated through authorized lifecycle events. In a permissioned implementation such as Hyperledger Fabric, a dNFT can be implemented as a chaincode-managed dynamic asset record rather than an ERC-style token. This distinction is important because the scientific contribution is not the use of an Ethereum token standard, but the modeling of a non-fungible product state whose lifecycle is updated and audited through blockchain events.

3.2.4. Traceability, Accountability, and Compliance Support

Blockchain can support SCM by improving provenance, accountability, and visibility over sourcing, handling, transportation, storage, and delivery processes [25,29,30]. In traditional supply chains, product information is often fragmented across several actors and information systems, which makes it difficult to verify where a product came from, who handled it, when a transfer occurred, and whether the product followed the expected route. A shared ledger can reduce these inconsistencies by maintaining a tamper-evident record of committed lifecycle events.

From a traceability perspective, blockchain is particularly useful when product events must be reconstructed after the fact. For example, a product recall, delivery dispute, quality-control investigation, or compliance audit may require the reconstruction of the complete product history. In such cases, the ledger can provide an auditable sequence of events, including tokenization, custody transfer, warehouse check-in, warehouse check-out, location update, delivery confirmation, and recall return. This capability supports both forward traceability, from supplier to consumer, and backward traceability, from consumer or warehouse back to the supplier.

Smart contracts can further strengthen accountability by enforcing predefined rules before accepting lifecycle updates. For instance, only an authorized transporter should submit transportation events, only a warehouse should submit warehouse check-in or check-out events, and only a consumer or authorized receiver should confirm delivery. When combined with oracle validation and evidence references, these rules help ensure that product histories are not only recorded but also attributable to authenticated actors.

However, blockchain-based traceability should not be interpreted as automatic proof of physical truth. A blockchain can preserve and verify the integrity of committed records, but it cannot independently confirm that a physical product exists, that a location update is accurate, or that off-chain evidence has not been manipulated unless appropriate oracle, sensor, or verification mechanisms are added. Therefore, reliable traceability requires both ledger integrity and trustworthy physical-to-digital anchoring.

3.3. Industry Innovations: Blockchain Projects and Platforms

In this section, we position representative platforms along dimensions that directly impact SCM deployments: governance, programmability, privacy mechanisms, and suitability for enterprise workflows. The objective is not to benchmark these platforms, but to clarify why different SCM contexts may require different blockchain or DLT infrastructures.

This positioning clarifies why permissioned frameworks, particularly Hyperledger Fabric, are often selected for enterprise traceability use cases requiring identity management, governance, data access control, and private information sharing [21]. Public blockchains may be suitable for open provenance and consumer-facing verification, whereas permissioned frameworks are more appropriate when participants require controlled membership, confidentiality, and role-based business workflows (Table 2).

Table 2. Qualitative comparison of selected blockchain/DLT platforms for supply chain management (SCM) use cases.

Platform	Governance	Smart Contracts	Privacy Support	SCM Relevance
Ethereum/EVM	Public permissionless	Yes, via EVM/Solidity	Limited by default on public L1; privacy requires additional design such as off-chain storage, encryption, ZK mechanisms, or private/permissioned deployments	Open tokenization, public provenance, consumer-facing verification, and anchoring of traceability proofs. Less suitable for confidential enterprise SCM data without additional privacy layers.
Polygon PoS/EVM	Public Ethereum-anchored EVM chain	Yes, via EVM/Solidity	Limited by default; application-level privacy must be added separately	Lower-cost anchoring of traceability events and consumer-facing provenance proofs, while preserving EVM tooling compatibility.
Avalanche L1s/Subnets	Public ecosystem with customizable Avalanche L1/subnet governance	Yes, via C-Chain, Subnet-EVM, or custom virtual machines	Varies by L1/subnet design; membership, validation rules, and compliance constraints can be customized	Dedicated SCM consortium networks with custom validator sets, compliance rules, and application-specific execution environments.
Solana	Public permissionless	Yes, via on-chain programs	Limited by default; public ledger visibility unless application-level privacy is added	High-throughput event logging, asset minting, and consumer-facing traceability applications, but limited native confidentiality for enterprise SCM.
Hyperledger Fabric	Permissioned enterprise/consortium	Yes, via chaincode	Strong support through channels, private data collections, access-control lists, MSP identities, and endorsement policies	Enterprise traceability, controlled multi-organization workflows, private data sharing, custody-transfer automation, and auditable product history.
Hyperledger Besu	Public or permissioned Ethereum client; enterprise deployments can use permissioned networks	Yes, via EVM/Solidity	Node/account permissioning is supported; privacy mechanisms are deployment- and version-dependent	Consortium SCM where Ethereum tooling, Solidity contracts, and permissioned network governance are required.
GoQuorum	Permissioned/consortium Ethereum	Yes, via EVM/Solidity	Private transactions and private contracts through privacy managers and transaction visibility sets	Confidential B2B workflows, controlled data disclosure, and enterprise processes requiring Ethereum compatibility.
Corda	Permissioned/consortium DLT	Yes, through contracts and flows	Point-to-point, need-to-know data sharing rather than global broadcast	Inter-firm business processes, bilateral or multi-party coordination, trade finance, logistics contracts, and legal-state synchronization.
Hedera Hashgraph	Public network with council governance	Yes, EVM-compatible smart-contract service	Limited by default on public network; confidentiality must be handled at application or data-management layer	Tamper-evident timestamping, event anchoring, audit trails, and high-integrity provenance records.
IOTA / Tangle	Public DLT/DAG ecosystem	Yes, through IOTA Smart Contracts / IOTA EVM	Varies by smart-contract chain and application design; public anchoring is not confidential by default	IoT telemetry anchoring, logistics data integrity, device-originated events, and lightweight traceability scenarios.
VeChainThor	Public blockchain with authority-based governance	Yes	Limited by default; privacy and selective disclosure require application-level design	Product provenance, anti-counterfeiting, consumer-facing product verification, and industry-oriented traceability use cases.
MultiChain	Permissioned private/consortium blockchain	Limited; native assets, streams, and permission primitives rather than full general-purpose smart contracts	Network-level access control and permissions	Private asset tracking, controlled data sharing, append-only event streams, and lightweight consortium traceability.

Table 2. *Cont.*

Platform	Governance	Smart Contracts	Privacy Support	SCM Relevance
IBM Food Trust	Managed Fabric-based industry network	Managed application logic on Hyperledger Fabric	Permissioned access controls and role-based data sharing	Food traceability, provenance, compliance support, and industry-specific supply chain data sharing. Included as an operational SCM network rather than a general-purpose blockchain platform.

Note. This table is a qualitative positioning of representative platforms and networks for SCM use cases. It is not a throughput, latency, cost, or security benchmark. Platform characteristics are synthesized from official or technical documentation for public smart-contract platforms, permissioned enterprise ledgers, and SCM-oriented industry networks [33–45].

Platform selection in SCM is principally a governance and data-disclosure decision rather than a generic ranking exercise. Public smart-contract networks can support open provenance and consumer-facing verification, whereas permissioned networks are better aligned with authenticated organizations, configurable endorsement, and restricted commercial data. Hyperledger Fabric is used in this study because its identity, endorsement, channel, and private-data mechanisms correspond directly to the 3T authorization and enterprise-governance requirements [21,37]. The study-level comparison in Section 3.5 complements the preceding platform positioning, while EVM portability is treated separately in Section 4.4.1.

3.4. Academic Insights on Blockchain and NFTs in SCM

The academic literature on blockchain in SCM has grown significantly, with several review studies examining adoption drivers, benefits, limitations, and integration with technologies such as IoT [46–52]. These studies generally agree that blockchain can improve traceability, trust, auditability, and inter-organizational coordination. They also identify recurring barriers, including scalability, data privacy, integration with legacy systems, governance, regulatory uncertainty, and the difficulty of ensuring that on-chain data accurately reflects physical-world events.

Beyond review studies, several application-oriented works have explored blockchain-based SCM solutions. Ref. [53] investigate blockchain for decentralized logistics services procurement, focusing on carrier-shipper mapping, information transparency, and power rebalancing. Their work shows the potential of blockchain for autonomous and trustworthy logistics allocation, but also highlights the importance of oracles and permissioned infrastructure for privacy-sensitive transport applications. NFT-based approaches have also emerged in traceability research. In healthcare logistics, Refs. [54,55] propose NFT-based solutions for healthcare products and drug traceability. These works demonstrate the relevance of NFTs for digital certification, product identification, and provenance, while also noting challenges related to NFT standards, scalability, regulation, and practical deployment. Similarly, Ref. [56] propose NFT-based traceability and ownership management for medical devices, but identify limitations related to scalability, real-time traceability, and integration with existing systems. Other works explore NFTs in logistics and transactional workflows. Ref. [57] propose a blockchain, smart-contract, and NFT-based cash-on-delivery model connecting sellers, shippers, and buyers. Their model uses ERC-721 NFTs to store package-related information and demonstrates the suitability of NFTs for package-level digital representation. However, such approaches are often based on static NFT records or public-chain assumptions, and they do not fully address enterprise requirements such as permissioned identity, authorized lifecycle updates, private data handling, oracle-anchored product validation, and reproducible performance evaluation.

3.5. Summary of Existing Studies, Research Gap, and Contributions

The existing studies can be organized into three complementary themes. First, blockchain-based SCM platforms emphasize shared provenance, tamper-evident event histories, and smart-contract automation, but the physical origin and reliability of submitted data generally remain dependent on external actors, devices, or information systems. Among Fabric-specific implementations, Marchese and Tomarchio [58] demonstrate organizational roles, product and batch histories, runtime quality rules, and request/accept transfer workflows, but do not formalize a multi-validator tokenization threshold or report a matched saturation and trace query evaluation. Second, lifecycle-oriented digital product representations—including the NFT-based implementations reviewed above—seek to maintain a persistent identity and lifecycle record for each item; however, they do not consistently formalize role-based authorization and business-state preconditions for each update. Third, oracle- and IoT-assisted approaches improve physical-to-digital data capture, but the reviewed works do not jointly specify a threshold-gated creation rule, typed authorized state transitions, and a reproducible traceability benchmark within one common lifecycle model.

Table 3 shows that the individual mechanisms used by 3T have clear precedents. The Hyperledger-based comparison includes the Ethereum/Hyperledger Sawtooth implementation of Caro et al., the Hyperledger Fabric prototype of Marchese and Tomarchio, and the Fabric-based IBM Food Trust system. The identified gap is therefore not that prior platforms are incapable of implementing the relevant mechanisms, but that the cited studies and systems do not jointly specify and evaluate the complete chain from physical-product admission to authorized state evolution and auditable trace reconstruction. Three combined limitations remain: (i) product creation is not consistently conditioned on an explicit multi-validator rule; (ii) actor identity, role, current state, and event-specific evidence are not jointly expressed as acceptance conditions for every lifecycle transition; and (iii) the resulting trace model is not consistently linked to a reproducible write-load and history-query evaluation.

Table 3. Analytical comparison of representative blockchain traceability studies and systems with the proposed 3T architecture.

Study or System	Physical Anchoring or Oracle Treatment	Product or Process Representation	Authorization and State Semantics	Privacy and Enterprise Orientation	Validation Scope Reported
Caro et al. [29] (Ethereum and Hyperledger Sawtooth)	IoT-originated traceability data are recorded; a threshold validator policy is not formalized.	Product and batch traceability records.	Application workflow transactions; no formal Auth/Pre predicate pair is reported.	Enterprise confidentiality is not the principal comparison dimension.	Practical implementation and platform-oriented performance measurements.
Marchese and Tomarchio [58]	Product and batch data are submitted by authenticated organizations; no multi-validator oracle threshold is formalized.	Fabric-managed product and batch resources with lifecycle histories; not a dynamic NFT representation.	Organization roles, resource states, runtime quality rules, and request/accept transfer operations; no unified Auth/Pre/Valid acceptance model is reported.	Permissioned Fabric prototype with organization-specific services; field-level selective disclosure is not the reported focus.	Prototype and multi-step use-case validation; no matched saturation or trace query benchmark.
IBM Food Trust [45] (Hyperledger Fabric)	Traceability data are contributed by permissioned participants and connected enterprise systems; physical truth remains dependent on the contributing source.	Managed food product identity, provenance, and lifecycle records rather than the formal dynamic-state model used by 3T.	Role-based workflows and permissioned data sharing; the cited operational documentation does not present the unified Auth/Pre/Valid predicate set used by 3T.	Managed Fabric-based industry network with permissioned access and role-oriented data sharing.	Operational documentation; no matched public benchmark under the 3T workload and testbed.
Henry et al. [53]	Oracle dependence is identified for logistics-service information.	Logistics procurement and carrier-shipper records rather than a dynamic product NFT.	Procurement and allocation workflow; no typed product-state transition model is reported.	Permissioned infrastructure is motivated for privacy-sensitive logistics.	Conceptual and analytical evaluation of decentralized logistics procurement.

Table 3. Cont.

Study or System	Physical Anchoring or Oracle Treatment	Product or Process Representation	Authorization and State Semantics	Privacy and Enterprise Orientation	Validation Scope Reported
Musamih et al., Turki et al., and Gebreab et al. [54–56]	Product, device, or IoT data are linked to blockchain records; no common threshold-attestation rule is specified across these studies.	NFT-based healthcare, pharmaceutical, or medical-device identity and provenance.	Smart-contract lifecycle workflows; formal role- and state-dependent acceptance predicates are not consistently specified.	Public-chain and off-chain designs are considered; field-level enterprise selective disclosure is not a central evaluated property.	Prototype, use-case, security, or cost analyses, depending on the study.
Huong et al. [57]	Package and handoff data are submitted by workflow participants; no threshold oracle rule is reported.	ERC-721 package representation.	Seller–shipper–buyer cash-on-delivery workflow.	Public-chain assumptions; enterprise confidentiality is not the primary focus.	Functional model and implementation-oriented validation.
Proposed 3T architecture	Threshold-gated product tokenization and event-specific evidence references through proofRef.	Dynamic non-fungible product state linked to an auditable lifecycle history.	Typed event tuple, $\text{Auth}(u, e, x)$, $\text{Pre}(e, x)$, $\text{Valid}(e)$, transition function δ , and Fabric endorsement mapping.	Permissioned identities and endorsement; the evaluated one-channel prototype does not implement field-level selective disclosure.	Controlled throughput, latency, CPU, success rate, saturation, and trace query evaluation; no matched competing-system baseline.

Interpretation note. The table records mechanisms explicitly specified or evaluated in the cited publications as characterized in the preceding review. A mechanism marked as not reported is not assumed to be technically impossible on the underlying platform. Numerical performance values are not compared across studies because their hardware, network topology, transaction semantics, endorsement or consensus configuration, and measurement procedures are not equivalent.

The proposed 3T architecture addresses these identified limitations, as outlined in the contributions presented in the Introduction. The next section introduces the architecture, explaining how its design is directly motivated by these limitations before presenting its implementation and validation using Hyperledger Fabric.

4. Proposed 3T Architecture

This section presents the proposed architecture aligned with the requirements introduced in Sections 2 and 3, namely: (i) physical-to-digital anchoring through oracles, (ii) explicit event semantics and authorization, and (iii) implementation-level automation through smart contracts on a permissioned blockchain. We structure the system using the 3T model (Tokenize–Track–Trace), where products are represented as dynamic state objects (dNFT) and lifecycle updates are committed as auditable events (Figure 3).

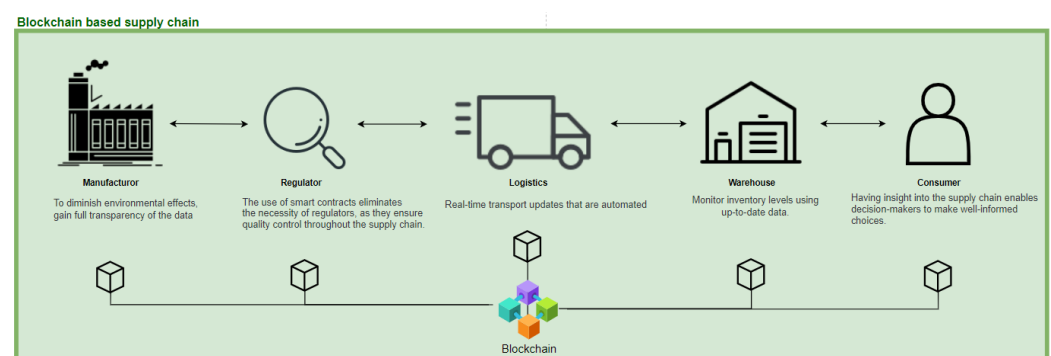


Figure 3. Blockchain integration to supply chain.

Distinct from static NFTs, which primarily encode provenance and ownership without reflecting operational changes, dynamic NFTs allow controlled updates of asset attributes. This enables the representation of status changes (e.g., location, condition, custody) while preserving a unique on-chain identity. Such controlled updates support transparency and can contribute to more efficient operations (e.g., reduced disputes, faster recalls) when tied to explicit event semantics and validation [59]. Moreover, the synergy between dynamic NFTs and smart contracts enables event-driven automation of inter-organizational

workflows (e.g., custody transfer, delivery confirmation, conditional release), thereby reducing administrative overhead and improving operational responsiveness.

4.1. 3T Architecture: Tokenize, Track, and Trace

The 3T architecture digitizes each physical product $p \in \mathcal{P}$ as an on-chain traceability object $x_p \in \mathcal{X}$ (dNFT state), and records lifecycle changes as a sequence of admitted and committed events. Let $\mathcal{E}$ denote the set of logistics events and let $\text{Hist}(p) = \langle e_1, \dots, e_k \rangle$ be the event history.

Tokenize:

Tokenization maps a physical product to an initial on-chain state conditioned on oracle validation:

$$x_p = f_{\text{tok}}(p) \quad \text{s.t.} \quad \text{Valid}(p) = 1, \quad (8)$$

where $\text{Valid}(p)$ follows the threshold oracle rule (Equation (5)).

Track:

Tracking applies event-triggered state transitions:

$$x_i = \delta(x_{i-1}, e_i), \quad i \in \{1, \dots, k\}, \quad (9)$$

where $\delta(\cdot)$ is defined only if authorization and validation predicates hold and the business-state precondition $\text{Pre}(e_i, x_{i-1}) = 1$ is satisfied (Section 4.2).

Trace:

Tracing provides auditable extraction of product history:

$$\text{Hist}(p) = q(x_p) = \langle e_1, e_2, \dots, e_k \rangle. \quad (10)$$

In a generic supply chain network instrumented with the 3T model (illustrated later in Section 5.4), oracles and smart contracts maintain a verifiable custody chain. At tokenization time, validators confirm the existence and required attributes of the physical product; after validation, the product is represented on-chain as a dNFT state capturing identifiers and required metadata. At each operational milestone (e.g., pickup, warehouse check-in/out, delivery), an authorized actor emits an event, which is committed on-chain and appended to the product's history. The final delivery confirmation event closes the custody chain and enables automated settlement when required.

4.2. Event Semantics and Authorization Model

4.2.1. Event Model

We represent a logistics event as:

$$e = (\text{type}, \text{id}, \text{emitter}, t_e, \text{payload}, \text{proofRef}), \quad (11)$$

where id is the product identifier, emitter is the authenticated identity, t_e is a timestamp, payload contains event-specific data (e.g., location, condition), and proofRef optionally references oracle evidence or external documentation.

4.2.2. Role-Based Authorization

Let $\text{role}(u) \in \{\text{Supplier}, \text{Transporter}, \text{Warehouse/Hub}, \text{Client}, \text{Oracle}\}$ be the role bound to identity u by the membership subsystem (e.g., MSP in permissioned settings). For each event type, the system enforces an authorization predicate:

$$\text{Auth}(u, e, x) \in \{0, 1\}, \quad (12)$$

which must evaluate to 1 before committing the event and applying the corresponding state transition. For example, only the *Supplier* may emit tokenization events, only the current custodian may emit *LocationUpdate* events, only the *Warehouse/Hub* may emit warehouse check-in/check-out events, only the *Client* may emit *ConfirmDelivery*, and return-flow events (*ReturnCheckIn*, *RecallReturn*) may be emitted only by the actor receiving the returned product (warehouse or supplier, respectively).

4.2.3. State-Transition Precondition

Authorization alone does not make an event a valid successor of the current product state. We therefore define the business-state predicate

$$\text{Pre}(e, x) \in \{0, 1\}, \quad (13)$$

which evaluates whether event e is admissible from state x . For example, a custody transfer must be initiated by the current custodian and target a different authorized custodian, while *ConfirmDelivery* is admissible only before the product has reached a terminal delivered state. This predicate is a specification-level requirement of the revised 3T model: a compliant tracking implementation accepts a transition only when $\text{Auth}(u, e, x) = 1$, $\text{Pre}(e, x) = 1$, and, for evidence-dependent event types, $\text{Valid}(e) = 1$. The distinction separates identity and role authorization from business-state consistency and specifies the rule needed to reject overlapping or duplicate lifecycle operations; the present benchmark does not separately test every such boundary case.

4.2.4. Validation Predicate for Anchored Events

Certain events may require evidence under a deployment-specific policy. We define:

$$\text{Valid}(e) \in \{0, 1\}, \quad (14)$$

In the present case study, the two-of-two threshold is implemented for product tokenization through $\text{Valid}(p)$ (Equation (5)). For downstream events, $\text{Valid}(e)$ is a specification-level policy predicate. The prototype data model records *proofRef* and supports the configured application-level reference or approval check, but the study does not claim retrieval and cryptographic validation of external documents, independent device signatures, timestamp freshness, nonce uniqueness, or replay-resistant event binding. These stronger controls are specified as a future integration strategy in Section 4.4.4.

4.3. Formal Role of the Model and Event-Acceptance Condition

The equations in Sections 4.1 and 4.2 are specification-level constructs: they define admissible product creation, event acceptance, state evolution, and history reconstruction. They are not presented as new cryptographic primitives, consensus protocols, optimization models, or independent theoretical results. Their purpose is to make the architecture's acceptance conditions explicit and to permit an implementation to be checked against them.

Let $\text{Req}(e) \in \{0, 1\}$ indicate whether the type of event e requires external evidence. The complete event-acceptance condition is defined as

$$\begin{aligned} \text{Acceptable}(u, e, x) = & \mathbb{I}(\text{emitter}(e) = u) \text{Auth}(u, e, x) \text{Pre}(e, x) \\ & \times (1 - \text{Req}(e) + \text{Req}(e) \text{Valid}(e)). \end{aligned} \quad (15)$$

Equation (15) is a composite specification, not a separate correctness proposition. It requires the authenticated invoker to match the event emitter, the role-authorization and business-state predicates to hold, and the configured evidence predicate to hold whenever the event type requires evidence. Because the transition function δ is deterministic, replaying the

same ledger-ordered sequence of accepted events from the same initial state yields the same reconstructed product state.

4.4. 3T Architecture Components

The architecture depicted in Figure 4 is platform-independent only at the level of the product state, event tuple, threshold-attestation rule, authorization and state-transition predicates, and history semantics. It is not implementation-agnostic: identity binding, organizational approval, privacy, transaction finality, evidence verification, and operational cost must be realized with platform-specific primitives. The present implementation uses Hyperledger Fabric for these concerns.

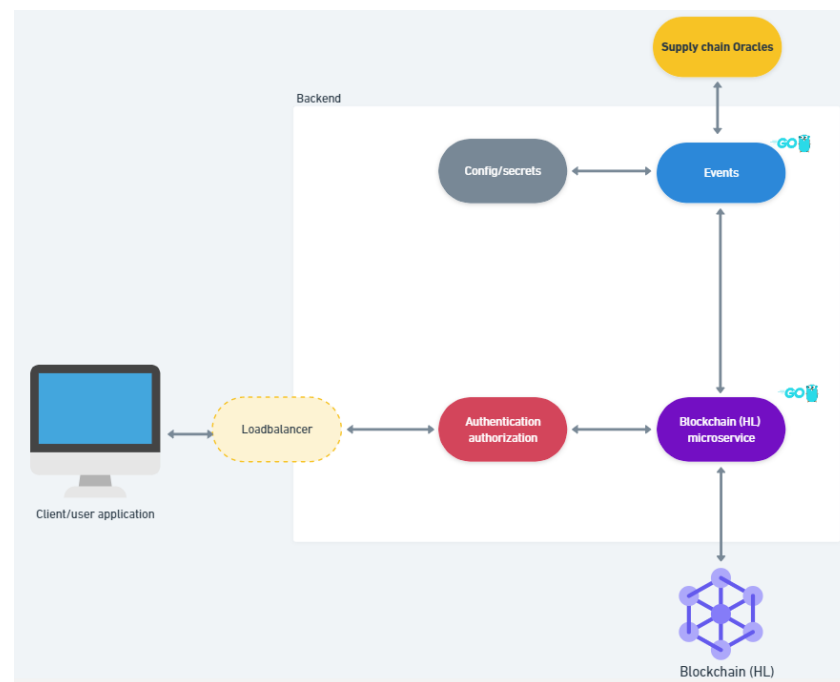


Figure 4. The 3T architecture.

4.4.1. Portability to EVM-Compatible Platforms

An EVM deployment would preserve the abstract 3T semantics but replace the Fabric-specific realization. The current product state would be represented by Solidity structures indexed by a unique product identifier, optionally coupled to an ERC-721 token identifier, while immutable Solidity events would expose the trace sequence for indexed retrieval. The authenticated emitter would be derived from the transaction signer, and $\text{Auth}(u, e, x)$ would be implemented through an on-chain role registry or access-control contract. The threshold rule would be implemented by verifying distinct validator signatures over a canonical, product- and event-bound message before executing the transition. Fabric endorsement policies would therefore be replaced by contract-level multisignature or threshold-approval logic, and Fabric read-version validation would be replaced by explicit state preconditions and, where optimistic concurrency is required, an expected state version or nonce. The evidence hash or content address represented by proofRef could remain on-chain while the full evidence remains off-chain. No fundamental redesign of the event tuple or transition semantics is required; however, privacy, finality, gas and storage cost, identity governance, and history-query design must be re-evaluated for the selected EVM network. This mapping is a proposed portability strategy and has not been implemented or benchmarked in the present study.

This architecture employs a range of services to facilitate interaction with oracles and on-chain logic. These services use REST APIs to retrieve data and events from the

oracle layer and invoke smart contracts through the blockchain Software Development Kit (SDK). Each real-world update is represented as an event $e \in \mathcal{E}$ following Equation (11). The revised model admits a transition only if $\text{Auth}(u, e, x) = 1$, $\text{Pre}(e, x) = 1$, and, for an evidence-dependent event type, $\text{Valid}(e) = 1$. In the evaluated prototype, the available chaincode authorization and transition logic and the configured proofRef handling are mapped to these predicates; however, full branch-level conformance and cryptographic validation of external evidence were not separately demonstrated. Tokenization is the implemented threshold case: $\text{Valid}(p) = 1$ is evaluated using the rule in Equation (5) before the on-chain object is created.

Each component of this architecture has a clear function and maps to the 3T lifecycle. In particular, the Tokenize–Track–Trace workflow is operationalized by Algorithms 1–3.

Algorithm 1 Oracle-validated tokenization (Tokenize)

Require: Physical product $p \in \mathcal{P}$ with identifier $\text{id}(p)$ and attributes $\mathbf{a}(p)$; validator set $\mathcal{V}$; threshold τ ; oracle evidence reference proofRef

Ensure: On-chain state $x_p \in \mathcal{X}$ is created iff $\text{Valid}(p) = 1$

- 1: Let u be the authenticated supplier identity initiating tokenization
 - 2: Collect oracle attestations $\{\sigma_v(p)\}_{v \in \mathcal{V}}$
 - 3: Compute $\text{Valid}(p) \leftarrow \mathbb{I}(\sum_{v \in \mathcal{V}} \sigma_v(p) \geq \tau)$ ▷ Equation (5)
 - 4: **if** $\text{Valid}(p) = 0$ **then**
 - 5: **return reject**
 - 6: **end if**
 - 7: Set proofRef as a reference to the oracle evidence (e.g., hash/pointer to the collected validator decisions and metadata)
 - 8: Initialize $x_p \leftarrow (\text{id}(p), \mathbf{a}(p), \text{owner} = u, \text{status} = \text{Manufactured}, \text{histRef} = \emptyset)$
 - 9: Invoke $\text{CreateAsset}(\text{id}(p), \mathbf{a}(p), \text{proofRef})$
 - 10: Append event $e_1 = (\text{Tokenized}, \text{id}(p), u, t_e, \mathbf{a}(p), \text{proofRef})$ to history
 - 11: **return accept**
-

Algorithm 2 Authorized event registration (Track)

Require: Invoker identity u with role $\text{role}(u)$; event $e = (\text{type}, \text{id}, \text{emitter}, t_e, \text{payload}, \text{proofRef})$

Ensure: Event committed and state updated only if the identity, authorization, state, and configured evidence conditions hold

- 1: Retrieve current state $x \leftarrow \text{ReadAsset}(\text{id})$
 - 2: **if** $\text{emitter} \neq u$ **then**
 - 3: **return reject** ▷ Emitter–invoker mismatch
 - 4: **end if**
 - 5: **if** $\text{Auth}(u, e, x) = 0$ **then**
 - 6: **return reject** ▷ Authorization failed
 - 7: **end if**
 - 8: **if** $\text{Pre}(e, x) = 0$ **then**
 - 9: **return reject** ▷ State precondition failed
 - 10: **end if**
 - 11: $v_e \leftarrow \text{EvaluateEvidencePolicy}(e, \text{proofRef})$
 - 12: **if** $v_e = 0$ **then**
 - 13: **return reject** ▷ Evidence policy failed
 - 14: **end if**
 - 15: Compute $x' \leftarrow \delta(x, e)$
 - 16: Invoke $\text{RegisterEvent}(\text{id}, \text{type}, \text{payload}, \text{proofRef})$ and persist x'
 - 17: **return accept**
-

Algorithm 3 Trace query and policy verification (Trace)**Require:** Product identifier id ; verification policy Π **Ensure:** History $Hist(id)$ and policy-check outcome

```

1: Retrieve history  $Hist \leftarrow GetHistory(id) = \langle e_1, \dots, e_k \rangle$ 
2: for  $j = 1$  to  $k$  do
3:    $a_j \leftarrow CheckLedgerAttribution(e_j)$ 
4:   if  $a_j = 0$  then
5:     return ( $Hist$ , not-policy-compliant)
6:   end if
7:    $v_j \leftarrow EvaluateEvidencePolicy(\Pi, e_j, proofRef(e_j))$ 
8:   if  $v_j = 0$  then
9:     return ( $Hist$ , not-policy-compliant)
10:  end if
11: end for
12: return ( $Hist$ , policy-compliant)

```

4.4.2. Supply Chain Oracles: Integrating Real-World Data into Blockchain Systems

Supply chain oracles serve as connectors between external data sources and the blockchain network, enabling traceability of goods and assets [60]. In our dNFT-based model, oracles provide the physical-to-digital anchoring required before a product is tokenized and during selected lifecycle events. Concretely, a validator set $\mathcal{V}$ (e.g., a plant manager and a quality-control supervisor at tokenization time, or warehouse and transporter agents for downstream events) produces binary attestations $\sigma_v(p) \in \{0, 1\}$ regarding the existence and required attributes of a physical product p . Tokenization is permitted only when the threshold rule in Equation (5) holds, which is enforced by Algorithm 1. For downstream evidence-dependent events, $proofRef$ carries the configured evidence reference and is evaluated under the application policy expressed by Algorithms 2 and 3; the prototype does not claim cryptographic retrieval and validation of the external evidence itself. This design bridges the physical and digital realms while keeping the trust assumption explicit via the threshold parameter τ .

4.4.3. Prototype Conformance Boundary

Table 4 separates mechanisms demonstrated by the available prototype from specification requirements and proposed deployment extensions. This distinction prevents the formal predicates from being interpreted as evidence that every branch, external evidence check, or storage alternative was implemented and tested.

Table 4. Prototype conformance boundary for the formal 3T elements.

Formal Element	Behavior Supported by the Prototype Description	Not Claimed or Reserved for Future Validation
$Valid(p)$	Two supplier-side validator decisions are recorded and the case-study threshold $\tau = 2$ is evaluated before product creation.	Dynamic validator governance, cross-organization independence, strategic-behavior testing, and Byzantine oracle tolerance.
$Valid(e)$ and $proofRef$	The event model records $proofRef$ as a hash or document pointer and permits a configured reference or approval condition.	Retrieval and cryptographic validation of external evidence, device signatures, freshness, nonce uniqueness, and replay-resistant binding.
$Pre(e, x)$	The predicate specifies the state-admissibility checks required of a compliant transition implementation and maps to the available chain-code transition logic.	Exhaustive documentation, implementation, and branch-level testing of every duplicate, terminal-state, or out-of-order guard.
$Hist(p)$	The asset model includes a lightweight <code>EventHistory</code> array, and every state update is also represented by an immutable Fabric transaction and key version.	Attribution of the reported <code>QueryHistory</code> latency to one internal retrieval path, and byte-level comparison of embedded-history and key-history storage.

4.4.4. IoT Evidence Integration Strategy

The evaluated prototype does not include automated sensing. A production extension could connect GPS/GNSS trackers, RFID or NFC readers, barcode scanners, temperature and humidity probes, shock or tilt sensors, and tamper-evident devices through an authenticated edge gateway using protocols such as MQTT or HTTPS/REST. For each observation, the device or gateway would construct a canonical evidence envelope containing at least the device identifier, product identifier, event type, observation time, sequence number or nonce, measurement payload, and relevant device or calibration metadata. The envelope hash would be digitally signed with the device or gateway private key. The full signed envelope would remain in an authorized off-chain repository, while proofRef would contain its cryptographic hash and retrieval locator or content address. Before invoking RegisterEvent, the Events Service would verify the device certificate and revocation status, signature, product and event binding, timestamp freshness, and sequence number or nonce. This design detects post-capture modification and discourages evidence replay; it does not by itself prevent a compromised or incorrectly calibrated sensor from signing a false measurement. The proposed sensing and binding workflow remains future work and was not included in the reported experiments.

4.4.5. Load Balancer: Gateway for Data Access

The load balancer serves as the entry point for users accessing data through a browser interface. It manages network traffic and distributes requests across application services to improve availability and response times under concurrent load. This component primarily affects system-level performance and does not alter the on-chain trust model.

4.4.6. Authentication/Authorization: Ensuring Secure and Controlled Access

The Authentication/Authorization service performs two security functions:

- Authentication: verifies the identity of users before they can invoke blockchain operations.
- Authorization: enforces access rights according to role-based policies, corresponding to $\text{Auth}(u, e, x)$ in Equation (12).

This ensures that only authorized actors can emit specific event types and update the dNFT state. In permissioned deployments, identities are managed through X.509 certificates issued by organizational certificate authorities and bound to roles by the membership subsystem.

4.4.7. Config/Secrets: Centralized Management of Sensitive Data

The Config/Secrets service provides centralized management of configuration parameters and sensitive credentials (e.g., keys, tokens, endpoints). This improves operational security and deployment consistency across distributed services, particularly when multiple application instances interact with the blockchain SDK and oracle interfaces.

4.4.8. Events Service: Bridging Oracles and Blockchain for Dynamic Data Interaction

The Events Service acts as a conduit between the oracle layer and the blockchain network. In the evaluated prototype, every state-changing command—whether emitted by the supplier, transporter, warehouse, or consumer—is routed to a single application-level writer actor. The actor exclusively owns the mutable application-side command state and processes its mailbox sequentially, so only one application-side command handler is active at a time, and no application-level mutex is required. This actor does not own the replicated Fabric world state and does not replace ledger validation. Each dequeued command is mapped to an event e (Equation (11)) and submitted through the blockchain SDK after application-side ordering. The existing chaincode authorization and transition

logic, the endorsement policy, and the configured evidence-reference handling remain authoritative; the formal predicate $\text{Pre}(e, x)$ specifies the state-admissibility condition expected of that transition logic. The experiment did not instrument the number of already submitted transactions concurrently progressing through Fabric's endorsement, ordering, and validation pipeline, so no stronger claim about ledger-pipeline concurrency is made.

4.4.9. Chaincode/Smart Contracts: Core of Data Storage and Workflow Automation

Chaincode defines the data model and state-transition logic for dNFT states, including methods for creating assets, registering lifecycle events, transferring custody, and querying state and history. In the revised model-to-implementation mapping, the existing tokenization, authorization, and transition logic is interpreted through $\text{Valid}(p)$, $\text{Auth}(u, e, x)$, and $\text{Pre}(e, x)$, respectively. Here, $\text{Pre}(e, x)$ is a specification-level abstraction of the state-dependent checks that a compliant transition implementation must enforce. The chaincode also exposes the state and history queries used for auditing (Algorithm 3); the benchmark evaluates the complete workflow but does not isolate or exhaustively test every individual guard. Efficient query design (e.g., indexing and key structure) is important for retrieving product histories and current state under realistic workloads.

4.5. Integration with Existing Enterprise Information Systems

The 3T architecture is intended to complement, rather than replace, the enterprise information systems that already manage operational processes and master data. The present prototype evaluates the blockchain workflow and does not implement production connectors to enterprise resource planning (ERP), manufacturing execution (MES), warehouse management (WMS), transportation management (TMS), product lifecycle management (PLM), or legacy database platforms. For deployment, we therefore propose an adapter-based integration layer around the Events Service described above.

Enterprise applications, barcode or RFID readers, and IoT gateways can publish business events through REST APIs, message brokers, an enterprise service bus, or platform-specific SDK adapters. Each connector maps a source record to the canonical event tuple in Equation (11), including the product identifier, event type, authenticated emitter, timestamp, event-specific payload, and evidence reference. The mapping should also preserve a source system event identifier so that repeated submissions can be detected and reconciled with the originating record.

The integration layer should enforce an explicit on-chain/off-chain separation. Only traceability-critical identifiers, authorized lifecycle events, state transitions, and evidence hashes or pointers are committed to the ledger. Large documents, commercially sensitive attributes, and operational master data remain in the source enterprise system or an authorized off-chain repository, while their integrity can be anchored through *proofRef*. Enterprise users and service accounts are associated with Hyperledger Fabric X.509 identities, organizations, and roles through certificate authorities and Membership Service Providers, thereby linking existing access-control responsibilities to the predicate $\text{Auth}(u, e, x)$.

After schema validation and authorization, the Events Service submits the mapped transaction through the Fabric SDK and returns the transaction identifier and commit status to the source system. Transient communication or ordering failures can be retried with the same source-event identifier, whereas authorization failures or inconsistencies between the enterprise record and the committed ledger state should be placed in a reconciliation queue for correction or approved manual resolution. Commercial confidentiality can be enforced through channels, private data collections, access-control policies, and application-level encryption, according to which organizations require each data element.

This strategy supports incremental adoption: a consortium can begin with selected products, processes, or organizations and expand the integration after validating data mappings and governance rules. It is presented as a technically credible deployment strategy rather than as an experimentally implemented enterprise integration; production connectors, bidirectional synchronization, and recovery procedures remain to be validated in a multi-system pilot.

4.6. Algorithms

This subsection provides algorithmic descriptions of the 3T lifecycle. Algorithm 1 formalizes *Tokenize* as an oracle-threshold decision followed by on-chain asset creation. Algorithm 2 formalizes *Track* as an authorized event commitment with a configured evidence condition for selected event types. Algorithm 3 formalizes *Trace* as history retrieval, ledger-attribution checking, and policy checking of required evidence references.

4.6.1. Tokenize (Oracle-Validated Asset Creation)

Algorithm 1 creates a new on-chain traceability object only when a sufficient number of validators confirm the physical product. This ensures the physical-to-digital anchoring requirement introduced in Section 2 is enforced before minting the corresponding dNFT state.

4.6.2. Track (Authorized and Conditionally Validated Event Registration)

Algorithm 2 operationalizes the composite acceptance condition in Equation (15): (i) the invoker identity must match the event emitter, (ii) the role-based authorization predicate must hold, (iii) the business-state precondition $\text{Pre}(e, x)$ must hold, and (iv) the configured evidence policy must be satisfied whenever the event type requires evidence. The function `EvaluateEvidencePolicy` returns 1 when no evidence is required and otherwise evaluates the deployment-specific condition represented by $\text{Valid}(e)$. The algorithm is a specification-level procedure; Table 4 states which checks were demonstrated by the prototype. Tokenization is excluded because it is validated separately at the product level in Algorithm 1.

4.6.3. Trace (History Reconstruction and Audit Verification)

Algorithm 3 retrieves the product history, checks ledger attribution, and evaluates the configured `proofRef` condition for event types that require evidence. This is a policy-compliance procedure; it does not imply cryptographic validation of external evidence or proof of the underlying physical statement. Here, $\text{Hist}(\text{id})$ denotes the history associated with the product identifier $\text{id} = \text{id}(p)$. This operationalizes the traceability requirement defined in Section 3.

4.6.4. Computational and Storage Characteristics

Let $n = |\mathcal{V}|$ be the validator-set size, let r_e be the number of authorization and state rules evaluated for event e , let a_e be the number of attestations or configured evidence checks evaluated for that event, and let h be the product history length. Tokenization requires $O(n)$ attestation checks. Tracking requires $O(r_e + a_e)$ application-level checks before ledger processing; r_e is constant only when the policy set is fixed. Trace reconstruction and policy checking require $O\left(h + \sum_{j=1}^h a_{e_j}\right)$ work. These counts exclude platform-dependent endorsement, ordering, replication, and network communication.

Section 5 includes a lightweight `EventHistory` array in the current asset value, while Fabric also preserves each committed asset version in the immutable channel ledger. The experiment reports end-to-end application-level `QueryHistory` latency but does not isolate whether the returned sequence is obtained from the embedded array, the key-history API,

or a combination of both. The storage discussion is therefore a conditional asymptotic analysis rather than a measured byte-level result.

Let s_x denote the fixed serialized size of a product state excluding its embedded history, let s_e denote the mean serialized size of an event reference, and let r denote the number of peers that replicate the channel ledger. If the latest asset value embeds h event references, its replicated world-state content is approximately $r(s_x + hs_e)$. If the complete embedded history is serialized and rewritten after every update, repeated serialization yields cumulative application payload growth of $O(rh^2)$. By contrast, storing immutable events under separate composite keys, or relying on ledger key history while retaining only the current state and evidence hashes or pointers in the primary asset, yields $O(rh)$ content growth. The present benchmark does not instrument byte-level ledger growth or isolate the marginal cost of each rule, evidence check, or storage representation.

Following the detailed exploration of the 3T components and algorithms, the next step is to examine the blockchain infrastructure underpinning our implementation, namely Hyperledger Fabric, which provides governance, access control, and enterprise-grade deployment support.

5. Implementation on Top of Hyperledger Fabric

This section instantiates the proposed 3T model (Section 4) on *Hyperledger Fabric*, a permissioned blockchain framework that provides (i) authenticated identities, (ii) configurable endorsement policies, (iii) transaction ordering, and (iv) ledger/state separation suitable for enterprise SCM deployments. In our implementation, the on-chain traceability object $x_p \in \mathcal{X}$ (dNFT state) is stored in the *world state*. The data model in Figure 8 also contains a lightweight *EventHistory* array, while each update is independently preserved as a committed Fabric transaction and product-key version. These representations are complementary rather than interchangeable: the application-level history provides semantic event references, and the ledger provides immutable transaction evidence. The reported *QueryHistory* benchmark does not isolate its internal retrieval path, so neither representation is claimed as the sole measured source. This mapping preserves the definitions introduced in Section 4.1. The Fabric realization maps authenticated identities and chaincode authorization to $\text{Auth}(u, e, x)$, evaluates the implemented tokenization rule through $\text{Valid}(p)$, treats downstream $\text{Valid}(e)$ as the configured evidence-reference policy, and uses $\text{Pre}(e, x)$ as the state-admissibility requirement against which transition logic is assessed.

The implementation is presented as a validation mapping rather than as a general description of Fabric software components. Table 5 identifies which platform mechanism realizes each formal construct and what aspect of the architecture it makes observable or enforceable.

Table 5. Mapping from 3T constructs to the Hyperledger Fabric realization.

3T Construct	Fabric Realization	Research Implication
Authenticated actor and role(u)	MSP-issued X.509 identity and organization/attribute binding, inspected by chaincode	Makes event attribution and role-dependent authorization enforceable rather than descriptive.
Current state x_p , transition δ , and $\text{Pre}(e, x)$	Product-key world state and deterministic chaincode execution	Provides the implementation locus for the dynamic non-fungible product state and state-admissibility checks; the present benchmark does not isolate or exhaustively test every boundary case.
$\text{Valid}(p)$, $\text{Valid}(e)$, and proofRef	Implemented two-of-two tokenization approval; downstream recording and configured handling of evidence references	Connects the formal evidence policy to the transition while explicitly excluding unsupported claims of external cryptographic evidence validation or physical truth.

Table 5. Cont.

3T Construct	Fabric Realization	Research Implication
Inter-organizational approval	Endorsement policy EP and matching proposal responses	Separates the actor permitted to request an event from the organizations required to approve its state change.
Hist(p) and trace audit	Embedded EventHistory references plus immutable channel transactions and product-key versions	Supports semantic event reporting and ledger attribution; the benchmark does not isolate the internal QueryHistory retrieval path.
Concurrent transition safety	Read/write versioning and commit-time validation	Invalidates stale writes; business-level duplicate or out-of-order events still require $\text{Pre}(e, x)$.
Confidential data boundary	One shared channel in the experiment; channels and private data collections as deployment mechanisms	Delimits the privacy claim: field-level selective disclosure was not evaluated in the prototype.

5.1. Hyperledger Fabric Components

A Hyperledger Fabric network is composed of the following main building blocks:

- Ledger: a tamper-evident record of state transitions resulting from chaincode invocations. It comprises (i) the *world state* and (ii) the immutable blockchain log.
- World state: a database storing the latest values of ledger keys (typically key–value pairs). In our case, the key is the product identifier $\text{id}(p)$ and the value is a serialized representation of x_p .
- Blockchain log: an append-only sequence of blocks storing committed transactions and product-key versions; it provides immutable attribution and version history, while the application data model also exposes lightweight semantic EventHistory references.
- Channels: logical partitions of the network enabling private data sharing among a subset of organizations; all peers on a channel maintain a consistent ledger for that channel.
- Chaincode: the smart-contract layer implementing the data model and the transition function $\delta(\cdot)$, including CreateAsset and RegisterEvent.
- Ordering service: establishes a total order of endorsed transactions and packages them into blocks for dissemination to channel peers.
- Endorsing peers: execute chaincode proposals (simulation) and sign results according to the endorsement policy.
- Committing peers: validate ordered transactions (endorsement + read-write consistency) and update the world state and blockchain log.
- Endorsement policy: a rule specifying which organizations must endorse a transaction before it can be committed.
- Membership Service Provider (MSP): binds identities to organizations and roles; it is the foundation used to enforce $\text{role}(u)$ and evaluate authorization constraints.
- Application SDK: client libraries used to assemble proposals, collect endorsements, submit transactions to the ordering service, and receive commit notifications.

5.1.1. State Mapping to the 3T Model

Let WS denote the world state as a mapping from keys to values:

$$\text{WS} : \mathcal{K} \rightarrow \mathcal{V}. \quad (16)$$

For each product $p \in \mathcal{P}$, we store x_p under key $\text{id}(p)$:

$$\text{WS}[\text{id}(p)] = \text{Serialize}(x_p). \quad (17)$$

Each lifecycle update is represented by a committed transaction and key version. Algorithm 3 abstracts the application-level history query over these available representations; the experiment does not isolate their respective latency or storage contributions.

5.1.2. Endorsement Policy as a Validation Predicate

Let $\mathcal{O}$ be the set of organizations on a channel and let $S \subseteq \mathcal{O}$ denote the set of organizations that provided valid endorsements for a proposal. We represent an endorsement policy as a Boolean predicate $EP(\cdot)$ over endorsement sets. A transaction is endorsement-valid if:

$$\text{EndorseOK}(S) = EP(S) = 1. \quad (18)$$

In our setting, $\text{EndorseOK}(\cdot)$ complements the application-layer authorization predicate $\text{Auth}(u, e, x)$ (Equation (12)): Auth restricts *who may request* an event type, while EndorseOK enforces *which organizations must approve* the resulting state transition before commitment.

5.2. Transaction Flow

The proposal–endorsement–ordering–validation pipeline is retained because it explains how the abstract acceptance predicates become commit conditions and where the measured latency and concurrency behavior arise.

Before a transaction is committed on a Fabric channel, it progresses through the standard proposal–endorsement–ordering–validation pipeline (Figure 5). We summarize the steps and explicitly relate them to the 3T lifecycle (Algorithms 1–3):

1. Proposal submission (client $\rightarrow$ endorsers). The client (SDK) creates a chaincode invocation proposal for `CreateAsset` (Tokenize) or `RegisterEvent` (Track), signed by the invoker identity u .
2. Verification and simulation (endorsers). Endorsers verify identity and channel permissions (via MSP), then simulate chaincode execution to produce a *read set* and a *write set*. In the revised model-to-implementation mapping, the existing chaincode authorization and transition logic is interpreted through $\text{Auth}(u, e, x)$ and the specification-level condition $\text{Pre}(e, x)$. The implemented tokenization approvals are evaluated through $\text{Valid}(p)$; downstream $\text{Valid}(e)$ denotes the configured evidence-reference condition rather than a claim of external cryptographic evidence verification.
3. Endorsement collection (client). The client checks that proposal responses match and collects enough endorsements to satisfy $\text{EndorseOK}(S) = 1$ (Equation (18)).
4. Ordering (ordering service). The endorsed transaction is broadcast to the ordering service, ordered with other transactions, and packed into blocks.
5. Validation and commitment (peers). Committing peers validate endorsement satisfaction and perform a read-write consistency check (e.g., MVCC-style validation) to ensure that the read set has not been invalidated by a concurrent committed transaction. Valid transactions update the blockchain log and apply the write set to the world state (Equation (17)).

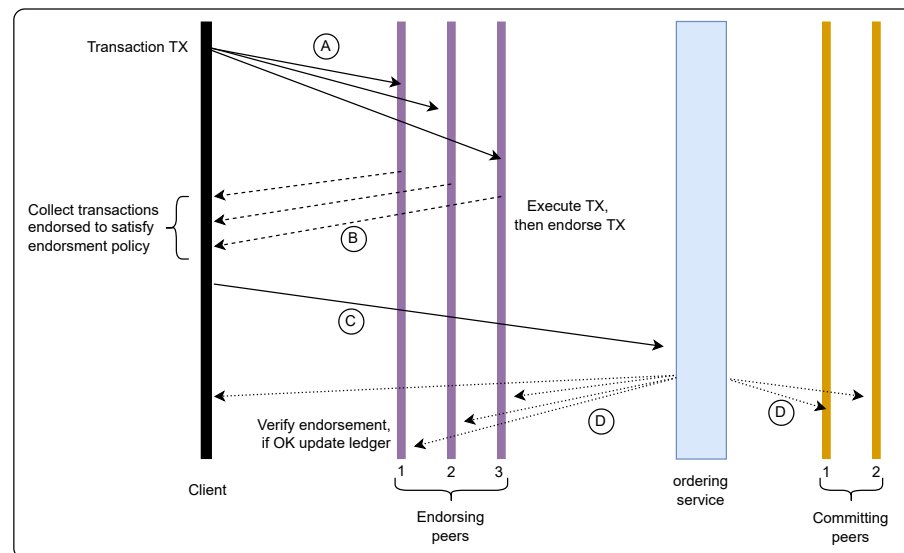


Figure 5. Hyperledger Fabric transaction flow (proposal, endorsement, ordering, validation, commitment).

5.2.1. Concurrent Updates and Duplicate Events

Messages from different supply chain actors may reach the application concurrently, but the current prototype routes them to one writer actor and processes them sequentially from its mailbox. This serializes application-side command handling and transaction submission, prevents simultaneous mutation of application-owned state, and removes the need for an in-process lock. The writer is not a global lock over the Fabric world state and does not control transactions submitted through any future independent gateway. The benchmark also did not instrument the number of already submitted transactions concurrently progressing through Fabric's internal pipeline. If competing transactions nevertheless target the same product-key version, Fabric's commit-time multi-version concurrency control marks a stale transaction invalid and does not apply its write set.

Business semantics require a separate state check. In the revised formal model, a compliant implementation evaluates the latest committed custodian and terminal status before accepting a subsequent transfer or delivery command. Consequently, an incompatible custody transfer or duplicate `ConfirmDelivery` should be rejected through $\text{Pre}(e, x)$ even when the invoker has a valid identity and role. A client may retry an MVCC-invalid transaction only after reading the latest committed state and confirming that $\text{Auth}(u, e, x)$ and $\text{Pre}(e, x)$ still hold. The reported workload did not include a dedicated experiment with overlapping custody transfers or duplicate delivery confirmations; this is therefore a specified handling rule rather than an empirically demonstrated conflict result. The single writer simplifies application concurrency but can introduce a throughput bottleneck and availability dependency; a production extension could partition writers by product identifier while preserving one writer per product and adding durable mailbox recovery.

5.2.2. Fabric-Aligned Execution Procedure for 3T Operations

Algorithm 4 refines the abstract lifecycle procedures (Algorithms 1 and 2) by making explicit (i) endorsement policy satisfaction and (ii) commit-time read/write validation, which are central to reproducible evaluation on Fabric.

Algorithm 4 Fabric execution flow for 3T transactions (Tokenize/Track)

Require: Invoker identity u ; chaincode function $fn \in \{\text{CreateAsset}, \text{RegisterEvent}\}$; inputs in ; endorsement policy EP

Ensure: Transaction committed iff authorization, state-transition precondition, (optional) oracle validation, endorsement, and commit checks succeed

- 1: Client SDK builds proposal $\pi \leftarrow (fn, in)$ and signs it with u
- 2: Send π to endorsing peers and collect responses $\{(rw_j, \sigma_j)\}_j$
- 3: **Client check:** verify signatures σ_j and response equivalence; derive endorsement set S
- 4: **if** $EP(S) = 0$ **then** ▷ Equation (18)
- 5: **return reject**
- 6: **end if**
- 7: Submit endorsed transaction tx to ordering service
- 8: Ordering service sequences tx into a block and broadcasts to committing peers
- 9: **Commit-time validation:** peers validate endorsements and verify read/write consistency
- 10: **if** validation fails **then**
- 11: Mark tx invalid; do not apply write set
- 12: **return reject**
- 13: **else**
- 14: Append tx to blockchain log; apply write set to world state WS
- 15: **return accept**
- 16: **end if**

5.3. Choice of Hyperledger Fabric

We select Hyperledger Fabric because its design primitives directly support the requirements of the proposed 3T architecture:

- Authenticated identities and role governance: permissioned identities (MSP) enable explicit binding of users to roles $role(u)$ and systematic enforcement of $Auth(u, e, x)$ at the smart-contract layer.
- Configurable endorsement for inter-organizational control: endorsement policies (Equation (18)) provide a native mechanism to require cross-organization approval for sensitive state transitions (e.g., custody transfer or delivery confirmation).
- Privacy via channels: channels support partitioned ledgers for business confidentiality, consistent with SCM contexts where not all stakeholders should observe all events.
- Separation of concerns (execute–order–validate): the transaction pipeline supports controlled performance experiments by isolating simulation/endorsement, ordering, and validation/commitment phases, improving interpretability of throughput/latency measurements.
- Smart contracts in general-purpose languages: chaincode can be implemented in widely used languages (e.g., Go/Java/Node.js), reducing engineering barriers for enterprise deployment.

Selective Disclosure and Confidential Attributes

The evaluated configuration uses one shared channel and does not provide field-level selective disclosure: peers authorized to maintain that channel receive the shared asset and event fields, so application-level query restrictions alone do not conceal those fields from the participating organizations. A production data model should separate a minimal shared traceability record—product identifier, event type, authorized emitter, timestamp, state transition, and evidence hash—from confidential attributes such as prices, personal identities, commercial documents, or detailed sensor payloads. Stable whole-ledger partitions can use separate channels, whereas organization-specific or subset-specific attributes are better placed in Fabric private data collections, with only their hashes committed to

the shared ledger [37]. Additional options include encrypting off-chain attributes under recipient-specific keys and storing only a hash and locator on-chain. Zero-knowledge proofs could support selected predicates, such as demonstrating that a temperature remained within an accepted range without revealing the full series, but this would add circuit, key management, and verification overhead. These selective disclosure mechanisms were not implemented or evaluated in the present prototype.

The next section presents the case study and experimental configuration used to evaluate the proposed system under controlled workloads and explicitly stated parameters.

5.4. Case Study Overview

This section outlines a case study to demonstrate the practicality and effectiveness of our proposed supply chain architecture. We establish a simplified yet representative supply chain network, including essential entities (Figure 6): a supplier, transporter, warehouse, and consumer. While this model is basic, it serves as a foundation that can be extended to more complex and larger networks.

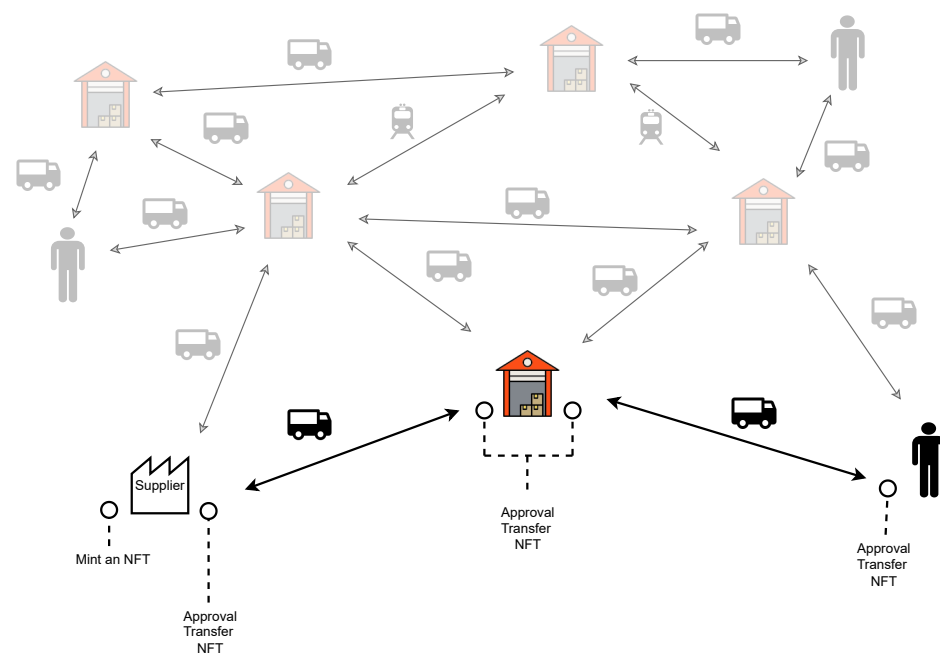


Figure 6. Generic supply chain network.

5.4.1. Network Configurations and Participant Roles

The network is composed of these four key organizations, collaboratively establishing a Hyperledger Fabric network, detailed in Figure 7. The four participants all need secure communication within the network. Every peer will keep a replica of the ledger associated with channel C. Network configuration NC1 is the policy that governs the network, which is managed by the four organizations. Similarly, channel configuration CC1 is the policy that governs the channel and is also managed by the four organizations. The ordering service O1 is joined to the application channel through the channel participation API (the legacy system channel is not used, consistent with Hyperledger Fabric 2.5.x), and orders transactions into blocks for distribution to the committing peers. Each of the four organizations has its own preferred Certificate Authority.

5.4.2. Incorporating Oracle Configuration for Product Verification

A notable aspect of our network is the implementation of our oracle configuration at the supplier node. Within the supplier node, the oracle policy mandates that two designated

agents, such as the plant manager and quality control supervisor, authenticate the realness of each product. This policy ensures that the product's physical attributes are accurately matched to its digital counterpart. To align with the oracle model introduced in Section 2.2, we consider a validator set $\mathcal{V}$ and binary validator attestations $\sigma_v(p) \in \{0, 1\}$. In this case study, we set the threshold to $\tau = 2$, so that tokenization is permitted only if:

$$\text{Valid}(p) = \mathbb{I}\left(\sum_{v \in \mathcal{V}} \sigma_v(p) \geq \tau\right), \quad \tau = 2, \quad (19)$$

which is consistent with Equation (5) and enforced by Algorithm 1.

5.4.3. Case-Study Initialization and Product Scenarios

For this case study, we initialize the ledger with a small set of reference product records used to illustrate different traceability situations. The first product, ProductX, is kept as the baseline scenario and is initially allocated to the supplier as a single traceable unit. Three additional product records are also introduced to demonstrate that the proposed mechanism is not limited to a single product journey: ProductY represents an incomplete delivery case where the product is currently stored at the warehouse; ProductZ represents a recall or return case; and ProductW represents an alternative route in which the product is transferred directly from the supplier to the warehouse before final delivery.

In Hyperledger Fabric, the *genesis* configuration block bootstraps the channel and does not encode application assets. Therefore, these products are created through controlled *bootstrap application transactions* on channel C, corresponding to initial invocations of the asset-creation chaincode function. This initialization provides a deterministic starting point for the experimental scenario while preserving the same tokenization semantics used for subsequently manufactured products.

For each product $p \in \{\text{ProductX}, \text{ProductY}, \text{ProductZ}, \text{ProductW}\}$, the corresponding dNFT is minted only after the product has successfully passed the oracle validation process, i.e., only if $\text{Valid}(p) = 1$. This validation step ensures that the product's real-world existence and required specifications are authenticated before its digital representation is committed to the ledger. Upon validation, the blockchain ledger records the supplier as the initial owner of the product and appends the first lifecycle event to its history.

The resulting product records, structured according to the dNFT asset model shown in Figure 8, are then ready for tracking and traceability operations throughout their respective supply chain journeys. Subsequent lifecycle events, such as custody transfer, location update, warehouse check-in, delivery confirmation, return check-in, or recall return, are appended to the product history as committed *EventRef* entries. This multi-product initialization supports the reconstruction of different traceability paths while maintaining a unified 3T logic across all products.

5.5. Results of the Experiments

5.5.1. Infrastructure Performance Evaluation

Our experiments aim to assess the performance of the proposed Hyperledger Fabric-based supply chain network in the context of the four-actor case study composed of a supplier, a transporter, a warehouse, and a consumer. The evaluation focuses on the main operations of the Tokenize–Track–Trace lifecycle: product tokenization, custody transfer, event registration, warehouse handling, delivery confirmation, and traceability queries. The objective is to validate the network's capacity to support dynamic supply chain operations under controlled workload-generation conditions.

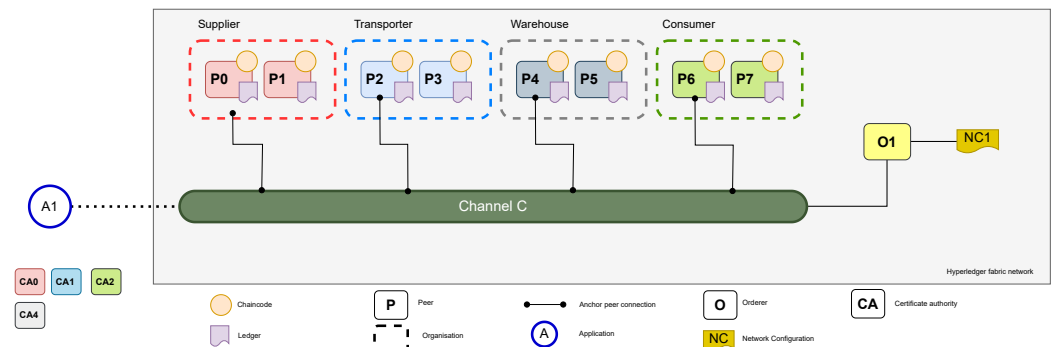


Figure 7. Application of the 3T architecture on the case study.

```

type EventRef struct {
    TxID      string `json:"txId"`
    Type      string `json:"type"`           // MINT, TRANSFER, LOCATION, DELIVERY, ...
    Emitter   string `json:"emitter"`       // MSP ID or participant ID
    Time      string `json:"time"`         // RFC3339 timestamp
    ProofRef  string `json:"proofRef"`     // hash / document pointer
}

type Asset struct {
    ID          string `json:"id"`
    Owner       string `json:"owner"`
    ManufacturerID string `json:"manufacturerId"`
    ManufactureDate string `json:"manufactureDate"`
    BatchNumber string `json:"batchNumber"`
    Quantity    int    `json:"quantity"`
    ValidationStatus bool `json:"validationStatus"`
    ValidatorIDs []string `json:"validatorIds"`
    EventHistory []EventRef `json:"eventHistory"`
}

```

Figure 8. Excerpt of the chaincode data model used to represent assets (dNFTs) and event references.

5.5.2. Hardware Specifications

The evaluation was carried out on a local hardware infrastructure. The specifications include:

- Processor: Intel(R) Core(TM) i7-10875H CPU @ 2.30 GHz.
- Memory: 32 GB, ensuring efficient data processing and multitasking capabilities.
- Storage: 1 TB SSD, providing quick data access and storage for blockchain data.
- Network Connection: Gigabit Ethernet, facilitating local communication within the experimental network.

This hardware setup corresponds to a controlled local deployment rather than a production-scale distributed cluster. Therefore, the reported results should be interpreted as a feasibility-oriented benchmark of the proposed 3T workload under a single-machine Docker-based configuration.

5.5.3. Experimental Configuration and Workload Design

The experimental network follows the case-study configuration introduced earlier. It is composed of four organizations: Supplier, Transporter, Warehouse, and Consumer. Each organization owns two peers, resulting in eight peers connected to one shared application channel. One ordering node is used in a local single-orderer configuration to order transactions into blocks and distribute them to the committing peers. The workload is generated through a fixed-rate open-loop client application that submits commands at increasing target rates to the single-writer Events Service, which processes application commands one

at a time and invokes the Fabric Gateway client API. The benchmark did not separately record the number of submitted transactions concurrently resident in the Fabric pipeline (Table 6).

Table 6. Experimental configuration used for the controlled workload.

Parameter	Value
Blockchain framework	Hyperledger Fabric 2.5.x
Deployment type	Docker-based local permissioned network
Organizations	4: Supplier, Transporter, Warehouse, Consumer
Peers	8 peers in total: 2 peers per organization
Ordering service	1 ordering node in a local single-orderer configuration
Channel	One shared application channel
State database	LevelDB
Chaincode language	Go
Asset model	Dynamic asset record representing the product dNFT state
Oracle policy	Two supplier-side validators: plant manager and quality-control supervisor
Tokenization rule	Product is minted only if both validators approve, i.e., $\tau = 2$
Endorsement policy	Actor-organization endorsement for event registration; source–destination endorsement for custody transfer
Block configuration	Maximum 50 transactions per block; batch timeout of 500 ms
Workload generator	Fixed-rate open-loop client submitting commands to the single-writer Events Service, which invokes the Fabric Gateway client API
Submission policy	Target send rate is controlled per stage; no per-stage tuning of chaincode, endorsement policy, block size, or batching parameters
Product-key generation	Independent product identifiers; no shared global counter or intentional hot-key update
Measurement window	10 min per workload stage
Warm-up period	2 min per stage, excluded from measurement
Repetitions	3 runs per stage; mean values are reported
Reported throughput	Committed valid write transactions per second
Commit deadline	30 s per transaction
Failure accounting	Transactions not committed within the 30 s deadline are counted as unsuccessful; invalid transactions are recorded separately

The workload reproduces the operational lifecycle of a product moving from the supplier to the final consumer. Each product is first tokenized as a dNFT after oracle validation. Then, the product is transferred to the transporter, updated during transportation, checked in and out at the warehouse, and finally delivered to the consumer. The write workload mix is summarized in Table 7. Read-only traceability queries are executed separately and are not included in the committed write-throughput calculation.

Table 7. Write workload mix used during the experiment.

Operation	Actor	Share
<i>CreateAsset / MINT</i>	Supplier	20%
<i>TransferAsset</i>	Supplier → Transporter	15%
<i>LocationUpdate</i>	Transporter	20%
<i>WarehouseCheckIn</i>	Warehouse	15%
<i>WarehouseCheckOut</i>	Warehouse	15%
<i>ConfirmDelivery</i>	Consumer	15%

To make the workload stages comparable, the same chaincode, endorsement policies, block configuration, product-key generation strategy, and measurement window are used across all stages. Only the target offered load is varied. Product identifiers are generated independently, and the workload generator avoids intentional simultaneous conflicting writes to the same product state. This design reduces artificial multi-version concurrency control conflicts and makes overload behavior easier to interpret.

Accordingly, this benchmark characterizes the disclosed single-writer workload and predominantly low-contention product-key access. It does not constitute a systematic test of parallel same-product proposals, overlapping custody transfers, duplicate delivery confirmations, or the number of concurrently in-flight Fabric transactions. Section 5.2.1

distinguishes mailbox serialization, Fabric MVCC validation, and the formal semantic rule $\text{Pre}(e, x)$; conflict-heavy operation remains a separate experimental limitation.

5.5.4. Evaluation Claim and Comparative Boundary

The experiment is designed to determine whether the Fabric realization executes the complete 3T workload reproducibly and to identify its saturation and trace query behavior under the disclosed configuration. It does not demonstrate that 3T has higher throughput or lower cost than a competing system. Table 3 therefore provides a capability-based comparison, while direct numerical comparisons with published results are avoided because the hardware, topology, transaction mix, consensus or endorsement policy, and measurement procedures differ.

A fair quantitative attribution of 3T overhead would require a same-testbed baseline or ablation that holds the network and workload constant while comparing: (i) a minimal asset-state update, (ii) an update with identity, role, and state checks, and (iii) the complete 3T update with evidence verification and history handling. Such an ablation, together with multi-host, larger-consortium, multi-channel, and conflict-heavy workloads, was not part of the available experimental data and is identified as future validation rather than reported as a completed result.

5.5.5. Performance Metrics

Our performance evaluation focuses on:

- Transaction throughput (TPS): the number of committed valid transactions per second. Throughput is computed as $TPS = N / \Delta t$, where N is the number of committed valid transactions and Δt is the measurement window.
- Transaction latency: the elapsed time between client submission and commit notification. For each transaction i , latency is computed as $L_i = t_{\text{commit}}^{(i)} - t_{\text{submit}}^{(i)}$. We report mean, median, p95, and p99 latency.
- Load-scaling and saturation under a fixed topology: the response of the disclosed network configuration to increasing offered load before and after the observed saturation region.
- Resource efficiency: CPU utilization under increasing transaction loads, used as an indicator of computational demand during transaction processing.
- Success rate: the percentage of submitted transaction attempts that are committed as valid transactions within the measurement window. Since the workload generator reaches the target submission rate during each measurement stage, success rate is computed as $SR = (TPS_{\text{committed}} / TPS_{\text{target}}) \times 100$.

5.5.6. Results Overview

The developed infrastructure's performance results are illustrated in Figure 9. The figure reports transaction throughput, transaction latency, load-scaling and saturation behavior, and CPU utilization across the workload stages.

The throughput results show that the Fabric-based implementation increases from 25 committed TPS at the initial workload stage to a maximum of approximately 221 committed TPS under the tested local configuration. Throughput increases almost linearly up to 200 TPS. Beyond this point, the network reaches a saturation region: increasing the offered load to 250 TPS, 300 TPS, and 350 TPS does not produce proportional throughput gains, and the committed throughput remains close to 220 TPS.

The transaction latency results follow the expected load-dependent behavior. As the offered load increases, average latency increases from 0.32 s at 25 TPS to 2.05 s at 200 TPS. After the saturation point, latency increases more sharply, reaching 4.10 s at 250 TPS, 6.80 s

at 300 TPS, and 9.60 s at 350 TPS. Tail latency follows the same trend: p99 latency increases from 0.70 s at H0 to 22.50 s at H7. This behavior is consistent with queueing effects in the endorsement, ordering, and commit pipeline under overload (Table 8).

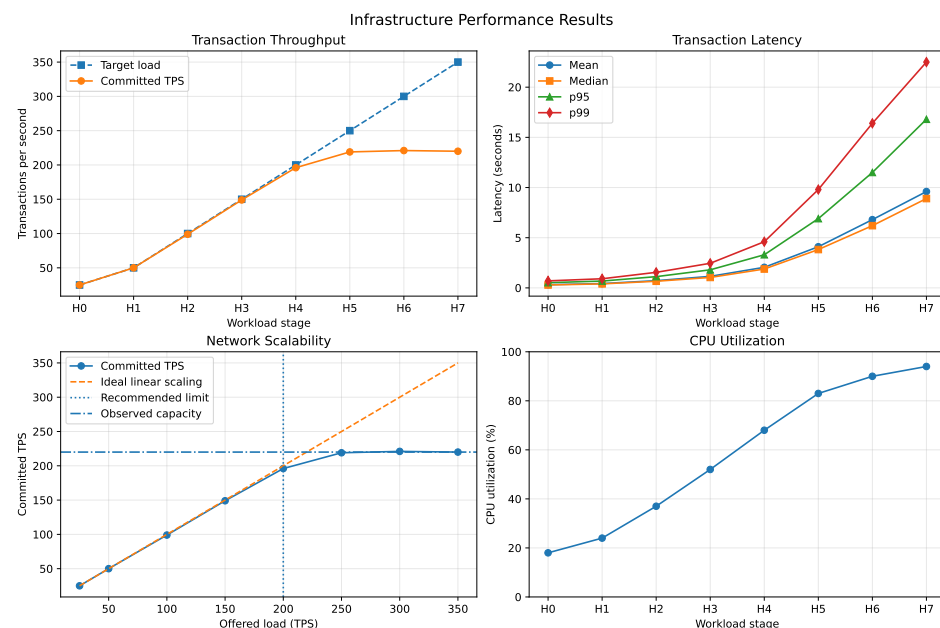


Figure 9. Infrastructure performance results under the controlled supply-chain workload.

Table 8. Performance results under increasing offered load.

Stage	Target Load	Committed TPS	Mean Latency	Median Latency	p95 Latency	p99 Latency	CPU	Success Rate
H0	25 TPS	25 TPS	0.32 s	0.29 s	0.52 s	0.70 s	18%	100.0%
H1	50 TPS	50 TPS	0.43 s	0.40 s	0.68 s	0.91 s	24%	100.0%
H2	100 TPS	99 TPS	0.72 s	0.66 s	1.12 s	1.55 s	37%	99.0%
H3	150 TPS	149 TPS	1.15 s	1.04 s	1.80 s	2.45 s	52%	99.3%
H4	200 TPS	196 TPS	2.05 s	1.88 s	3.30 s	4.60 s	68%	98.0%
H5	250 TPS	219 TPS	4.10 s	3.82 s	6.90 s	9.80 s	83%	87.6%
H6	300 TPS	221 TPS	6.80 s	6.20 s	11.50 s	16.40 s	90%	73.7%
H7	350 TPS	220 TPS	9.60 s	8.90 s	16.80 s	22.50 s	94%	62.9%

The load-scaling curve confirms that the disclosed fixed topology maintains high committed throughput and a success rate of at least 98% up to approximately 200 TPS. Beyond this threshold, the number of committed valid transactions stabilizes, while the success rate decreases sharply. This behavior indicates an observed saturation region for the tested single-machine configuration, not a general scalability limit of Fabric or of the 3T model.

At 25 TPS and 50 TPS, all submitted transactions are committed successfully. At 100–200 TPS, the success rate remains between 98.0% and 99.3%. The residual unsuccessful transactions may include endorsement response timeouts, deadline misses, or occasional MVCC invalidation; the experiment did not instrument a complete causal breakdown by failure category. The workload avoids intentional simultaneous writes to the same product key, so these stages should not be interpreted as a conflict-heavy benchmark. Under overload, the increasing latency and decreasing success rate are consistent with queueing and back-pressure, but the available data do not support assigning every unsuccessful transaction to a specific cause.

The resource-efficiency results show that CPU utilization increases from 18% to 68% as throughput increases from 25 TPS to 196 TPS. Under overload conditions, CPU utilization rises to more than 83%, while the success rate decreases to 87.6% at H5, 73.7% at H6,

and 62.9% at H7. These results indicate that the proposed infrastructure can support moderate-throughput enterprise traceability workloads under the tested local configuration, but higher fixed-topology capacity or broader deployment scale would require distributed deployment, optimized endorsement policies, more ordering resources, or further tuning of block and batching parameters.

5.5.7. Workload Composition and Saturation Analysis

Instead of relying on a direct quantitative comparison with public blockchain platforms, we analyze the behavior of the proposed system under its own supply chain workload. This is more appropriate because public and permissioned blockchains differ in governance, consensus mechanisms, privacy assumptions, finality models, and workload definitions. Therefore, cross-platform throughput and latency values should not be interpreted as directly comparable unless all platforms are evaluated under the same experimental protocol.

Figure 10 summarizes the workload composition and the saturation behavior of the proposed Fabric-based infrastructure. The workload is distributed across lifecycle operations, including tokenization, custody transfer, location updates, warehouse handling, and delivery confirmation. This composition reflects the operational structure of the case study, where each product generates several blockchain events during its journey from supplier to consumer.

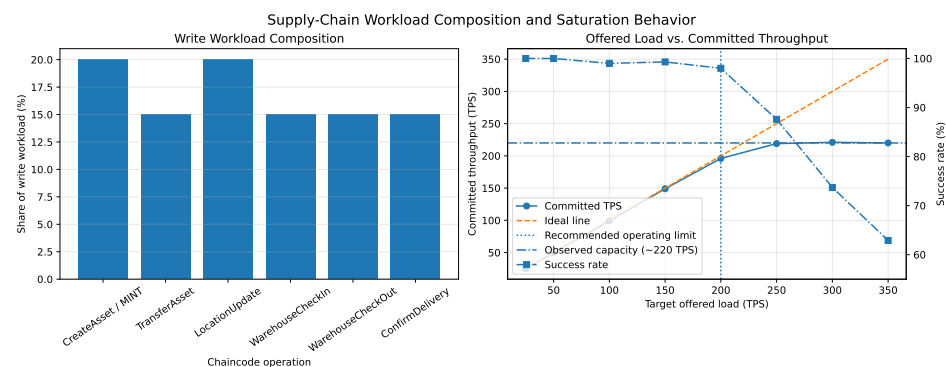


Figure 10. Supply-chain workload composition and saturation behavior.

The saturation curve shows that committed throughput follows the offered load up to approximately 200 TPS and then stabilizes around 220 TPS. After this threshold, additional submitted transactions mainly increase queueing, latency, and timeout probability rather than increasing committed throughput. This result is important for supply chain deployment because it identifies the safe operating region of the infrastructure. Under the tested configuration, the recommended operating region is at or below 200 TPS, where throughput remains high, latency remains controlled, and the success rate remains at least 98%.

5.5.8. Traceability Query Performance

Since the proposed architecture is based on the Tokenize–Track–Trace lifecycle, the performance of traceability queries is evaluated separately from write transactions. This separation is necessary because read-only queries do not pass through the full endorsement–ordering–commitment pipeline used by state-changing transactions. Instead, they access either the current product state or the historical sequence of committed events associated with a product.

The reported QueryHistory values are end-to-end application-level latencies. The prototype data model contains an embedded EventHistory array, and Fabric also retains committed product-key versions. The experiment did not separately instrument the inter-

nal retrieval path or the relative cost of these two representations. Consequently, the query results characterize the exposed application operation but should not be used to infer byte-level ledger growth or the latency of the Fabric key-history API alone (Table 9).

Table 9. Read-only traceability query performance.

Query Type	History Length	Mean Latency	Median Latency	p95 Latency	p99 Latency
<i>QueryAsset</i>	Current state only	0.08 s	0.07 s	0.13 s	0.18 s
<i>QueryHistory</i>	5 events	0.14 s	0.12 s	0.22 s	0.31 s
<i>QueryHistory</i>	20 events	0.31 s	0.29 s	0.48 s	0.65 s
<i>QueryHistory</i>	50 events	0.72 s	0.68 s	1.10 s	1.42 s

The query results show that current-state access through *QueryAsset* is faster than full history reconstruction, with a mean latency of 0.08 s. As expected, *QueryHistory* latency increases with the number of lifecycle events associated with a product. A short product history containing five events is retrieved with a mean latency of 0.14 s, while a longer history containing fifty events reaches a mean latency of 0.72 s and a p99 latency of 1.42 s. These results are consistent with the design of the 3T architecture: current-state lookup is suitable for operational monitoring, whereas history reconstruction supports auditability, provenance verification, route reconstruction, and recall analysis.

5.5.9. Security Assumptions and Limitations

Threat Model and Adversarial Capabilities

We consider a permissioned consortium in which a user, organization, application service, oracle validator, or device may deviate from the prescribed workflow while possessing a valid credential. An adversary may submit false or malformed payloads; invoke an event outside its assigned role or the product's current state; replay an earlier evidence reference; submit a stale or duplicate lifecycle command; withhold or equivocate on an oracle attestation; collude with other validators; or act through a compromised user, device, or validator key. A network adversary may delay or drop messages and cause availability failures, but is assumed unable to forge uncompromised signatures or break the employed hash functions.

The primary security goals are that every accepted lifecycle event is attributable to an authenticated identity, that unauthorized or state-inadmissible transitions do not update the world state, and that the committed history remains tamper-evident. The evaluated deployment assumes secure cryptographic primitives, uncompromised root certification authorities and required signing keys, deterministic chaincode, and correct operation of Fabric endorsement, ordering, and validation under the configured fault assumptions. The single state-owning actor reduces shared-state races inside the application service but is not a trust anchor and does not replace chaincode authorization, endorsement, or commit-time validation.

These controls preserve the integrity and attribution of accepted ledger records under the stated assumptions; they do not prove that an accepted physical observation is true. False acceptance remains possible if enough validators collude or are compromised to satisfy the configured threshold. Certification authority or ordering service compromise, denial of service, network partitioning, compromise or miscalibration at the point of sensing, and unavailability of an off-chain evidence repository are not experimentally addressed. Replay-resistant evidence binding is specified as a deployment extension and was not implemented in the reported prototype. The single-orderer case study also does not evaluate Byzantine ordering faults.

The proposed architecture improves traceability by combining permissioned identities, role-based authorization, oracle-anchored physical events, and immutable ledger history. However, blockchain does not eliminate the need for explicit trust assumptions, especially when physical-world events are mapped to digital records. In the proposed system, tokenization and selected lifecycle events depend on oracle attestations, while event updates depend on authenticated actors submitting accurate information.

The first assumption concerns the oracle model. The blockchain can verify that the required validator attestations were submitted and that the threshold rule was satisfied, but it cannot independently prove that the physical statement is true. Therefore, the validity of product tokenization depends on the independence and honesty of the validators. In the case study, two supplier-side validators are used for product validation. This configuration is sufficient to demonstrate feasibility, but industrial deployments should strengthen the model by including validators from different organizations, such as the supplier, warehouse, transporter, or an external auditor.

The second assumption concerns identity and authorization. Hyperledger Fabric's Membership Service Provider binds users to organizations and roles, while endorsement policies and chaincode logic restrict who can submit or approve specific lifecycle events. This protects the system against unauthorized updates, but it does not protect against compromised private keys. Production deployments should therefore include certificate revocation, key rotation, and monitoring of abnormal validator or actor behavior.

The third assumption concerns off-chain evidence. In the present prototype, *proofRef* is a string field that records a hash or document pointer; the available implementation does not establish automatic retrieval, independent signature verification, freshness checking, nonce enforcement, or replay-resistant event binding. A production implementation should anchor a cryptographic hash of the referenced evidence and bind it to the product identifier, event type, emitter, timestamp, and a nonce. These controls are proposed extensions rather than mechanisms evaluated in the reported experiment.

Operational Cost Boundary and Validator Governance

The present single-host experiment does not support a monetary cost estimate per product lifecycle. It measures aggregate CPU utilization and transaction performance, but it does not instrument transaction-level network bytes, replicated ledger growth, external evidence storage, device operation, validator labor, or cloud and administration charges. A deployment-specific estimate should decompose lifecycle cost as $C_{\text{life}} = C_{\text{compute}} + C_{\text{storage}} + C_{\text{network}} + C_{\text{evidence}} + C_{\text{oracle}}$. For a lifecycle containing m events, with a_j attestations, b_j on-chain bytes, and q_j off-chain evidence bytes for event j , the dominant verification work scales with $\sum_{j=1}^m a_j$, replicated ledger payload scales approximately with the number of committing replicas times $\sum_{j=1}^m b_j$, and external evidence storage scales with $\sum_{j=1}^m q_j$. Monetary values additionally depend on deployment topology, retention policy, service pricing, and organizational labor and therefore cannot be inferred from the reported benchmark.

The permissioned design does not assume a token reward for oracle participation. In an industrial consortium, honest participation should instead be supported by a governance charter, service-level obligations, shared operational benefits, auditable attribution, conflict-of-interest rules, and contractual penalties for demonstrable misconduct. Validator independence can be strengthened by selecting validators from different organizations, preventing one organization from satisfying the threshold alone, rotating or periodically re-authorizing validator roles, and involving an external auditor for high-impact events. A bond or stake mechanism could be added only where the consortium's legal and economic model supports objective slashing conditions. These governance and incentive

mechanisms are proposed deployment controls; the evaluated case study uses a static pair of supplier-side validators and does not test strategic or collusive behavior.

Finally, manual location updates remain a limitation. The system can authenticate the actor who submitted a location update and record the update immutably, but it cannot guarantee that the reported location corresponds to the physical position of the product. Section 4.4.4 specifies a prospective sensing and cryptographic-binding strategy; that integration was not implemented or evaluated, and the present results therefore make no claim about sensor reliability, sensing latency, or resistance to device compromise.

5.5.10. Transparent and Automated Supply Chain Using Smart Contracts

Our blockchain-based supply chain system, utilizing smart contracts, introduces transparency and automation from product tokenization until final delivery. We illustrate the workflow on ProductX. The lifecycle begins when ProductX, accepted under the configured tokenization-oracle policy, is tokenized as a dNFT through the `CreateAsset` chaincode function and prepared for dispatch.

Next, the `TransferAsset` function executes the custody transfer from the supplier to the transporter, committing the transaction on the blockchain to maintain a transparent custody record. ProductX then moves through the warehouse, where check-in and check-out events are registered via `RegisterEvent`. Upon reaching the consumer, the `ConfirmDelivery` event finalizes the custody chain and transfers responsibility to the consumer. Each phase is therefore documented on the blockchain through the chaincode interface introduced in Section 5, providing an immutable and transparent trail of ProductX's journey.

To show that the proposed mechanism is not limited to a single product instance, Figure 11 presents a ledger-reconstructed traceability graph involving multiple products. The figure is generated from application-level histories associated with committed lifecycle updates, rather than manually drawn from external records. ProductX is kept as the main completed delivery scenario. ProductY illustrates a product that has reached the warehouse but has not yet been delivered. ProductZ illustrates a recall or return case, where the product is traced backward from the consumer to the supplier. ProductW illustrates an alternative route in which the product is sent directly from the supplier to the warehouse before final delivery.

To generate the graph in Figure 11, the application-level history operation is invoked using the product identifier. The trace query retrieves the sequence of committed events, verifies their ledger attribution, and evaluates the configured condition associated with *proofRef* when an evidence reference is required. The resulting event sequence is then projected into a product-lane trace matrix. Actor columns correspond to supply chain roles, while event markers and directed edges represent committed lifecycle events such as custody transfer, warehouse check-in, delivery confirmation, or recall return.

The distinction between custody-transfer events and tracking-only events is important. Custody-transfer events change the actor responsible for the product, whereas location updates provide additional tracking information while the product remains under the responsibility of the current actor. In Figure 11, ProductX includes a tracking-only location update submitted by the transporter during transit. This event is represented separately from the main custody path, which avoids conflating physical movement, responsibility transfer, and final ownership transfer.

The current `LocationUpdate` remains a manually submitted declaration by the product's custodian. The ledger can authenticate and preserve that declaration but cannot independently establish the product's physical position. To avoid duplicating the security discussion, the proposed sensor types, gateway protocols, signed evidence envelope, re-

play controls, and remaining device trust limitations are specified in Section 4.4.4. Their implementation and experimental validation remain future work.

Overall, the experiment demonstrates the feasibility of the disclosed Hyperledger Fabric workflow and identifies its configuration-specific operating region. To avoid repeating the same deployment qualifications in the Results, the implications, limitations, and future validation program are consolidated in Sections 6 and 7.

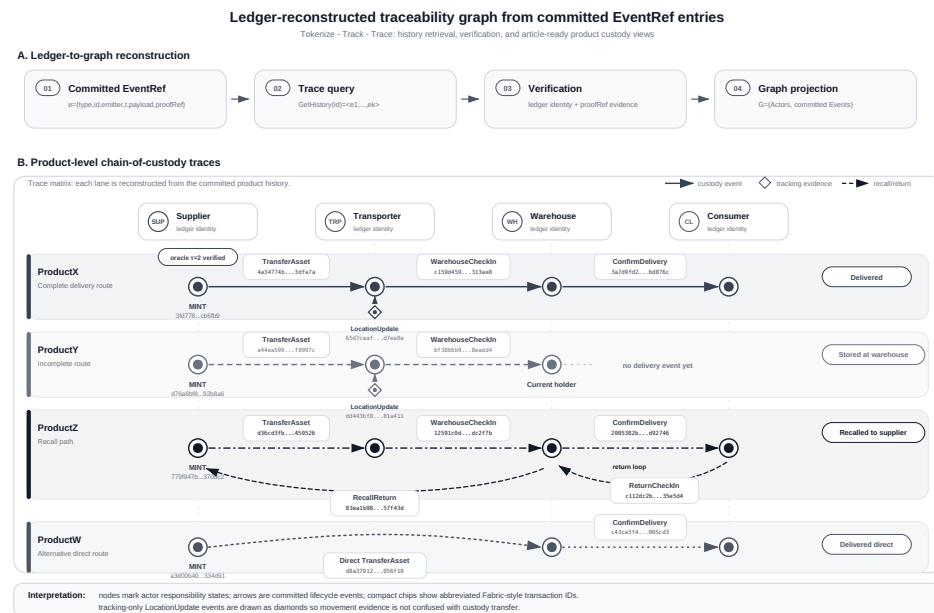


Figure 11. Ledger-reconstructed traceability graph for multiple products. Panel (A) summarizes the reconstruction pipeline from committed *EventRef* entries to an auditable graph through history retrieval, ledger-identity verification, and optional *proofRef* policy checking. Panel (B) presents product-level chain-of-custody traces in a grayscale trace-matrix view. Actor columns represent the supply chain roles involved in custody or observation, while product lanes represent reconstructed histories for ProductX, ProductY, ProductZ, and ProductW. Circular nodes indicate custody or state-changing events, diamond nodes indicate tracking-only location updates, and directed edges represent committed lifecycle transitions labeled with abbreviated transaction identifiers. ProductX shows a completed delivery route, ProductY an incomplete route currently stored at the warehouse, ProductZ a recall/return path, and ProductW an alternative direct warehouse route.

6. Discussion

6.1. Interpretation of the Findings

The experiment demonstrates that the complete 3T workflow can be executed and audited on the disclosed Fabric configuration, and it identifies a configuration-specific transition from near-linear throughput growth to queueing-dominated saturation. The approximately 220-TPS plateau is therefore evidence about this single-host topology and workload, not a universal capacity of Fabric or of the 3T model. Likewise, the increase in history-query latency with the number of events is consistent with increasing history-processing work and supports the distinction between fast current-state access and more expensive audit reconstruction; the benchmark did not isolate the internal history retrieval path or its storage contribution.

6.2. Scientific and Industrial Implications

For research, the principal implication is that physical admission, actor authorization, business-state admissibility, evidence validation, and trace reconstruction can be expressed as one checkable lifecycle specification rather than as loosely connected implementation

features. For industrial deployment, the proposed architecture is particularly relevant for multi-actor supply chains where traceability, provenance, and controlled data sharing are critical, such as manufacturing, food, pharmaceutical, and logistics sectors. However, practical adoption requires several organizational and technological prerequisites, including reliable IoT/oracle infrastructures, digital identity management, appropriate governance models among supply chain partners, and integration mechanisms with existing enterprise information systems such as ERP platforms. In addition, challenges related to implementation costs, organizational readiness, interoperability with legacy systems, and stakeholder coordination remain important considerations for large-scale adoption. Ledger integrity must therefore be combined with independent oracle governance, key lifecycle management, selective disclosure, enterprise-system reconciliation, and operational monitoring. Sections 4.4.4, 4.5, 5.3 and 5.5.9 describe technically credible strategies for these concerns while distinguishing them from features actually evaluated in the prototype.

6.3. Research Limitations

The reported experiments were conducted in a Docker-based, single-host laboratory environment. They do not reproduce a production-scale distributed cluster, wide-area delay, network partitioning, geographically separated organizations, or the failure behavior of a multi-node ordering service. The measured saturation point and latency values are specific to the disclosed hardware and workload and should be interpreted as feasibility results rather than as universal deployment limits.

The evaluated network contains four organizations, eight peers, one shared application channel, and one ordering node. Larger consortium sizes, multiple privacy domains, heterogeneous endorsement policies, cross-channel or cross-chain workflows, and sustained same-product contention were not evaluated. No matched baseline or feature ablation was performed, so the experiment does not isolate the performance overhead attributable to authorization, evidence verification, or history handling and does not establish superiority over competing traceability systems.

The case-study oracle policy uses a static supplier-side validator set and a two-of-two tokenization threshold. It does not experimentally assess validator collusion, dynamic admission and revocation, strategic incentives, or governance disputes. Field-level selective disclosure, automated IoT evidence, production enterprise connectors, and monetary lifecycle cost were specified only as deployment strategies or future work. The current proofRef field records a hash or document pointer, but external evidence retrieval, signature verification, freshness, nonce enforcement, and replay-resistant binding were not implemented. Likewise, $\text{Pre}(e, x)$ is a specification-level conformance requirement; the available implementation description and benchmark do not document or test every state-transition guard.

The application layer uses one writer actor: this removes in-process shared-state races but can introduce a submission bottleneck and availability dependency, while product-key sharding and durable mailbox recovery were not evaluated. The benchmark did not record the number of already submitted transactions concurrently resident in the Fabric pipeline. The data model also exposes an embedded EventHistory while Fabric retains committed key versions; the QueryHistory experiment did not isolate the retrieval path or byte-level storage growth. Finally, increasing roles, event types, evidence rules, state preconditions, and endorsement combinations can increase verification, serialization, policy-maintenance, and governance overhead; the current benchmark does not isolate these marginal costs. These limitations delimit the claims of the study.

7. Conclusions

This paper contributes an architecture-level specification for coupling physical-product admission, dynamic non-fungible state, typed authorized transitions, event-specific evidence, and auditable trace reconstruction within the Tokenize–Track–Trace lifecycle. The contribution is not a new ledger, token standard, or oracle primitive; it is the explicit integration of these mechanisms and their mapping to an executable permissioned-ledger implementation.

The Hyperledger Fabric case study demonstrates feasibility and characterizes saturation, latency, resource use, success rate, and history-query behavior under one controlled single-host configuration. These results support the internal validity of the implementation but do not establish comparative superiority, production-scale resilience, or quantified environmental impact. The implications and claim boundaries are discussed in Section 6. The proposed 3T architecture demonstrates how product tokenization, event-driven smart contracts, oracle validation, and ledger-based history reconstruction can be combined to support end-to-end traceability. The Hyperledger Fabric implementation shows that the system sustains moderate-throughput traceability workloads (approximately 220 committed TPS with sub-second to low-second latencies in the recommended operating region) under the tested single-host configuration, while the traceability-query evaluation confirms that product histories can be reconstructed from committed ledger events. These results support the feasibility of using dynamic NFTs as evolving digital representations of physical products throughout their lifecycle.

Looking forward, four research directions emerge from the identified limitations. First, the integration of IoT devices would automate data collection, particularly for real-time tracking and verification of products, reducing the current reliance on manual location declarations. Second, the security and governance model should be strengthened through multi-organization oracle validators, certificate revocation, key-rotation policies, replay-protection mechanisms, and cryptographic hashing of off-chain evidence referenced by *proofRef*. Third, deployment scaling beyond the fixed-topology saturation region observed in our experiments calls for distributed multi-host deployment, multi-node ordering services, optimized endorsement policies, and the evaluation of newer ordering options such as the BFT consensus introduced in Hyperledger Fabric 3.x. Finally, interoperability across supply chain networks could be pursued through standardized protocols or frameworks for blockchain implementation in SCM, enhancing information sharing and collaboration across organizations. Further evaluation should include conflict-heavy same-product workloads, per-lifecycle storage and bandwidth instrumentation, field-level confidentiality mechanisms, and an EVM proof-of-concept evaluated under its platform-specific cost, privacy, and finality assumptions. To make these directions more testable, future work should: (i) implement the signed IoT evidence strategy with device identity, freshness checks, and calibration or compromise monitoring; (ii) investigate AI-assisted anomaly detection only as an advisory mechanism evaluated for explainability, bias, adversarial manipulation, and false decisions; (iii) test multi-organization oracle selection, revocation, incentives, dispute resolution, and collusion; (iv) conduct geographically distributed, conflict-heavy, and matched-baseline experiments followed by an industrial pilot; and (v) evaluate cross-chain and enterprise interoperability through standardized event schemas, identity federation, privacy-preserving data exchange, and an EVM proof of concept. These refinements extend the original directions without changing the present claim boundary: the reported results apply only to the mechanisms and experimental configuration actually evaluated.

As blockchain continues to mature, its role in SCM will depend on combining ledger integrity with reliable physical-world data, clear governance rules, and interoperable

enterprise systems. The 3T architecture presented in this paper constitute a step in that direction, providing a foundation on which these extensions can be built.

Author Contributions: Conceptualization, Y.I.M., T.C., A.B., and D.T.; methodology, Y.I.M., T.C., A.B., and D.T.; software, Y.I.M.; validation, T.C., D.T., and A.B.; writing—original draft preparation, Y.I.M.; writing—review and editing, T.C., A.B., and D.T.; supervision, A.B., D.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the French National Research Agency (ANR) under the project ANR-25-CE10-5362.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Acknowledgments: An AI tool was used solely for language editing, including sentence rephrasing and spelling correction.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Prasanna, R.; Jayasundara, J.; Naradda Gamage, S.K.; Ekanayake, E.; Rajapakshe, P.; Abeyrathne, G. Sustainability of SMEs in the competition: A systemic review on technological challenges and SME performance. *J. Open Innov.* **2019**, *5*, 100.
2. Frizzo-Barker, J.; Chow-White, P.A.; Adams, P.R.; Mentanko, J.; Ha, D.; Green, S. Blockchain as a disruptive technology for business: A systematic review. *Int. J. Inf. Manag.* **2020**, *51*, 102029.
3. Bekrar, A.; El Cadi, A.A.; Todosijevic, R.; Sarkis, J. Digitalizing the closing-of-the-loop for supply chains: A transportation and Blockchain perspective. *Sustainability* **2021**, *13*, 2895.
4. Treiblmaier, H.; Beck, R. *Business Transformation Through Blockchain*; Springer: Berlin/Heidelberg, Germany, 2019; Volume 1.
5. Rakshit, S.; Islam, N.; Mondal, S.; Paul, T. Influence of blockchain technology in SME internationalization: Evidence from high-tech SMEs in India. *Technovation* **2022**, *115*, 102518.
6. Mendez Royo, P.; Rodriguez-Molina, J.; Garbajosa, J.; Castillojo, P. Towards Blockchain-Based Internet of Things Systems for Energy Smart Contracts with Constrained Hardware Devices and Cloud Infrastructure. *IEEE Access* **2021**, *9*, 77742–77757. <https://doi.org/10.1109/ACCESS.2021.3081932>.
7. Nakamoto, S. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Bus. Rev.* **2008**, *4*, 21260.
8. Treiblmaier, H. The impact of the blockchain on the supply chain: A theory-based research framework and a call for action. *Supply Chain. Manag. Int. J.* **2018**, *23*, 545–559.
9. Hasselgren, A.; Kravetska, K.; Gligoroski, D.; Pedersen, S.A.; Faxvaag, A. Blockchain in healthcare and health sciences—A scoping review. *Int. J. Med Inform.* **2020**, *134*, 104040.
10. Yuan, Y.; Wang, F.Y. Towards blockchain-based intelligent transportation systems. In *Proceedings of the 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*; IEEE: New York, NY, USA, 2016; pp. 2663–2668.
11. Andoni, M.; Robu, V.; Flynn, D.; Abram, S.; Geach, D.; Jenkins, D.; McCallum, P.; Peacock, A. Blockchain technology in the energy sector: A systematic review of challenges and opportunities. *Renew. Sustain. Energy Rev.* **2019**, *100*, 143–174.
12. Jeon, H.j.; Youn, H.c.; Ko, S.m.; Kim, T.h. Blockchain and AI Meet in the Metaverse. In *Advances in the Convergence of Blockchain and Artificial Intelligence*; IntechOpen: London, UK, 2022; p. 73.
13. Davydov, V.A.; Kruglik, S.A.; Yanovich, Y.A. Comparison of Banking and Peer-to-Peer Lending Risks. *Autom. Remote Control* **2021**, *82*, 2155–2168. <https://doi.org/10.1134/S0005117921120079>.
14. Schär, F. Decentralized Finance: On Blockchain- and Smart Contract-Based Financial Markets. *Fed. Reserve Bank St. Louis Rev.* **2021**, *103*, 153–174. <https://doi.org/10.20955/r.103.153-74>.
15. Fraga-Lamas, P.; Fernández-Caramés, T.M.; Rosado da Cruz, A.M.; Lopes, S.I. An Overview of Blockchain for Industry 5.0: Towards Human-Centric, Sustainable and Resilient Applications. *IEEE Access* **2024**, *12*, 116162–116201. <https://doi.org/10.1109/ACCESS.2024.3435374>.
16. Chargui, T.; Mahjoub, Y.I.; Bekrar, A.; Trentesaux, D. Integration of Blockchain and the Physical Internet: A Review and Future Prospects. In *Proceedings of the International Workshop on Service Orientation in Holonic and Multi-Agent Manufacturing*; Springer: Berlin/Heidelberg, Germany, 2023; pp. 583–592.

17. Panwar, A.; Bhatnagar, V. Distributed ledger technology (DLT): The beginning of a technological revolution for blockchain. In *Proceedings of the 2nd International Conference on Data, Engineering and Applications (IDEA)*; IEEE: New York, NY, USA, 2020; pp. 1–5.
18. Ma, J. The Impact of Technological Changes on Small and Medium Enterprises in Vietnamese Organic Food Industry. In *Proceedings of the 2022 7th International Conference on Financial Innovation and Economic Development (ICFIED 2022)*; Atlantis Press: Dordrecht, The Netherlands, 2022; pp. 411–416.
19. Mahjoub, Y.I.; Hassoun, M.; Trentesaux, D. Blockchain adoption for SMEs: Opportunities and challenges. *IFAC-PapersOnLine* **2022**, *55*, 1834–1839.
20. Mingxiao, D.; Xiaofeng, M.; Zhe, Z.; Xiangwei, W.; Qijun, C. A review on consensus algorithm of blockchain. In *Proceedings of the 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*; IEEE: New York, NY, USA, 2017; pp. 2567–2572.
21. Androulaki, E.; Barger, A.; Bortnikov, V.; Cachin, C.; Christidis, K.; De Caro, A.; Enyeart, D.; Ferris, C.; Laventman, G.; Manevich, Y.; et al. Hyperledger fabric: A distributed operating system for permissioned blockchains. In *Proceedings of the Thirteenth EuroSys Conference*, Porto, Portugal, 23–26 April 2018; pp. 1–15.
22. Ilbiz, E.; Durst, S. The appropriation of blockchain for small and medium-sized enterprises. *J. Innov. Manag.* **2019**, *7*, 26–45.
23. Szabo, N. Formalizing and securing relationships on public networks. *First Monday* **1997**, *2*. <https://doi.org/10.5210/fm.v2i9.548>.
24. Kamilaris, A.; Fonts, A.; Prenafeta-Boldú, F.X. The rise of blockchain technology in agriculture and food supply chains. *Trends Food Sci. Technol.* **2019**, *91*, 640–652.
25. Li, K.; Lee, J.Y.; Gharehgozli, A. Blockchain in food supply chains: A literature review and synthesis analysis of platforms, benefits and challenges. *Int. J. Prod. Res.* **2021**, *61*, 3527–3546.
26. Dutta, S.; Saini, K. Statistical assessment of hybrid blockchain for SME sector. *WSEAS Trans. Syst. Control* **2021**, *16*, 83–95.
27. Sassi, H.; Trentesaux, D.; Mahjoub, Y.I. The Added Value of Blockchain in Logistics: A Comparative Study with Conventional Systems using simulation. In *Proceedings of the 52nd International Conference on Computers and Industrial Engineering (CIE52)*, Lyon, France, 29–31 October 2025.
28. Agrawal, S.; Kumar, D.; Singh, R.K.; Singh, R.K. Coordination issues in managing the reverse supply chain: A systematic literature review and future research directions. *Benchmarking Int. J.* **2023**, *30*, 1259–1299.
29. Caro, M.P.; Ali, M.S.; Vecchio, M.; Gialfreda, R. Blockchain-based traceability in Agri-Food supply chain management: A practical implementation. In *Proceedings of the 2018 IoT Vertical and Topical Summit on Agriculture-Tuscany (IOT Tuscany)*; IEEE: New York, NY, USA, 2018; pp. 1–4.
30. Mahjoub, Y.I.; Chargui, T.; Bekrar, A.; Trentesaux, D. Supply chain Application of Blockchain-based solutions for Cyber-Physical Systems: Review and prospects. In *Proceedings of the 11th Workshop on Service Oriented, Holonic and Multi-Agent Manufacturing Systems*, Cluny, France, 18–19 November 2021.
31. Javaid, M.; Haleem, A.; Pratap Singh, R.; Khan, S.; Suman, R. Blockchain technology applications for Industry 4.0: A literature-based review. *Blockchain Res. Appl.* **2021**, *2*, 100027.
32. Bamakan, S.M.H.; Nezhadsistani, N.; Bodaghi, O.; Qu, Q. A Decentralized Framework for Patents and Intellectual Property as NFT in Blockchain Networks. *Preprint* **2021**. <https://doi.org/10.21203/rs.3.rs-951089/v1>.
33. Ethereum.org. Introduction to Smart Contracts. 2026. Available online: <https://ethereum.org/developers/docs/smart-contracts/> (accessed on 10 June 2026).
34. Polygon Labs. Polygon Chain Overview. 2026. Available online: <https://docs.polygon.technology/pos/overview> (accessed on 10 June 2026).
35. Avalanche. Avalanche L1s. 2026. Available online: <https://build.avax.network/docs/avalanche-l1s> (accessed on 10 June 2026).
36. Solana Foundation. Solana Documentation. 2026. Available online: <https://solana.com/docs> (accessed on 10 June 2026).
37. Hyperledger Fabric. Private Data. 2026. Available online: <https://hyperledger-fabric.readthedocs.io/en/release-2.5/private-data/private-data.html> (accessed on 10 June 2026).
38. Hyperledger Besu. Besu for Private and Permissioned Networks. 2026. Available online: <https://besu.hyperledger.org/private-networks> (accessed on 10 June 2026).
39. ConsenSys. GoQuorum: Public and Private Transactions. 2023. Available online: <https://docs.goquorum.consensys.io/concepts/privacy/private-and-public> (accessed on 10 June 2026).
40. R3. Getting Started with Corda. 2021. Available online: <https://r3.com/wp-content/uploads/2024/10/getting-started-with-corda-081021-1.pdf> (accessed on 10 June 2026).
41. Hedera. Hedera Smart Contract Service. 2026. Available online: <https://hedera.com/service/smart-contract-service/> (accessed on 10 June 2026).
42. IOTA Foundation. IOTA Smart Contracts. 2026. Available online: <https://docs.iota.org/developer/iota-evm/> (accessed on 10 June 2026).
43. VeChain. Built-in Contracts: Authority Contract. 2026. Available online: <https://docs.vechain.org/developer-resources/built-in-contracts> (accessed on 10 June 2026).

44. MultiChain. Introducing MultiChain Streams. 2016. Available online: <https://www.multichain.com/blog/2016/09/introducing-multichain-streams/> (accessed on 10 June 2026).
45. IBM. How to Use IBM App Connect with IBM Food Trust. 2026. Available online: <https://www.ibm.com/docs/en/app-connect/12.0.x?topic=hga-food-trust> (accessed on 10 June 2026).
46. Cole, R.; Stevenson, M.; Aitken, J. Blockchain technology: Implications for operations and supply chain management. *Supply Chain Manag. Int. J.* **2019**, *24*, 469–483.
47. Gurtu, A.; Johnny, J. Potential of blockchain technology in supply chain management: A literature review. *Int. J. Phys. Distrib. Logist. Manag.* **2019**, *49*, 881–900.
48. Queiroz, M.M.; Telles, R.; Bonilla, S.H. Blockchain and supply chain management integration: A systematic review of the literature. *Supply Chain Manag. Int. J.* **2020**, *25*, 241–254.
49. Park, A.; Li, H. The effect of blockchain technology on supply chain sustainability performances. *Sustainability* **2021**, *13*, 1726.
50. Peng, Y.; Chen, X.; Wang, X. Enhancing supply chain flows through blockchain: A comprehensive literature review. *Int. J. Prod. Res.* **2023**, *61*, 4503–4524.
51. Daghighi, A.; Shoushtari, F. Toward Sustainability of Supply Chain by Applying Blockchain Technology. *Int. J. Ind. Eng. Oper. Res.* **2023**, *5*, 60–72.
52. Azizi, N.; Malekzadeh, H.; Akhavan, P.; Haass, O.; Saremi, S.; Mirjalili, S. IoT-blockchain: Harnessing the power of internet of thing and blockchain for smart supply chain. *Sensors* **2021**, *21*, 6048.
53. Henry, T.; Beck, R.; Laga, N.; Gaaloul, W.; Pan, S. Decentralized procurement mechanisms for efficient logistics services mapping-a design science research approach. In *Proceedings of the Annual Hawaii International Conference on System Sciences*; IEEE Computer Society Press: New York, NY, USA, 2022.
54. Musamih, A.; Yaqoob, I.; Salah, K.; Jayaraman, R.; Omar, M.; Ellahham, S. Using NFTs for Product Management, Digital Certification, Trading, and Delivery in the Healthcare Supply Chain. *IEEE Trans. Eng. Manag.* **2022**, *71*, 4480–4501.
55. Turki, M.; Cheikhrouhou, S.; Dammak, B.; Baklouti, M.; Mars, R.; Dhahbi, A. NFT-IoT Pharma Chain: IoT Drug traceability system based on Blockchain and Non Fungible Tokens (NFTs). *J. King Saud Univ.-Comput. Inf. Sci.* **2023**, *35*, 527–543.
56. Gebreab, S.A.; Hasan, H.R.; Salah, K.; Jayaraman, R. NFT-based traceability and ownership management of medical devices. *IEEE Access* **2022**, *10*, 126394–126411.
57. Huong, L.H.; Tran, B.Q.; Doan, H.M.; Trong, N.D.; Le, H.V.; Phu, L.V.; Quoc, D.N.; Le, K.T.; Tran, N.H.; The, A.N.; et al. Towards a Cash-on-Delivery Management Solution Based on Blockchain, Smart Contracts, and NFT. In *Proceedings of the International Conference on Database Systems for Advanced Applications*; Springer: Berlin/Heidelberg, Germany, 2023; pp. 51–66.
58. Marchese, A.; Tomarchio, O. An Agri-Food Supply Chain Traceability Management System based on Hyperledger Fabric Blockchain. In *Proceedings of the 23rd International Conference on Enterprise Information Systems (ICEIS)*; SciTePress: Setúbal, Portugal, 2021; Volume 2, pp. 648–658. <https://doi.org/10.5220/0010447606480658>.
59. Esmaeilian, B.; Sarkis, J.; Lewis, K.; Behdad, S. Blockchain for the future of sustainable supply chain management in Industry 4.0. *Resour. Conserv. Recycl.* **2020**, *163*, 105064.
60. Bellavista, P.; Esposito, C.; Foschini, L.; Giannelli, C.; Mazzocca, N.; Montanari, R. Interoperable blockchains for highly-integrated supply chains in collaborative manufacturing. *Sensors* **2021**, *21*, 4955.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.