Blockchain: Research and Applications 3 (2022) 100072

Contents lists available at ScienceDirect

Blockchain: Research and Applications

journal homepage: www.journals.elsevier.com/blockchain-research-and-applications

Privacy preserving transparent supply chain management through Hyperledger Fabric

Deebthik Ravi [a], Sashank Ramachandran [a], Raahul Vignesh [a], Vinod Ramesh Falmari [b], M. Brindha [b,*]

[a] Department of Electrical and Electronics Engineering, National Institute of Technology, Tiruchirappalli, 620015, Tamil Nadu, India
[b] Department of Computer Science and Engineering, National Institute of Technology, Tiruchirappalli, 620015, Tamil Nadu, India

ARTICLE INFO

Keywords:
Permissioned blockchain
Supply chain
Hyperledger Fabric
Provenance
Privacy protection
Security
Modularity

ABSTRACT

The revolution of blockchain technology started in the form of a cryptocurrency called Bitcoin. In recent years, this decentralized technology has attained worldwide adoption and growth in multiple sectors, including e-governance, e-commerce, and asset management. Having disrupted these sectors, the choice of blockchain technology to solve real-world problems concerning supply chain seems to be an innovative strategy. In this paper, an attempt is made to analyze blockchain's ability to improve the standards of supply chain management. This paper includes a methodology to implement the suggested idea using a permissioned blockchain platform—Hyperledger Fabric. For the paper's scope, the existing supply chain problems, such as data integrity, provenance transparency, privacy, and security, are given emphasis more specifically in the context of the coffee supply chain industry, while also concurrently attempting to generalize the solution to manage other supply chain activities effectively. Thus, the final objective of this research is to identify whether permissioned blockchain platforms could help stakeholders in the supply chain industry engage in a less-corruptible alternative to traditional web technology and whether they could enable a more positively nuanced blockchain system that draws the best balance between traditional web technology and a public blockchain, including privacy protection and security.

1. Introduction

The advent of the Internet has positively impacted humanity in multiple ways, particularly by providing a platform for virtual environments. Today, every industry is actively attempting to move its operations virtually over web applications. Though the Internet and web applications have achieved a notable reputation, they also come with a dark side of being a rising hotbed of criminal activities, including online fraud and identity theft. Therefore, researchers have been actively involved in transforming the Internet and web applications into a more secure and credible source. In 2009, Satoshi Nakamoto introduced the concept of blockchain technology to the world through the breakthrough invention of Bitcoin [1]. Initially, limited to the financial domain, people gradually began realizing the importance of this ingenious peer-to-peer design architecture. The world believed in blockchain's potential, as it emerged as a promising technology to guarantee trust among the participating entities without intermediary involvement. Blockchain could build a secure communication model, wherein immutability and data integrity can be assured. Since then, this novel technology has been a forerunner in the race to solve the existing Internet problems and looks like a promising tool to guarantee the integrity and security of stored data compared to many other traditional technologies [2]. Gartner, a leading research and advisory company, has estimated that blockchain technology will potentially lead to an industry that would be worth more than US $176 billion of business value by 2025, and it will increase to US $3.1 trillion by 2030. One major sector that will be highly influenced by this ground-breaking technology is the supply chain and logistics industry.

The supply chain and logistics industry have been an integral part of most businesses and enterprises worldwide. The global supply chain market size was valued at US $15.85 billion in 2019 and is projected to reach US $37.41 billion by 2027, growing at a CAGR of 11.2% from 2020 to 2027. With such significance to the world economy, the supply chain is

* Corresponding author.
E-mail address: brindham@nitt.edu (M. Brindha).

https://doi.org/10.1016/j.bcra.2022.100072
Received 3 March 2021; Received in revised form 9 February 2022; Accepted 6 March 2022

highly vital to every end-consumer. Any supply chain's primary objective will be to make the products readily available to meet customer demand—and that includes delivery to the appropriate location, on time, and in adequate quantities.

In recent years, the supply chain industries have been forced to make their processes more transparent to end customers. Consumers not only demand the product's availability at the right time and location but also expect the product to be authentic. Every company ultimately aims to build the trust and satisfaction of its consumers. On carefully investing in the current supply chains, one can understand that the lack of information during multiple stages compromises the product's trust. One of the primary goals of supply chain management should be to improve the transparency, traceability, and auditability of materials flowing throughout the chain [3]. Therefore, the complete data relating to a product from its production until delivery should be made more accessible to consumers.

The contemporary system in place that incorporates a traditional non-blockchain system has its fair share of problems that are not viable in the current era of technology and development. The problems can be briefly categorized as follows:

- Semantics: Non-transparency, monopoly, and asymmetry
- Human Intervention: Susceptibility to tampering, falsification of data, fraud, bias, and corruption
- Underlying Technology: Limited scalability and single point of failure

The problems listed above mainly arise because of significant reliance on a centralized system, therefore, naturally, the solution here would be to shift the paradigm to a decentralized system that should comprise the following inherent features:

- Distributed system and distributed ledger technology (DLT)
- Transparency, immutability, and authenticity
- Cryptographic primitives as the underlying logic

In this paper, an attempt has been made to solve the opaque and complex nature of the supply chain using blockchain technology. By employing this blockchain-based ledger, a distributed form of the database [4], to record such essential information, the logistics industry's overall trust and authenticity can be enhanced. It also facilitates the solution of the problems of provenance tracking, centralized systems, immutability, authenticity, scalability, reliability, etc. Since the solution of implementing a typical public blockchain may result in an exaggerated and negatively excess solution, another solution is proposed that would serve as an extension of an ideal blockchain system. This paper thus aims to implement one such solution using a permissioned blockchain-based platform—Hyperledger Fabric, and draw a comparison of how it might mitigate the problems that might have arisen because of a full-scale public blockchain system, such as lack of confidentiality, need for private data sharing, lack of modularity, etc., and simultaneously accentuate the advantages of a typical public blockchain system by going a step further in giving more flexibility for enhancing privacy and security. For the scope of this paper, a real-world example of the coffee supply chain is used to explain the idea in detail. However, this does not mean that blockchain technology cannot be scaled to manage other supply chains. Indeed, the paper concludes by asserting that this platform could be modified and tailor-made to suit any real-world supply chain industry.

The contributions of this paper are:

- A novel permissioned blockchain-based digital supply chain management system is formulated and developed using Hyperledger Fabric.
- The channel system of Hyperledger Fabric allows the network to be made completely transparent to the defined stakeholders while also blocking access to any entity outside the list of authorized systems. Thus, the issue of non-transparency in the traditional (non-blockchain) system, as well as the issue of complete openness of a public blockchain system, are both solved in this case.
- Factors such as performance, scalability, and privacy are elevated by the usage of Hyperledger Fabric through its permissioned mode of operation and ability to restrict transparency when needed, to enable privacy and confidentiality.
- A pluggable and modular consensus framework is structured and developed using Hyperledger Fabric's inherent and extensive features.
- Privacy protection, anonymity, and blockchain security are handled with Hyperledger Fabric's underlying cryptographic services and digital certificate management system.
- The performance of the proposed system is analyzed through Hyperledger Caliper using metrics: transactions per second (TPS), transaction latency, and transaction throughput.

2. Literature review

In 2009, the first ever digital currency by the name of Bitcoin was introduced through a whitepaper that was published at the time by Satoshi Nakamoto, a pseudonym for the anonymous developer(s) behind this invention. Bitcoin successfully solved the double-spending problem without any involvement from intermediaries [1]. The decentralized architecture of Bitcoin using blockchain technology and the proof-of-work consensus algorithm increased the transparency, trust, and hence the verifiability of a transaction. Thus, gradually, the terms Bitcoin and blockchain gained respect and influence in the technological society. Bitcoin is a legendary technology revolution that redefined the way one thinks about the banking and financial sector. While this cryptocurrency looked promising initially, it also had its flaws sourced from its own script. In late 2014, Vitalik Buterin released a white paper that addressed the issues of Bitcoin, and as a result, he subsequently designed and developed the then novel concept of smart contracts [5]. The introduction of smart contracts and the existing benefits of blockchain technology drew business stakeholders' eyes to employ this new technology in their enterprise solutions. This interest in building market solutions using Blockchain led to the introduction of several platforms, including the recent version of Hyperledger Fabric [6], which gained momentum better than other competitors, which had a few architectural limitations [7]. Thus, the design architecture and the core protocols of Hyperledger Fabric satisfied the needs of enterprise-level solutions, particularly in the supply chain and logistics industry. Recent advancements in technology have constantly been driving innovation in supply chain management to reach greater heights. Several new technologies, such as the internet-of-things (IoT), machine learning, cybersecurity, etc., have transformed traditional supply chains into smart supply chains. In the slipstream of these developments, blockchain technology and its implications for supply chain management have lately been receiving increasing attention. This ledger-based technology is considered a means of improving traceability and transparency during logistics and other supply chain processes.

As in supply chain management, discussed in Section 1, fraud may happen during production, processing, packing, and transportation. Hence, the problem of maintaining food security is reduced by the transparency and traceability of each activity. Here, some literature regarding transparency, traceability, and supply chain systems is focused. Jing et al. used the neural network backpropagation methodology to focus on the traceability issues in agriculture [8]. However, to train the

network, large amounts of test data are required. Further, radio frequency identification (RFID) technology is applied in supply chain management. RFID is a non-contact automatic identification communication system [9]. Using RFID, supply chain systems are designed in Refs. [10,11]. But the cost is higher because the RFID system needs a UHF RFID reader-printer, software to produce RFID tags, a thermal transfer ribbon, etc. Next, blockchain technology is applied in agriculture for transparency, traceability, and supply chain management in Refs. [8,9,12–21]. Salah et al. [13] proposed a solution using Ethereum, smart contracts, and IPFS (InterPlanetary File System) for soybean traceability in the supply chain, and it is naturally flawed in the sense that it lacks privacy, identity management, and scalability, as mentioned by the authors themselves in their conclusion. Organic food supply chain using Ethereum and smart contracts was developed in Ref. [17]. Hyperledger Sawtooth and smart contracts are used in Refs. [18,20]. IoT, along with blockchain, is used in food traceability by Tsang et al. [19]. Lin et al. [15] designed a food safety and traceability system using blockchain and the Electronic Product Code Information Services (EPCIS) and developed a prototype system through smart contracts and Ethereum. Tao et al. [16] introduced a hierarchical multi-domain blockchain network to maintain the network activities. A blockchain-based wine traceability system is proposed in Ref. [22]. The authors used blockchain to obtain secure, authenticated information to verify the provenance and purchase history of wine bottles. The proposed framework is implemented using Multichain, an open platform for implementing private blockchain solutions. However, a major downside to their approach was that it does not address scalability. In Ref. [23], Wang et al. proposed a product traceability system based on the Ethereum blockchain and the smart contract primitive. The system stores information related to the product life cycle and provides for the implementation of event-response mechanisms to verify the identities of both parties in all transactions at the time of their submission, so that their validity is guaranteed. While this idea had its positives, like the event-response mechanism and permanent log of events for future traceability, the drawback arises due to the fundamental nature of the blockchain, which is public, thus not being able to provide complete anonymity or privacy. Although the Ethereum blockchain offers reasonable encryption, there is definitely more scope for implementing more recent innovations. In Refs. [24,25], the authors proposed to maintain ownership information of manufactured goods on the blockchain. The products are coupled with RFID tags as they leave manufacturers, and the information in RFID tags is updated when the product is traded among different entities, storing ownership details on every transit. When a product reaches the retailer, a consumer can reject the product if the product history lacks the provenance information of the seller she is buying from. While this is a fairly effective approach for the traceability of products from manufacturer to retailers, the provenance of information prior to manufacturing cannot be obtained.

As mentioned until now, many papers cite implementations of blockchain for enterprise use-cases like supply chain, logistics, etc. [26–28]. However, from the perspective of a business model, there are certain constant factors, such as competition, sensitivity, and confidentiality that have to be maintained regardless of the underlying technology and development [29]. So there are certainly several challenges while integrating blockchain technology with a business use-case [30–32], mainly in protecting privacy [29,33,34] and security [35,36]. These challenges are attributed to most of the gaps in the aforementioned literature. Typical public blockchains are usually misconstrued with the expectation that they provide full anonymity, when in fact they only provide pseudoanonymity, and public blockchains also have scalability problems and usually lack a modular consensus mechanism, regardless of the record-keeping model [37].

In order to fill these gaps, a permissioned blockchain could be used. Permissioned blockchains, along with the advantages of a public blockchain, offer several features such as private data sharing, complete anonymity, etc., possibly making them the perfect solution for enterprises [38,39,40]. Permissioned blockchain systems such as Hyperledger Fabric offer a pluggable consensus mechanism, and they also offer cutting-edge blockchain security. Therefore, the primary research questions in this context would be: (i) What would be the paradigm shift from a centralized supply chain system to a decentralized one? (ii) How to overcome the drawbacks of a traditional non-blockchain system in the supply chain enterprise? (iii) In addition, how can one go further from a public blockchain and fill certain gaps such as privacy, scalability, and modularity? (iv) What would the solution be to overcome the crucial gap of lack of complete anonymity? (v) How to build a network using a permissioned blockchain system that provides a robust and balanced system satisfying all of the aforementioned factors?

In this paper, the coffee bean supply chain has been used to prove that permissioned blockchain technology could be integrated to improve the transparency, efficiency, security, privacy, scalability, and robustness in the supply chain industry. The supply chain for transacting coffee beans is a tedious process that includes planting, harvesting, hulling, drying, packing, bulking, blending, and roasting. In between these processes, the beans also pass through international traders, export sellers, and retailers like grocery stores, cafes, and specialty shops. The little or no communication between these involved organizations makes the whole system prone to fraud and scams. To make matters worse, the coffee trade supply chain industry is mostly reliant on emails and fax machines to send and receive contracts across the world, often making the network susceptible to errors and slowness. While digital web platforms have solved a few issues, the confidential data shared between the organizations are still vulnerable to hacker attacks. Therefore, a digital solution with better security and integrity for the shared data is of utmost significance. Possessing useful features such as identity management, identity mixer—anonymity, private data sharing, strong encryption protocols, and a pluggable framework, Hyperledger Fabric DLT is proposed as a suitable solution for this use-case of the coffee supply chain, aiming to illustrate a permissioned blockchain system that would serve as a solution to the previously mentioned research gaps.

3. Preliminaries

In this section, the fundamentals of the proposed system, i.e., the basics of blockchain technology, Hyperledger Fabric, and Hyperledger Caliper, are briefly described.

3.1. Blockchain technology

Blockchain provides distributed and shared storage to maintain the data. Thus, a single-point failure like a centralized system is automatically removed. The blockchain system becomes trustful because every peer in the system knows what is going on, and ultimately transparency is maintained. Also, it stores data in an immutable manner. Blockchain technology is an integration of different mechanisms and protocols. Fundamentally, it includes a consensus mechanism, cryptographic hash, distributed peer-to-peer network, immutable ledger, and mining, which are described as follows:

- Consensus mechanism: Granting transactions and validating transactions are performed through consensus protocols. For example, SOLO, KAFKA, and RAFT are commonly used.

- Cryptographic hashing: As cryptographic hashing is a one-way function and deterministic, it is used to create a chain of blocks. Blockchain technology uses various hash algorithms for connecting blocks.
- Immutable ledger: All transactions in a blockchain network are immutable so that they are tamper-proof and data cannot be modified during updates.
- Distributed peer-to-peer network: All transactions are broadcast through the network system among all peers to distribute and modify the data.
- Mining: Nonce values are used by miners to solve the challenge and enter the blockchain. This requires a high computation speed to achieve and obtain the reward.

In blockchain technology, network systems can be permissioned (private) or permissionless (public). In permissionless, any user can enter the system by solving a challenge. In permissioned, only authorized users can participate. Permissionless blockchain solutions commonly use the Ethereum framework and smart contracts. Smart contracts are like e-agreements between participants. In permissioned type, the Hyperledger fabric framework is popular.

3.2. Hyperledger Fabric

Hyperledger Fabric is a modular blockchain project governed by The Linux Foundation, which is a consortium that promotes decentralized innovations. As Hyperledger Fabric is permissioned and also provides smart contract support (chaincode), it is popular for a wide range of applications in several domains. Fabric allows the participants in a consortium to develop and deploy applications using blockchain. Hyperledger Fabric has a modular design and architecture, and thus possesses a high degree of flexibility and extensibility. Hyperledger Fabric can be divided into four components horizontally: identity management, ledger management, transaction management, and smart contracts; and vertically, Hyperledger Fabric can be divided into five other components: member management, consensus services, chaincode services, security, and cryptographic services. The blockchain network contains several peers (nodes) that host the blockchain, execute the chaincodes, and maintain the data in the ledger. Chaincodes are accessible to only authenticated peers to retain privacy. Other peers are unable to know what is going on. Thus, the roles of each peer are defined, and privacy is preserved. This is accomplished through the Fabric channels. Because of this channel, all chaincodes and data are only accessible to authenticated entities who are the participants in the channel. During the blockchain network setup, the identities are generated for authentication of the peers with respect to channels using cryptographic approaches. Ultimately, it can be determined whether a given peer is part of the corresponding channel or not. Furthermore, the Fabric network also requires an ordering service, which is provided by the orderer. The orderer is responsible for executing the total ordering of all transactions validated by the Fabric network on the corresponding channels. The executions of chaincode functions are essentially what make up transactions in a Fabric network. These transactions are isolated from other transactions of the same peer and also from the Fabric code, which is done by encapsulating chaincode executions within a Docker container. All chaincodes maintain a persistent state known as the key-value store. These key-value stores are updated by chaincode functions with the *get* and *put* methods that allow them to perform read and write operations for the key-value store. State databases such as LevelDB or CouchDB provide storage for these key-value pairs internally within the same node.

Fig. 1 shows the transaction steps in Fabric and are described as below:

1. Initiation of transaction: The client is the initiator of the transaction. Initially, a request proposal is created to invoke a chaincode function. Next, this proposal is signed by the client and submitted to the channel on which the chaincode is deployed. As per the policy of endorsement regarding chaincode, the client expects a number of endorsements to receive.
2. Execution of transaction: The endorsing peer (EP) verifies the signature. To do this, the endorsing peers do all the validity checks for the well-formedness, authenticity, replay protection, and authorization of the client. If all the checks are correctly passed, the transaction is executed by the peers against their own key-value stores, and then the peers generate an acknowledgment containing read-write sets produced as an output of the chaincode function execution. Next, all these values are signed by the peers and sent back to the client as an acknowledgment of the proposal or endorsement. No modifications to the ledger are made at this moment.
3. Collection of endorsements and ordering requests: A client receives all the endorsements and then examines, compares, and verifies

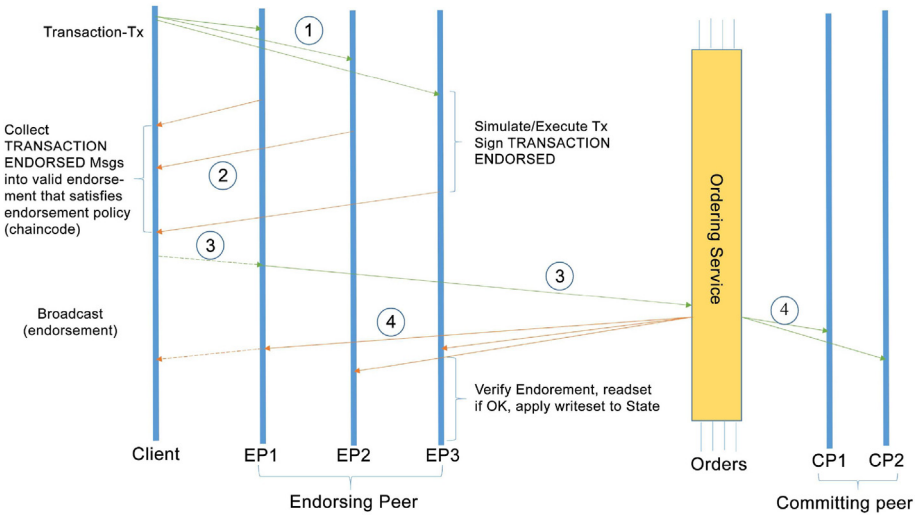

Fig. 1. Transaction flow in Hyperledger Fabric.

whether it has fulfilled all the requirements according to the endorsement and policy of the chaincode. For a read operation, the client does not send an ordering request. If the request is for a chaincode invocation, i.e., write, the endorsements are consolidated into a transaction and submitted to the orderer by the peer. The transaction and order are then verified by the orderer according to the channel.

4. Transaction validation and commit: The orderer delivers all ordered transactions within blocks to all peers on the channel. According to the endorsement policy, the transaction is verified by the peers, and if all the checks are correct, then the peers add the corresponding block to the ledger. It is mandatory for all the peers to commit (commit peer) the transaction. The endorsement can be done by only a particular subset of peers in the channel, and these peers are known as endorsing peers.

Through the use of Hyperledger Fabric to develop the coffee bean supply chain management system, a performance improvement over other technologies, such as Ethereum is achieved as Fabric provides fine-grained control over consensus. This would especially lead to relative performance gains for larger networks where the standard consensus algorithm that requires all nodes to achieve consensus might start to face some bottlenecks. In addition, the usage of Hyperledger Fabric also facilitates better performance (also scalability and privacy) because of its permissioned mode of operation, flexible transparency, and ability to extend pseudoanonymity (that is offered by Bitcoin or Ethereum blockchains) to full anonymity.

3.3. Hyperledger Caliper

One of the most critical aspects of blockchain users is the performance of a blockchain solution. However, presently there is no general tool for providing performance reviews of various blockchain technologies based on a collection of impartial and generally agreed guidelines. Some studies exist on the performance of various blockchain implementations in different contexts, but a blockchain benchmarking tool is not widely accepted. Hyperledger Caliper is a benchmark method for blockchain frames and is a benchmark goal for working on blockchain implementation. However, tools that can create a blockchain network quickly are ideal for working with Caliper. Hyperledger Caliper can generate reports containing various performance metrics, such as TPS, transaction latency, usage of resources, etc. The goal is to use Caliper results by other Hyperledger projects as they develop their applications and to support the option to implement a blockchain that meets the unique needs of a user.

3.4. Preliminary comparative analysis

There are mainly three different types of blockchain: public, private, and permissioned.

- In a public blockchain, anyone is free to join and participate in the core activities of the blockchain network.
- In a private blockchain, only selected entry of verified participants is allowed, and there is centralized control over the rights to overwrite, edit, or delete records on the blockchain.
- A permissioned blockchain draws a middle-ground between public and private blockchains, possessing properties of both. This type of blockchain has seen an increase in popularity, especially in enterprise use-cases, which is mainly attributed to their ability to allocate specific permissions to different users on the network.

The three types of blockchain mainly differ with regard to the following factors: decentralization, participation in the blockchain network, consensus mechanism, transparency, and anonymity. *Note: Since private blockchains are excessively centralized and used only for niche applications, they can be shelved from further analysis and consideration.*

In the context of this paper, the three different paradigms taken under analysis are the traditional system (non-blockchain), public blockchain, and permissioned blockchain (Hyperledger Fabric). There are two main components to any blockchain.

- **Blockchain Log** The blockchain log is fundamentally a transaction log that stores the history of all the transactions, i.e., changes to the state, in the blockchain's ledger. Each block being added to the blockchain contains a collection of transactions, which allows all participants to understand the history of changes in the ledger. This log is write-only, so if a specific transaction needs to be reverted or corrected, an additional transaction with new data has to be issued to override the previous transaction.
- **State Database** The state database describes the current state of all assets in the blockchain network. When the state of an asset changes, additional records are added with a new version number. In the absence of a state database, programs would have to calculate the current state by traversing through the entire transaction log. So the existence of a state database saves this overhead by allowing programs to directly access the current state value. In the context of Hyperledger Fabric, the state database is referred to as the world state, and the values stored inside are defined as key-value pairs. States can be created, updated, and erased, so the world state can change frequently.

Here, two more classifications can be drawn (independent of the paradigms described earlier) and added to the blockchain taxonomy based on how assets are defined, recorded and saved, and also based on the transition process from one state to another.

- **UTXO model (Unspent Transaction Output)** The UTXO model was initially designed to realize a peer-to-peer electronic cash system, i.e., Bitcoin. In the Bitcoin system, each transaction takes the UTXOs generated by the previous transactions and then “spends” them to create new UTXOs. The “balance” of an address at any given time is the sum of all the UTXOs currently spendable by that address. The global state at any given time is the set of all spendable UTXOs. The sum has to be calculated by traversing all the way down from the genesis block, which takes a significant amount of time, hence the overall slowness. Thus, in summary, the UTXO model is a stateless model in which all account balances over time are appended to the corresponding transaction records that are stored in the ledger. Examples of UTXO models include Bitcoin, Litecoin, Cardano, etc.
- **Account model** The Account model is different from the UTXO model in a fundamental way. Instead of tracking the location of an asset, the system simply maintains the account balance of each individual. The accounts can be controlled by a private key or a smart contract. A transaction is recorded by adding and subtracting the same amount of tokens from the sender and to the recipient, respectively. The Account mode was initially designed to facilitate smart contacts in the Ethereum blockchain. When a user creates an Ether wallet and receives the first transaction, an account that is controlled by a private key is added to the global state and stored across all nodes in the network. The control of the network is linked to the code by deploying smart contracts. Funds can be held by smart contracts, which are then redistributed by the smart contracts based on the conditions defined in the logic of the contract. Every account in Ethereum comprises a balance, storage, and code-space for calling other accounts. Thus, in summary, the Account model is a stateful model in which all account balances are stored in the ledger through the storage space allocated specifically for the state. Examples of Account models include Ethereum, Solana, Ripple, etc.

Note: Hyperledger Fabric 1.4.7 (the version that was used for the work described in this paper) uses the Account model by default, but it also supports

the UTXO model if needed. Unlike UTXO, the Account model is more effective for record-keeping when the same blockchain has to deal with multiple asset types, hence, it is inherently more appropriate for the work described in this paper.

When tabulated against the two components blockchain log and state database, the previously mentioned paradigms would have different permutations, as shown in Table 1.

As shown in Table 1, in the case of a public blockchain such as Bitcoin that incorporates the UTXO transaction model, the state database is absent, so the blockchain is traversed through to calculate the current state every time. Hyperledger Fabric, on the one hand, avoids this, for the sake of speed and efficiency. On the other hand, although account-based models such as Ethereum do possess a state database, Hyperledger Fabric nevertheless comes out on top as the better solution—Ledger states in Hyperledger Fabric are defined as key-value pairs by default, and this contributes a lot to the flexibility of the framework. The world state in Hyperledger Fabric is structurally implemented as a database so that the storage and retrieval of data ledger states are made simple and efficient. The world state database can be implemented in more than one way, depending on the type and complexity of values stored inside the ledger states. The two major choices for implementing the world state database are CouchDB and LevelDB. There is also a possibility of a completely custom implementation. This flexibility in pluggable implementations goes hand-in-hand with an important aspect of Hyperledger Fabric—modularity. Hyperledger Fabric provides immense flexibility in configuring the type of a ledger state; for example, it could be a temporal database, graph or data store, etc. This is more efficient to access based on the use-case and therefore allows the system to address a wide range of problems.

Conflating the aforementioned aspects and the context of the supply chain, Hyperledger Fabric would certainly be helpful in solving certain issues and enabling other advantageous features. In a typical supply chain, tracking an item's provenance—the place of origin, when it was created, every time it was exchanged or handed over elsewhere to multiple parties—is of utmost importance, as it ensures that there is a transparent and tamper-proof chain of titles verifying the ownership and status of the product that a business sells. In typical businesses, there is no log and only the current state, so there is no way to check and verify provenance. Evidently, the notions of decentralization and privacy are at odds, typically in such enterprise use-cases. Private channels in Hyperledger Fabric solve this issue, allowing multiple business parties to coexist in the same network.

Table 1
Comparison of paradigms with regard to components.

	Blockchain Log	State Database
Traditional System	Absent	Present
Public Blockchain	Present	Absent (UTXO) Present (Account)
Hyperledger Fabric	Present	Present

4. Overview of the coffee supply chain

Fig. 2 shows the traditional coffee supply chain. The coffee supply chain is the intricate string of processes and stakeholders involved in taking coffee beans from the farms where they are grown into cups all around the world. There are numerous stages involved in this complex project, and for the scope of this paper, the chain has only been consolidated to its salient stages: cultivation, farm inspector, harvester, exporter, importer, and processor.

1. Farmer

The farmer does the cultivation. Cultivation is the first stage in the start of a conversion of a natural resource to a consumer-useable product. Information such as the farm details and date of planting is taken here, and from here, the processes involved in getting it to all the consumers can begin. With the growing sizes of farms and modernization of agricultural practices, farm inspections have grown as a vital step in the supply chain process.

2. Farm inspector

Farm inspectors gather information such as the details pertaining to the materials used in the growth of the coffee plant, like the family of coffee, type of seed, or type of fertilizer. Coffee farmers are involved twice in this chain, as harvesting is a separate process after the farm inspection.

3. Harvester

Harvesters pluck, hull, polish, grade, and sort the beans that grow on the farm. During this process, a track of information on crop variety, temperature, humidity, and picking methods could be useful to the organizations or end-users.

4. Exporter

After harvesting, exporters play a crucial role in dispatching coffee beans throughout the world to satisfy the demand for coffee, including places where coffee cultivation is still tricky. Information such as the exporter details, transportation details, package details, and other necessary data are collected in this stage.

5. Importer

In succession to this step, the coffee beans are imported in bulk quantities by various vendors and are stored at warehouses before moving to the final stage of the supply chain. Data recorded during this step include importer details, transportation details, and warehouse details.

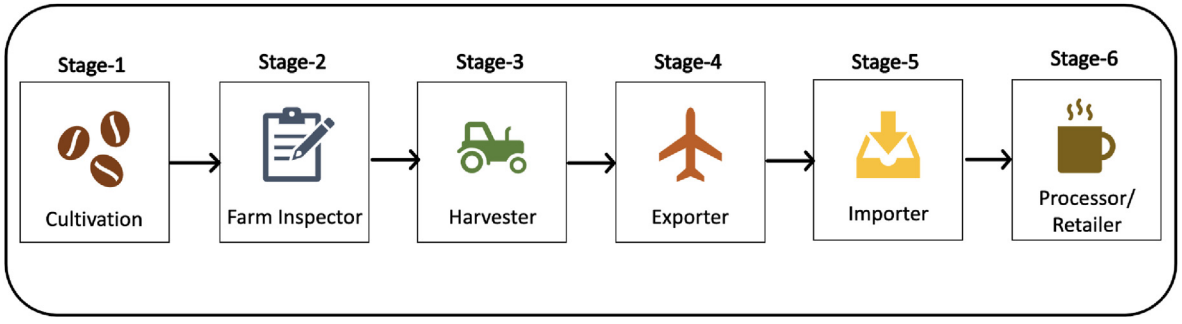

Fig. 2. Traditional coffee supply chain.

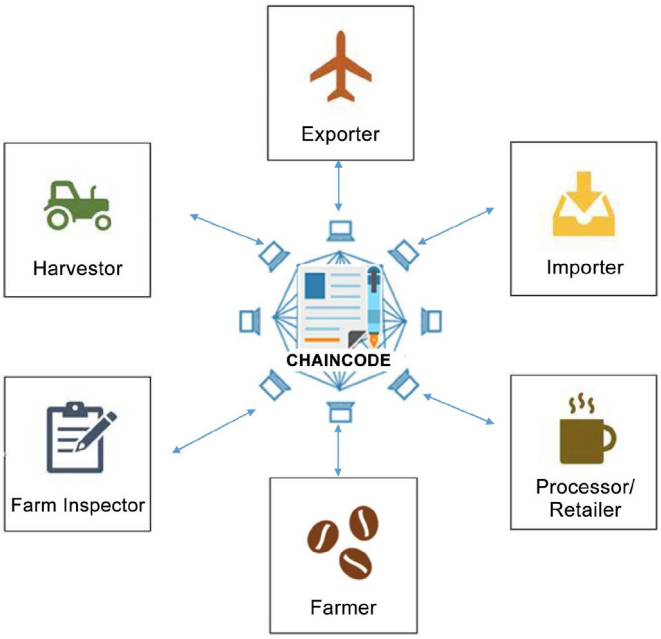

Fig. 3. Coffee supply chain using Hyperledger Fabric.

6. Processor

Processors are the face of the supply chain and are the source of the final coffee product to consumers. Notably, in the coffee industry, the types of processors vary greatly, ranging from direct sellers to supermarkets. With the advent of e-commerce, the variety of processors (or retailers) relating to the coffee industry has increased further. Processors make many strategic decisions and additions that could potentially alter the type of end-products. Therefore, the information during this processing phase, such as processing conditions, flavor additives, and other product information, becomes necessary to the eyes of end-users and is thus recorded for consumer safety.

By working through the details of the general supply chain process in the coffee industry, it is evident that each stage in the network holds many essential details that need to be made more available to the end-users. Therefore, an attempt to improve the overall visibility of the data involved in the coffee supply chain using permissioned blockchain via Hyperledger Fabric is shown in Fig. 3. The detailed work has been addressed in the next section.

5. Proposed system

The details of the participants are described through a class diagram as shown in Fig. 4. Each participant has *BatchID* and *Stage* to identify and process the stages. Additional attributes vary according to the nature of the stage.

Fig. 5 shows the proposed architecture using Hyperledger Fabric. The participants are connected to each other through Hyperledger Fabric. Each participant maintains the peers along with the chaincode and ledger. Each peer has to take membership to be involved in the fabric. The membership service provider issues certificates, ECert (Enrollment Certificates), through the CA (Certificate Authority) to the requested peers. These certificates act as identities for the peers. To execute transactions in the Hyperledger Fabric environment, chaincode procedures are called.

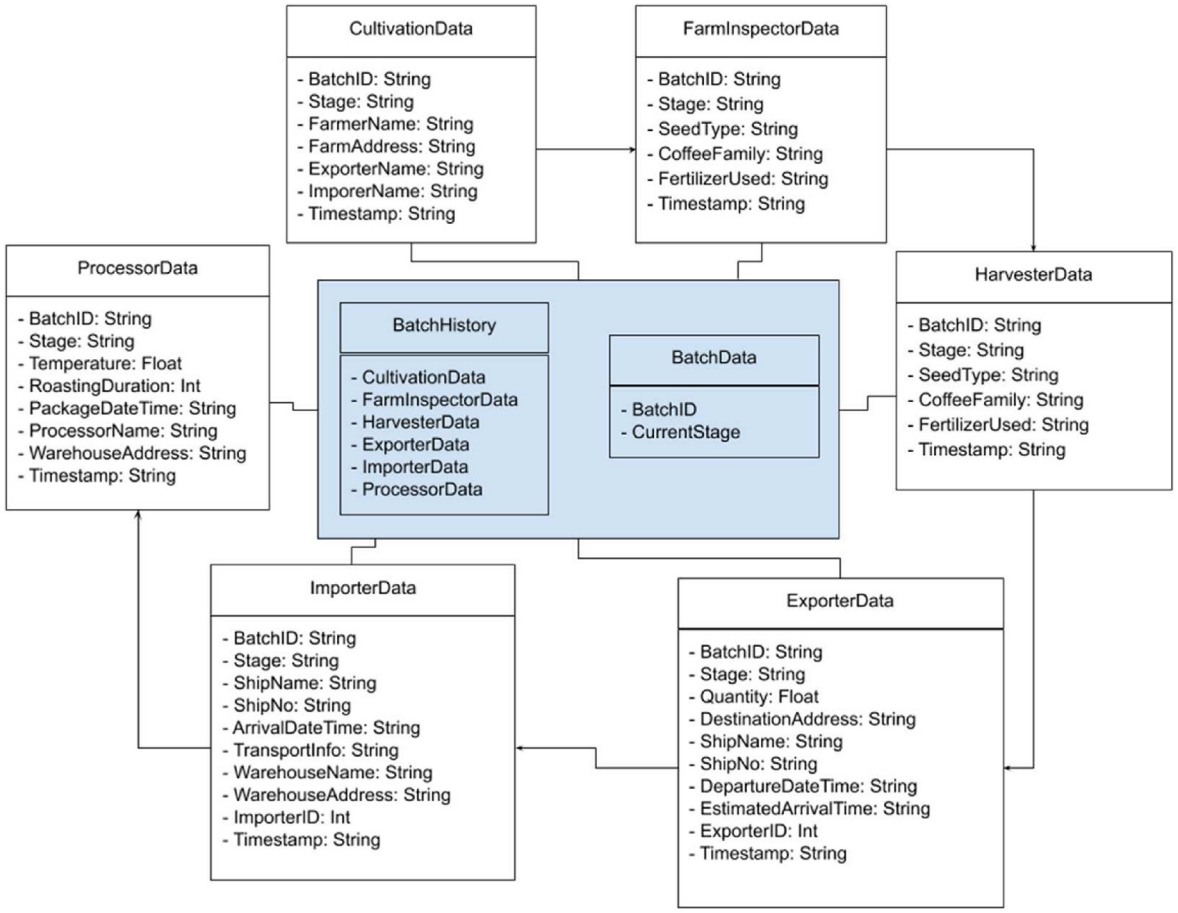

Fig. 4. Class Diagram of the system.

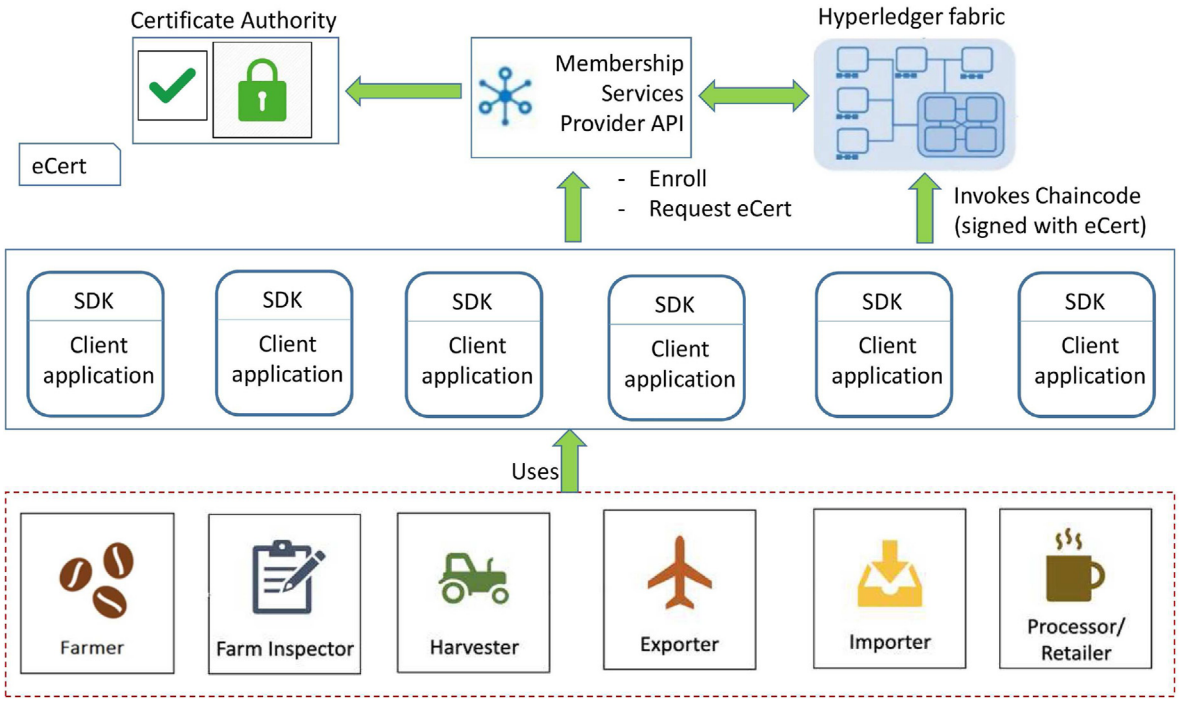

Fig. 5. Proposed architecture.

Chaincode procedures are initiated by any authenticated peer and executed by all peers in the network.

The detailed work is described in the next subsequent subsections.

5.1. *Network architecture*

Fig. 6 shows the blockchain network architecture used for the coffee supply chain. Under the proposed network architecture, every organization involved in the supply chain is part of one channel. Hyperledger Fabric networks are structured in a way that each channel is linked to a single ledger, and the ledger is only accessible to the peers within the same channel. Therefore, every organization has access to a copy of the same ledger, which contains all data related to the supply chain. Since the entire database is modeled on a ledger, the properties of DLT ensure data integrity—data on the ledger is immutable and tamper-proof. Each organization has multiple peers for the sake of extra redundancy of data. Within an organizational system, all the peers can communicate with peers within the organization. To communicate between organizations, a single peer from an organization is assigned as the anchor peer, which is directly able to exchange information via the channel. A peer that is not an anchor peer communicates with other organizations via the anchor peer. If the anchor peer is down at any time, a different peer will become the anchor peer, and thus communication between organizations would not be affected at any point of time due to this failsafe mechanism.

Each organization in the proposed network consists of two peers. The proposed network follows the raft consensus protocol. According to this protocol, the server cluster has leaders, candidates, and followers. The members of the cluster vote for a cluster leader. A single cluster can only have one cluster. When the client specifies a command to the server, the leader nodes fire an AppendEntry RPC (Remote Procedure Call) to instruct the followers to update/sync their logs to match the new entry in the leader's logs. When the majority of the servers commit the new change, the change is considered as committed. Once this is completed, the leader executes the new entry and sends the result of the same back to the client. Hyperledger Fabric exclusively uses Raft as its consensus algorithm, deprecating the Kafka support in the process due to the higher

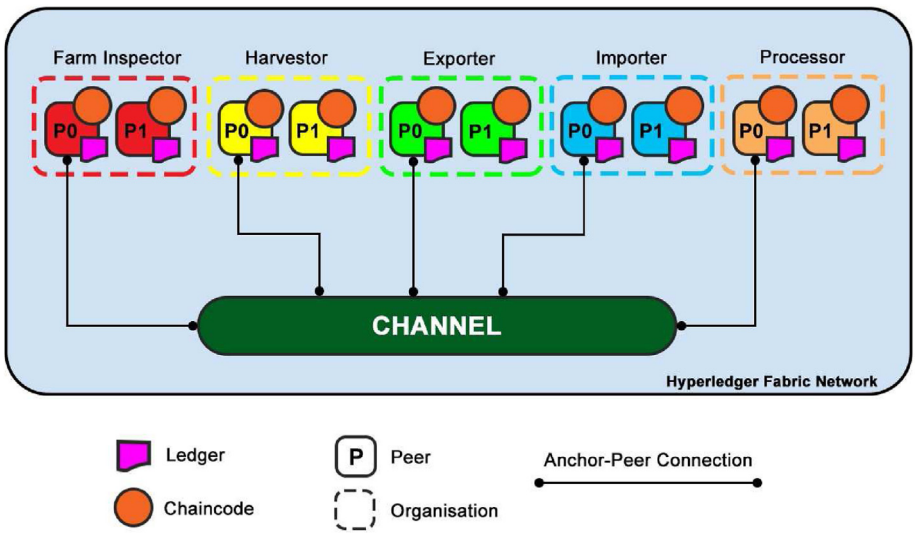

Fig. 6. Network architecture.

success rate and throughput of Raft relative to Kafka.

5.2. Chaincode (SDK layer)

All the chaincodes that are installed on the peers and interact with the channel form a layer of the application called the Fabric SDK Layer. Since the same chaincode is installed and instantiated on all peers of all the organizations, implicit permission for any organization to update any of the stages of the supply chain is provided. To avoid this, and to only allow an organization to update data pertinent to its own working and operation. Thus, the chaincode itself must be designed and programmed in a way that validates the current stage of the batch of coffee beans in question and only allows transactions to be approved if they are sent by the correct organization at the correct stage. With effective code to accomplish this, the chain becomes tamper-proof from within the organizations of the supply chain.

5.2.1. Asset definition

The chaincode enables users of different organizations to provide data from their corresponding stage. Each batch is identified by its batch ID. The information related to an individual stage is treated as an asset, and thus, for each stage in the supply chain, there is an asset definition in the chaincode. With all these asset definitions, the chaincode provides functionality to store stage data, query existing data, and complete batch history for any given batch of coffee beans.

5.3. REST API layer

The invocation of the chaincode is performed through the Fabric SDK. All transaction proposals to store or query are executed directly at this layer. However, it is a good practice to abstract this process behind an intermediate layer of communication. In the proposed work, the abstraction is done using a REST API layer. This layer exposes basic endpoints to the user, such as GET and POST, to call for chaincode invocations. This simplifies the usage of the blockchain network for end-users while abstracting and hiding the complexities of the chaincode business logic.

In order to create a REST API layer that successfully interacts with the blockchain, the identities of all the users must be stored in a wallet with the corresponding certificates and keys, which are generated at the time of network creation using the cryptogen tool. The Hyperledger Fabric SDK is used to check the certificates and register users in the wallet. This ensures that only users with the right credentials are able to access the network and invoke chaincode. If the new certificates are assigned in the future for any given user, that user will not be able to access the network untill their identity is registered in the wallet using the new certificates.

Each REST endpoint follows four steps in its execution. Firstly, the SDK establishes a connection with the Fabric gateway using a relevant identity in the wallet. Secondly, the SDK accesses the network channel on which the operations need to be performed. Subsequently, the specific contract containing the chaincode to be executed is called. If any of these steps fails due to the network being down, invalid credentials, or any other reason, the API will not submit a transaction to the network. If all three steps are executed successfully, a transaction is prepared using the input arguments and submitted to the Fabric layer. When the chaincode is invoked and sends the response to the REST layer, the same response is sent back to the user.

Fig. 7 describes the communication messages carried out between the farmer and the Hyperledger Fabric layer through the REST API layer during Create Batch. The farmer has to only request Create Batch along with the required data to the Hyperledger Fabric through the REST API Layer. The REST API layer takes the responsibility of all other things to execute CreateBatch. The communication between the REST API Layer and the Hyperledger Fabric Layer goes as given below:

1. The REST API layer requests for the connection to the Hyperledger Fabric layer.
2. If the connection succeeds, Hyperledger returns positive acknowledgment. The REST API layer requests for the channel.
3. In response to Hyperledger, send the corresponding channel reference.

5.4. Batch management

The proposed system is simulated through *batch* management. The *batch* is an object which contains information about all the participants of the supply chain, like a farmer, farm inspector, etc. *batch* management is achieved through three operations: Create batch, Update batch, and Query batch.

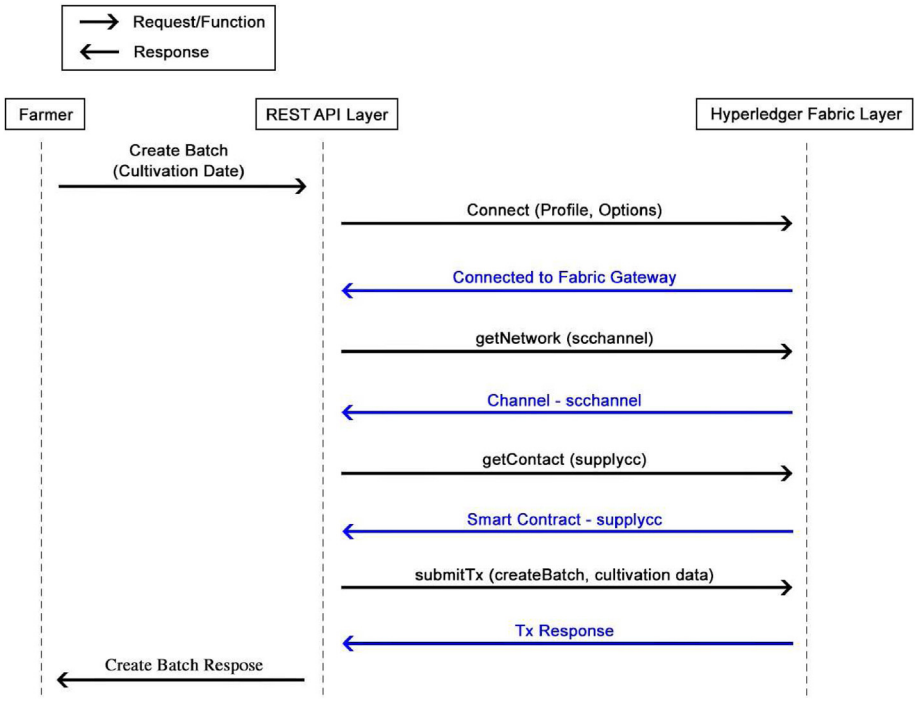

Fig. 7. Sample sequence diagram.

1. **CreateBatch** During the creation of a new batch, only information regarding cultivation is used. The attributes batchID, batchStage, FarmerName, FarmAddress, ExporterName, ImporterName, and Timestamp are updated with given input values to create an object. Creation of *batch* is done through the execution of Algorithm 1.
2. **UpdateBatch** As the process becomes progressive, the stages are updated by adding the next stage information to the batch. *batch* is extracted from the blockchain using *id*. The current stage and relevant attributes of the data are updated, and once again, *batch* is restored to the blockchain. The process is described in Algorithm 2.
3. **QueryBatch** To know the status of the *batch*, *id* is used to extract the corresponding batch from the blockchain. As per the batch's stage, the corresponding data are extracted. e.g., if batchStage is 'Farm-Inspector', then FarmInspector's data, i.e., SeedType, CoffeeFamily, FertilizerUsed, and Timestamp are displayed. For the other stages, the corresponding data, as shown in the class diagram, are displayed. The Query Batch process is described in Algorithm 3. Next, the REST API layer sends a request for the chaincode (smart contract). In response, Hyperledger returns the corresponding chaincode reference followed by a REST API request for the execution of the transaction for CreateBatch. Finally, the Hyperledger Fabric layer sends the status of the transaction, i.e., succeed or failed.

Algorithm 1: CreateBatch

```
input: Object cData which contain the cultivation data of the new batch as per the asset
       definitions
/* Store(key,value) is used to store data to the Blockchain as a key-value
   pair                                                                    */
if cData is an object of CultivationData then
    Create BatchDetails object batch;
    batch.batchID ← cData.batchID;
    batch.currentStage ← 'Cultivation Stage';
    batch.FarmerName ← cData.FarmName;
    batch.FarmeAddress ← cData.FarmAdress;
    batch.Exporter ← cData.Exporter ;
    batch.Timestamp ← cData.Timestamp;
    Store(batch.batchID, batch);
    if batch successfully stored then
        return 'success';
    else
        return 'invalid data';
    end
end
```

Algorithm 2: UpdateBatch

```
input: id, nextStage, data which are the id of the batch to be updated, the stage
       corresponding to the data being provided, data to be added respectively
if id does not exist in the blockchain then
    return 'invalid id';
if nextStage is not the correct stage to be updated then
    return 'invalid stage';
if data has incomplete fields then
    return 'incomplete data';
end
batch ← getBatch(id);
batch.currentStage ← nextStage;
Store(batch.batchID, batch);
return 'success';
```

Algorithm 3: QueryBatch

```
input: id which is the id of the batch to be queried
/* Get(key) is used to read the value of the given key */
batch ← Get(id);
Create object bHistory of BatchHistory;
if id exists in the blockchian then
    stage←'Cultivation';
    bHistory.CultivationData ← Get(id,stage);
    if stage == batch.currentStage then
        return bHistory;
    end
    stage←'FarmInspector';
    bHistory.FarmInspectorData ← Get(id,stage);
    if stage == batch.currentStage then
        return bHistory;
    end
    stage←'Harvester';
    bHistory.HarvesterData ← Get(id,stage);
    if stage == batch.currentStage then
        return bHistory;
    end
    stage←'Exporter';
    bHistory.ExporterData ← Get(id,stage);
    if stage == batch.currentStage then
        return bHistory;
    end
    stage←'Importer';
    bHistory.ImporterData ← Get(id,stage);
    if stage == batch.currentStage then
        return bHistory;
    end
    stage←'Processor';
    bHistory.ProcessorData ← Get(id,stage);
    return bHistory;
else
    return "batchID does not exist";
end
```

5.5. Membership and Access Control

Because of Hyperledger Fabric, the blockchain network becomes private, and only defined stakeholders can participate in the system. Hence, the security of the system is retained. This is because of both the special access mechanism and the channel settings. Therefore, it would be a combination of the following that helps enable privacy and security in Hyperledger Fabric:

- **Membership Service Provider (MSP):** Provides and manages the identities and authenticated participants. In particular, an MSP provides an abstraction of cryptographic mechanisms and protocols behind user authentication, issuing certificates, and verifying certificates. An MSP can also define its own notion of identity and the rules by which that identity is authenticated (signature generation and verification) and governed (identity validation). It is mandatory that every node has a local MSP defined.

- **Access Control List (ACL):** This is an additional layer of permission for user identities, and it also limits the permissions for various network operations. For example, it might allow one operation (invoke chaincode) for one user but restrict some other operation (deploying new chaincode) for a different user.

Fig. 8 depicts how the MSP is structured among organizations in this paper's context of the coffee supply chain. Although the actual architecture ought to have all the organizations, as shown in Fig. 3, the subarchitecture of just two organizations would suffice for a description, as the rest would be a mere extension of this. Taking the Farm Inspector (ORG1) and Harvestor (ORG2) organizations, for instance, this would be the flow of interaction—RootCA1 issues an identity to an administrator *A*. Using this identity, *A* connects to the peer, and tries to install a smart contract on the peer. During this process, the peer checks its local MSP, ORG1-MSP, to verify that the identity of *A* is that of a member belonging to ORG1. Successful verification will allow the installation to take place successfully. Finally, *A* is able to instantiate the smart contract on the channel and because this is an operation within a channel, all organizations participating in the channel must first agree to it. Therefore, the peer must always check the MSPs of the channel before it initiates this chain of commands.

5.5.1. *Public key infrastructure (PKI)*

Hyperledger Fabric implements a standard PKI-based MSP, which is implemented by default, as shown in Fig. 9. There are two main components to the MSP PKI: VerifierMSP and SignerMSP. The VerifierMSP is used on a per-channel basis, whereas the SignerMSP is used only on a local basis. Most of the aspects remain constant between the two components:

- The identity is governed by a standard PKI hierarchy: RootCA→IntermediateCA→LeafCA, using the standard x.509 certificates.
- Hyperledger Fabric predominantly supports ECDSA, with very limited support for RSA.
- Anonymity is not supported.

The VerifierMSP provides the signature verification mechanism for peers, orderers, and clients. The SignerMSP is used by the same entities to sign messages and authenticate off-chain messages.

5.5.2. *Notions of privacy*

As mentioned several times before, the primary aspect of this paper's proposed solution is to introduce the required notions of privacy. There are a few types of privacy in any blockchain system:

- **Transaction Data Privacy**: Transactional activity of an entity such as peer, orderer, or node
- **State Data Privacy**: Chaincode and smart contract data
- **Smart Contract Privacy**: Logic of the chaincode or smart contract
- **User Privacy**: Anonymity and unlinkability

As mentioned earlier in the paper, none of these aspects of privacy are supported by permissionless blockchain systems; they probably provide pseudoanonymity at maximum; therefore, these systems would not be compatible with enterprises, hence making permissioned blockchain systems the optimum solution.

The aspects of transaction privacy, state data privacy, and smart contract privacy are implemented by default by Hyperledger Fabric in a standard way that closes all the gaps in compliance with a standard permissioned blockchain system. However, the notion of user privacy—anonymity and unlinkability—is a fairly recent and flexible implementation in Hyperledger Fabric, which can be leveraged in a crucial way.

5.5.3. *Identity mixer*

Identity mixer is an open source project that was integrated into Hyperledger Fabric recently and uses cryptography as its core underlying logic. Anonymity and unlinkability are strongly guaranteed by Identity mixer.

Fig. 10 briefly shows the high-level functionality of the identity mixer. Fig. 10 is merely an extension of Fig. 5, with only a subset of the architecture taken under the purview, for two organizations—Harvestor and Exporter. The actual implementation would be just a scaled-up version of the architecture shown in Fig. 10.

The blockchain user has to obtain a certificate from a CA, so there is an MSP that interacts with the CA to issue certificates, and there is

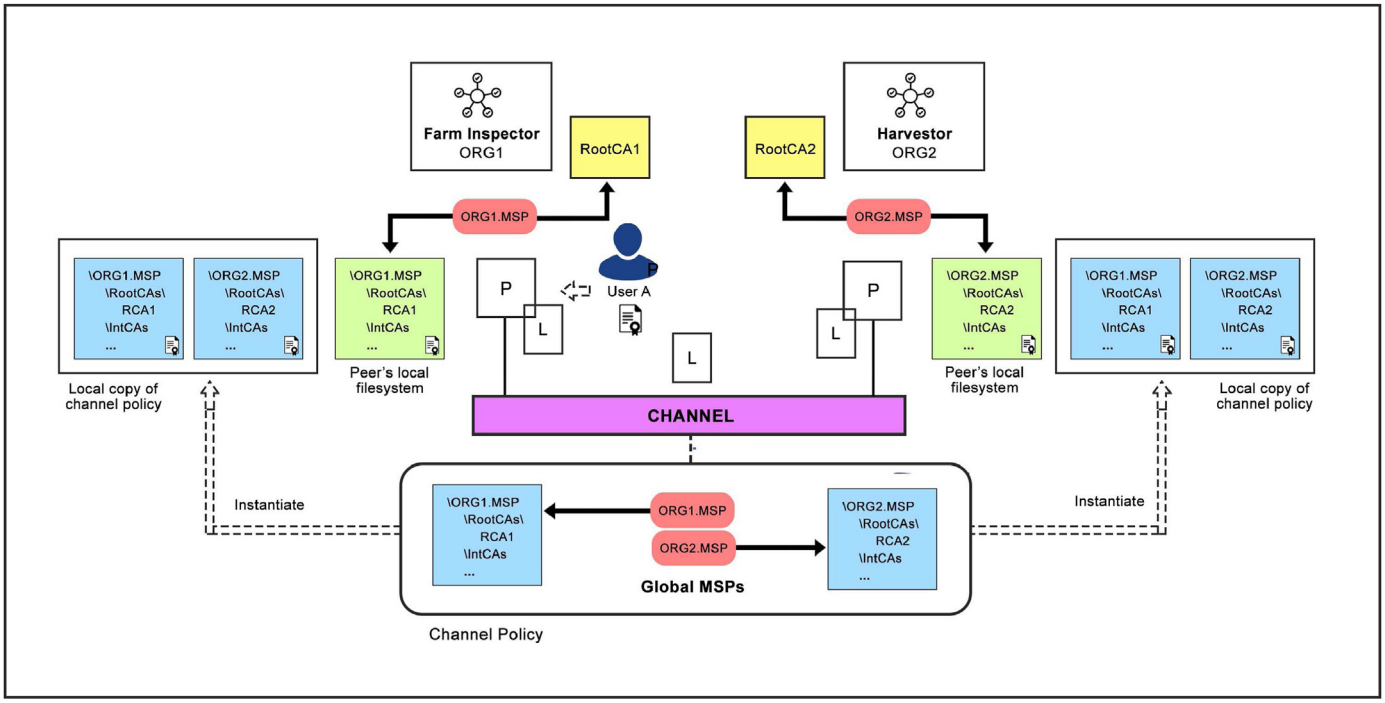

Fig. 8. Membership service provider (MSP) architecture.

an ECert. When the user invokes smart contract transactions, the transaction was previously signed by the ECert. Instead, what the identity mixer allows is for per-transaction certificates to be generated by the user, there are one-time-use certificates that can be used for each transaction. The same can be done with keys—one-time-use keys can be generated for encryption. Therefore, the user can prove that all

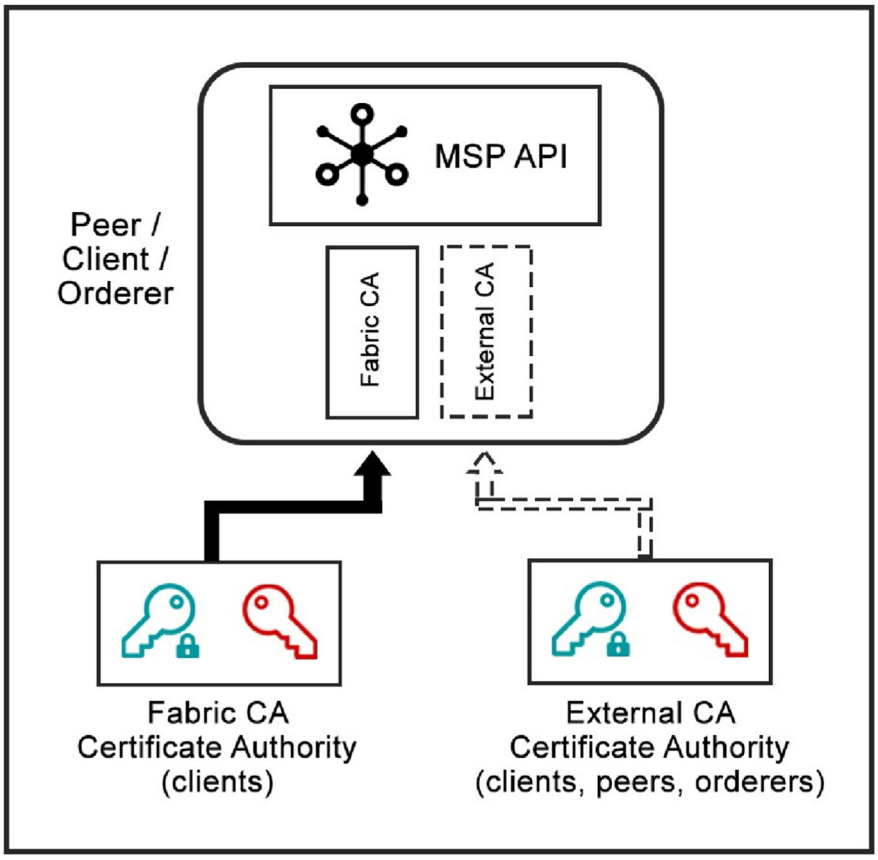

Fig. 9. Membership service provider (MSP) public key infrastructure. CA: Certificate Authority.

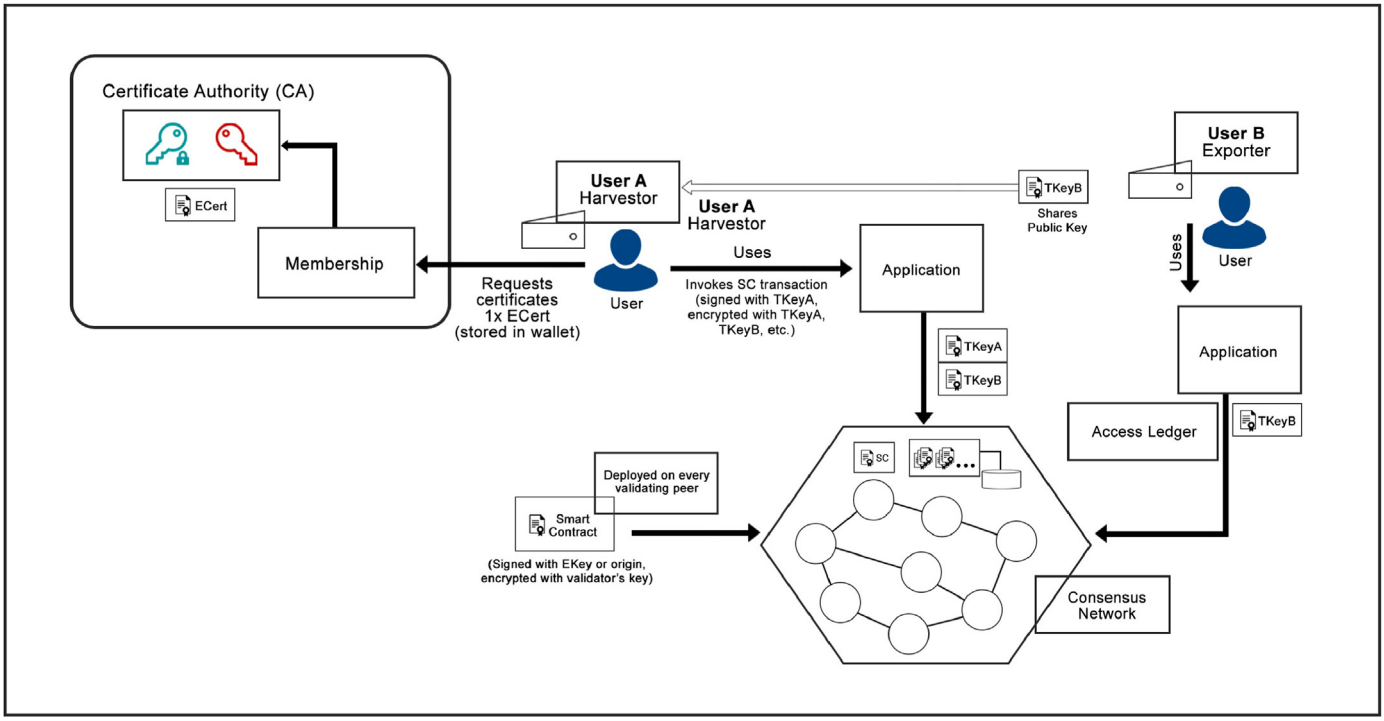

Fig. 10. Identity mixer.

these keys were derived from the ECert, thereby providing the unique ability where the user can generate multiple keys from the original ECert, and the user can also prove that they were generated by the same ECert, i.e., all transactions were performed by a particular user and can be proved to an auditor in the network if needed, but no other entity in the system would know that all the transactions were performed by the particular user.

5.5.4. Comparing X.509 and identity mixer

Identity mixer is distinct from the Fabric CA that was mentioned earlier, although it could be considered as an alternative implementation of the Fabric CA itself. The Fabric CA is based on X.509 certificates, whereas identity mixer goes about a slightly different method.

As shown in Fig. 11, X.509 works in the following way: There is a CA that issues a certificate to the user, and this certificate can have multiple attributes and whenever the user performs a transaction, it signs using this certificate. Any other entity in the system who can view these transactions will be able to see their attributes, thereby knowing that the transaction was performed by the particular user. In the case of identity mixer, as before, an ECert is issued to a user with multiple attributes, but here, the user has the additional flexibility of choosing to disclose only a specific attribute for a particular transaction, with the help of a presentation policy. In this generic example shown in Fig. 11, there can be two policies: based on policy 1, the user can choose to just disclose the first attribute, and based on policy 2 (which can be a different channel altogether, or it can be a different chaincode, or a different transaction within the same chaincode), it discloses only the second attribute. Therefore, there are distinct transaction certificates exposing different attributes in each case, abstracting these transactions as completely independent, unique, and unrelated when viewed by the other entities in the system. Hence, the notions of anonymity and unlinkability are both successfully satisfied.

5.6. Comparison between the proposed solution and contemporary methods

Based on the features and points posed till now, a comparison can be drawn between the proposed solution—Hyperledger Fabric DLT; and the contemporary systems in place—traditional system (non-blockchain) and public blockchain.

As shown in Table 2, Hyperledger Fabric clearly outweighs the other two contemporary solutions in terms of its practical advantages, especially in an enterprise scenario—supply chain. The traditional system that has no blockchain perspective to it and the public blockchain that follows right after are the two contemporary methods that have been used until the recent past. The public blockchain, in most ways, can be deemed as a negatively excess of a solution, as it successfully solves the problems of a centralized system, transparency, immutability, authenticity, reliability, etc., but on the other hand, engenders new issues like inflexibility in privacy and confidentiality, inability to share data privately, lack of modularity, etc. Hence, a permissioned blockchain, Hyperledger Fabric, allows us to mitigate these new issues by offering a set of features that meet the middle ground between public and private blockchains, and it also supports several customization options, thereby drawing a balance between the two. These features include identity verification of participants before they join a permissioned channel, and issuing these participants their designated permissions that enable them to perform only a particular set of tasks and access a particular set of data on the network.

In the context of this paper's use-case scenario of a coffee supply chain network, there are multiple phases and entities actively involved when a blockchain system is integrated. This includes managing all dealings with

Table 2
Comparison among traditional system, public blockchain, and hyperledger distributed ledger technology (DLT).

	Traditional system (non-blockchain)	Public blockchain	Hyperledger Fabric
Blockchain log	×	✓	✓
Pluggable state database	✓	×	✓
Distributed ledger	×	✓	✓
Flexible transparency	×	×	✓
Immutability	×	✓	✓
Private data sharing	×	×	✓
Confidentiality	×	×	✓
Modularity	×	×	✓
Collision unlikelihood	×	×	✓
Pluggable consensus mechanism	×	×	✓

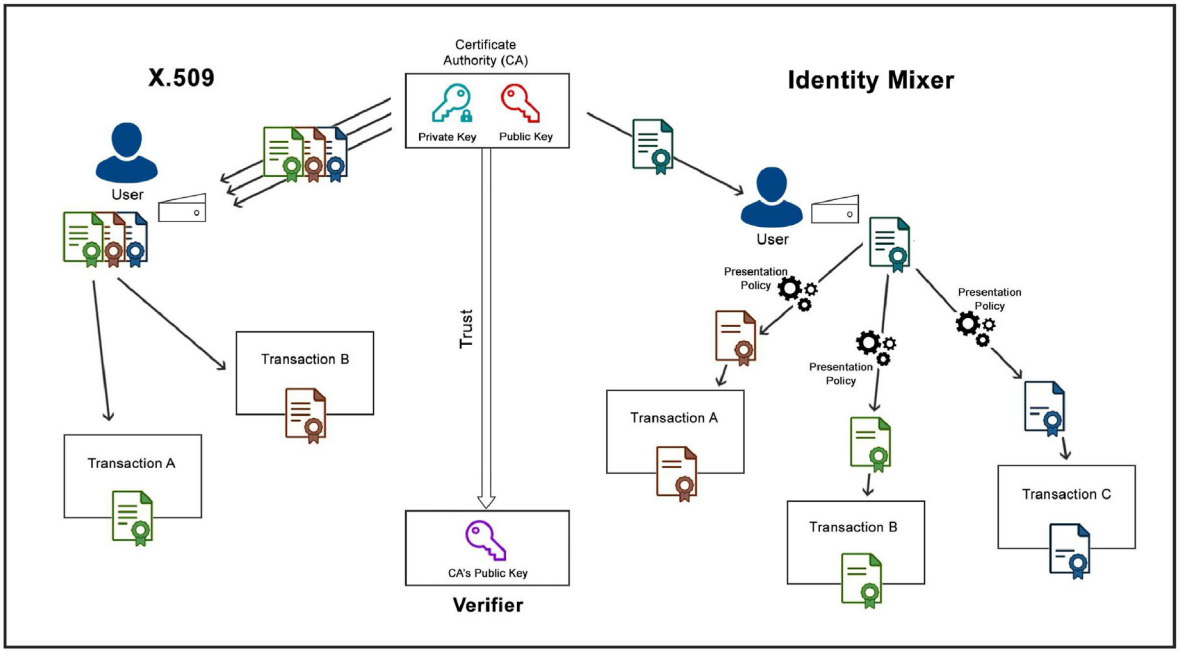

Fig. 11. Comparing X.509 and identity mixer.

the farm produce from its origin (the farm) to the end consumer (the market). Typically, a farmer cultivates the coffee bean and eventually ships it to multiple market locations across the world. In this transaction process, there are several other entities who get involved in crucial and specific roles—shipping companies to transport the produce, warehouse operators who safely store the product, customs authorities who approve the transport of produce between locations, etc. In such a network, permissioned channels would certainly provide the best implementation and solution. Naturally, it is possible that a farmer may fix varying prices for buyers across different locations, depending on the market, competition, and feasibility. In this case, the entities that come in between the farmer and buyer may not be required to know such information as it might be confidential and simply outside their purview. They would only need access to limited information, like quantity and quality specifications, to perform their function in supporting the overall transaction. This use-case is just a small subset of a much larger scope of solutions that can be implemented with Hyperledger Fabric, especially in the context of supply chain industries. To sum it up broadly, the aforementioned features and advantages of Hyperledger Fabric (over the alternative methods) cater to a spectrum of solutions to various existing problems and gaps in the contemporary (blockchain) supply chain industry, such as the following:

- **Control** Hyperledger Fabric provides full control of the network, having no dependency on mining and cost for updating the state of the supply chain. On the other hand, alternative blockchain solutions would require mining and each state update can only happen at a certain cost (cryptocurrency). This former is useful in the supply chain industry as private organizations and stakeholders cannot afford to fully decentralize the control of the blockchain infrastructure because they would need full power to make changes and amend an urgent problem if needed.
- **Peer Role** In other blockchain solutions, each peer has a role such that if a transaction takes place, several nodes have to participate in its completion, which leads to several issues such as scalability, privacy, and efficiency. Hyperledger Fabric incorporates a DLT that does not require each peer to be informed in order for a transaction to take place. This is extremely useful for solving the following problems:
  - Not all peers in a supply chain framework need to know or be aware of a particular transaction, thus paving the way for individual and separate permissioned networks facilitating privacy among the different and possibly competing organizations participating in the blockchain network.
  - The dependency on the participation of redundant peers for a particular transaction is reduced, thereby making the supply chain network more efficient and easier to scale.
- **Operation Mode** The operation mode in alternative public blockchains is public, meaning anyone can access the blockchain network, and no permission is needed to access the network. The status of the supply chain, which is being used to track an asset, being completely public is inexcusable for any private organizations using the network. Hyperledger Fabric is a permissioned blockchain, meaning only the authorized participants can access the network. In other words, the network is limited to a predefined community of participants with permission. This is the fundamental use-case of the proposed solution for a supply chain framework, as there would be multiple participants that are in different chains, and it's imperative from a business-logic perspective that a particular group of participants does not know the details of transactions being carried out by another group of (possibly competing) participants. So Hyperledger Fabric solves this by offering the flexibility of configuring this access/privacy setting among the different peers in the network with the help of permissioned channels.
- **Confidentiality and Privacy** This is an extension to the previous point about Operation Mode. If we consider public blockchain, it is a public network, which means all the transactions recorded on the blockchain network are visible and accessible by every peer. Whereas Hyperledger Fabric is private and permissioned, which means transactions taking place on the network are only visible to the authorized members. Confidentiality is enabled in Hyperledger Fabric through its permissioned network and channels, also providing an architecture that can handle private data. Sub-networks are established within channels by the members of the network, wherein every member has restricted access to only a particular set of transactions. Thus, components such as chaincode and transacted data within a channel are only accessible to nodes that participate in that specific channel, which preserves the confidentiality and privacy of these components. It is also possible for private data to flow between members of the same channel, this would have the same protection as a channel but without the overhead of creating and maintaining a separate channel. This use-case goes hand-in-hand with being compliant with the fundamental business logic of any private organization participating in the supply chain network, hence helping bridge the gap in terms of privacy and confidentiality.
- **Consensus Mechanism** Alternative blockchain solutions leverage consensus protocols like Proof-of-Work (PoW), Proof-of-Stake (PoS), etc. in which all the nodes have to reach a consensus. Hyperledger Fabric, on the other hand, uses a different type of consensus. In this implementation, nodes are allowed to choose between no consensus needed and an agreement protocol, again providing much needed flexibility in configuring the consensus protocol based on the use-case. It builds the consensus infrastructure as a combination of services called endorsing and ordering. The ordering of transactions is delegated to a modular component for consensus, which is logically separated from the peers that maintain the ledger and execute transactions. This flexibility would help bridge gaps in efficiency and thereby solve the following problems in a supply chain industry:
  - For instance, when deployed within a single-enterprise supply chain network, an absolute Byzantine Fault Tolerance (BFT) consensus protocol might be considered superfluous and an excessive drag on throughput and performance. In such scenarios, a Crash Fault Tolerance (CFT) consensus protocol might be a more efficient and effective choice.
  - On the other hand, in a decentralized multi-party supply chain network, a more traditional BFT consensus protocol might be required, as this use-case calls for a more robust and reliable consensus protocol capable of handling a larger number of transactions in parallel.
- **Speed** Given the relatively high speeds and throughputs achieved by using Hyperledger Fabric (compared to other alternative solutions), it helps solve the problem of real-time provenance tracking in a supply chain network with as little latency as possible.
- **Pluggable State Database** Hyperledger Fabric provides multiple options for the implementation of the state database, including the option of a fully custom one, hence making the approach pluggable. For instance, given the two major options available, LevelDB and CouchDB, a use-case comparison can be drawn for a typical supply chain framework:
  - **CouchDB** This implementation will be useful for extensive ledger states that are structured as JSON documents, and when there are "richer" data types being used to track the assets in business transactions. This implementation comes at a cost of performance as it requires more management because CouchDB runs as an independent process in the operating system, having a one-to-one relation between a peer node and a CouchDB instance. Here, access rights can be defined per database, and since this implementation is external, scalability is easy. The solution that this implementation will cater to is in the case of a multi-party or multi-enterprise supply chain network in which various assets need to be traced with an extensive amount of detail for each one of them, in which case the network must be able to grow on demand as well.

Here, the trade-off can be made in favor of robustness over speed and performance.

– **LevelDB** This database type is appropriate for use-cases where the ledger needs to contain simple key-value pairs. A LevelDB database is stored along with the peer node, and it is embedded within the same operating system process. This implementation improves performance and requires less management as it is embedded within the peer process. LevelDB, as opposed to CouchDB, can be used in the case of a small and contained single-enterprise supply chain network for a static asset being tracked with minimal information. Here there would be no need for an extensive or robust structure, so the trade-off would be in favor of performance and ease of management.

All peers on a channel are bound to use the same database, and the choice of database is always permanently fixed before deploying a production peer, so the database type cannot be modified later. This not only allows us to configure individual state databases for each channel depending on the business requirements and assets being tracked in the supply chain network, but also allows for immutability once the choice is made, so that it is impossible for the system to break in the future due to compatibility or modification. Overall, this helps in enhancing efficiency and flexibility by exploiting a wide range of state database implementations.

- **Unlinkability and Anonymity** The paper's proposed solution has implemented the identity mixer as opposed to the typical solution of using Fabric CA with X.509 certificates, mainly to introduce unlinkability and anonymity inside the supply chain network, along with minimal disclosure of attributes. The process of signature verification requires an X.509 certificate (containing the public key) in which all the attributes have to be disclosed. Since attributes are always disclosed, this shows that it is possible for all certificate instances to be linked. To preclude such linkability, fresh X.509 certificates can be used for each instance, but this would lead to overheads in communication and complex key management. To avoid such overhead and overcome the problem of linkability, identity mixers are used—they make sure that neither the CA nor verifiers are able to put together presentation tokens linking to the original set of credentials. The underlying Zero-Knowledge Proof (ZKP) help in proving that the signature on certain attributes is valid and that the user is in possession of the corresponding credential's secret key without revealing the signature and attribute values themselves. In terms of a practical supply chain use-case, this very much comes into the picture during an audit. Auditability is made more efficient and private with the help of the identity mixer. For instance, given that multiple TCerts (Transaction Certificates) are derived from an ECert, these TCerts can be used to independently sign various transactions across or within channels and chaincode. During the audit phase, a particular auditor need not require all the details to be revealed or be able to link them with other attributes that are disclosed. In some supply chain networks, third-party auditors might be employed, so there needs to be privacy between the transactions (across or within channels) as there are different business entities participating in the network. Here, the auditors will be able to verify the transaction without requiring all the attributes to be disclosed, as ZKP will allow for the proof to happen without disclosure of attributes outside of the specific transaction. In addition, some business requirements might require that particular components in each channel to be audited by one specific third-party auditor, so in this case, one auditor will be assigned to a specific attribute throughout the network, so any information outside of it will be inaccessible and hidden from the auditor. Hence, this solves the inherent problem of trustful auditing and bias in a supply chain network by adding features that transform it into a trustless auditing model insusceptible to bias, all without impacting the overall functionality of the process.

6. Performance analysis

The proposed system is simulated using a personal computer with an Intel Core i7 1.820 GHz × 4 processor and 8 GB memory with the Ubuntu 20.04 operating system. For development purposes, Hyperledger Fabric 1.4.7, Docker 19.03.8, Couch-DB v1.4, and the Go language are used. For consensus, RAFT is used. In the simulation, three orderers and five organizations are created. Each organization consists of two peers, namely, peer0 and peer1. Each peer is associated with the corresponding chaincode and ledger, as shown in Fig. 4. To do performance analysis, the Caliper tool is used.

Caliper is a framework developed by Hyperledger to run blockchain benchmarks. It measures the performance of blockchain network implementations and captures relevant metrics to depict the same. Caliper reports contain performance metrics like TPS, latency, CPU, disk utilization, etc. Caliper is used to analyze individual blockchain to assess their viability for a given use-case. The intent is not to compare different blockchain implementations as the setups and functionalities could vary to such an extent that the caliper reports will not be able to offer a sufficient comparison.

For the scope of this implementation, a Caliper benchmark was used to generate a report. The benchmark automated two transactions—CreateBatch and QueryBatch. The two transactions were chosen to show the performance of methods that write data as well as read data from the Blockchain network. The benchmark automated 100 transactions each at a fixed feedback rate of 25 TPS, i.e., transactions are sent at a fixed rate initially, but in the event of an unfinished transaction taking too much time, Caliper will wait to stop sending input transactions by temporarily sleeping. The benchmark is executed with all transactions submitted successfully. The performance of orderers and peers of all participants is measured for CreateBatch and QueryBatch through resource utilization. The resources are considered here: processor, memory, network, and disk. Tables 3 and 4 show the resource utilization as captured during the automation of the CreateBatch and QueryBatch functions, respectively, using attributes: CPU, Memory, Traffic In, Traffic Out, Disc Read, and Disc Write.

Next, the metrics, transactions executed per second (TPS), transaction latency (t_l) and transaction throughput (t_t) are used to measure the performance. Suppose c_t is transaction confirmation time, n_t network, and s_t is submit time for the transaction in the blockchain network, then transaction latency (t_l) is given by:

$$t_l = (c_t \times n_t) - s_t \tag{1}$$

From Eq. (1), transaction latency is nothing but the time taken by the transaction in the network. The latency and throughput are analyzed in Table 3 for sending 100 TPS to execute CreateBatch and QueryBatch. From Table 5, it is observed that the QueryBatch process is faster than CreateBatch because it performs only read operations.

Next, suppose t_{ct} is the transaction committed on the entire network, n_{nt} is the number of nodes in the network on which transactions are committed, and t_{ts} is the time taken to execute transactions successfully on n_{nt}, then transaction throughput is given by:

$$\mathrm{t_t} = \frac{t_{ct}}{t_{ts}} \times \mathrm{n_{nt}} \tag{2}$$

For further analysis, the impact of a varying number of transactions at fixed TPS and varying TPS at a fixed number of transactions are discussed in subsequent subsections.

6.1. Measurements with varying number of transactions

To understand the impact of the total number of transactions on blockchain throughput and latency, the total number of transactions varied from 50, 100, 150, 200, to 250, and these transactions were sent to the proposed network at a rate of 25 TPS.

Table 3
Resource utilization for CreateBatch.

No.	Name	CPU %(max)	CPU %(avg)	Memory (max) [MB]	Memory (avg) [MB]	Traffic In [KB]	Traffic Out [MB]	Disc Write [MB]	Disc Read [KB]
1	orderer1.supplychain.com	5.44	1.04	41.6	40.6	749	1.77	1.11	12.0
2	orderer2.supplychain.com	6.70	1.34	27.6	26.1	686	1.12	1.11	632
3	orderer3.supplychain.com	5.43	0.99	30.6	29.4	771	1.28	1.11	288
4	peer0.farminspector.supplychain.com	26.40	5.96	138	138	982	2.00	0.383	88.0
5	peer0-farminspector-couchdb	34.38	5.90	93.1	93.1	26.9	0.0228	0.00	0.00
6	peer1.farminspector.supplychain.com	13.60	3.46	65.9	65.6	675	0.139	0.383	16.0
7	peer1-farminspector-couchdb	39.22	6.88	59.0	58.5	32.5	0.0250	0.00	12.0
8	peer0.harvester.supplychain.com	22.69	4.80	229	229	942	0.968	0.285	16.0
9	peer0-harvester-couchdb	38.55	7.02	100	100	25.3	0.0210	0.00	32.0
10	peer1.harvester.supplychain.com	9.00	2.68	103	103	598	0.118	0.285	16.0
11	peer1-harvester-couchdb	32.46	5.83	60.2	59.4	25.1	0.0205	0.00	56.0
12	peer0.exporter.supplychain.com	22.47	4.84	181	181	943	0.966	0.285	12.0
13	peer0-exporter-couchdb	34.61	6.13	89.6	89.6	25.1	0.0209	0.00	0.00
14	peer1.exporter.supplychain.com	10.69	3.27	98.1	97.8	590	0.131	0.289	12.0
15	peer1-exporter-couchdb	38.22	6.89	57.4	57.1	25.4	0.0214	0.00	44.0
16	peer0.importer.supplychain.com	24.35	5.36	475	475	934	1.03	0.289	28.0
17	peer0-importer-couchdb	35.40	5.95	84.0	84.0	25.5	0.0218	0.00	0.00
18	peer1.importer.supplychain.com	10.90	3.43	93.4	93.2	576	0.116	0.289	8.00
19	peer1-importer-couchdb	31.93	5.77	58.2	57.3	25.5	0.0219	0.00	52.0
20	peer0.processor.supplychain.com	24.53	5.23	267	267	948	1.06	0.285	28.0
21	peer0-processor-couchdb	33.71	5.57	90.3	90.3	23.9	0.0177	0.00	0.00
22	peer1.processor.supplychain.com	10.96	3.09	93.8	93.5	659	0.127	0.285	68.0
23	peer1-processor-couchdb	34.88	5.71	60.4	59.5	25.0	0.0199	0.00	24.0

Table 4
Resource utilization for QueryBatch.

No.	Name	CPU %(max)	CPU %(avg)	Memory (max) [MB]	Memory (avg) [MB]	Traffic In [KB]	Traffic Out [MB]	Disc Write [MB]	Disc Read [KB]
1	orderer1.supplychain.com	0.26	0.25	41.7	41.7	8.52	8.43	0.00	0.00
2	orderer2.supplychain.com	0.40	0.35	29.0	29.0	16.4	16.3	0.00	0.00
3	orderer3.supplychain.com	0.27	0.25	31.8	31.8	8.52	8.43	0.00	0.00
4	peer0.farminspector.supplychain.com	21.71	8.70	137	136	267	292	0.00	0.00
5	peer0-farminspector-couchdb	37.23	13.00	93.3	93.2	53.7	34.8	0.00	4.00
6	peer1.farminspector.supplychain.com	2.85	2.54	66.9	66.9	32.0	30.7	0.00	0.00
7	peer1-farminspector-couchdb	3.14	1.52	59.2	59.2	0.0840	0.00	0.00	4.00
8	peer0.harvester.supplychain.com	20.42	8.22	222	222	296	343	0.00	0.00
9	peer0-harvester-couchdb	45.85	15.55	98.4	98.3	54.1	46.9	0.00	4.00
10	peer1.harvester.supplychain.com	2.06	1.44	104	103	31.8	31.7	0.00	0.00
11	peer1-harvester-couchdb	1.94	1.53	61.3	61.3	0.0840	0.00	0.00	4.00
12	peer0.exporter.supplychain.com	19.72	7.71	177	177	258	276	0.00	0.00
13	peer0-exporter-couchdb	39.77	13.61	89.8	89.7	53.3	32.3	0.00	8.00
14	peer1.exporter.supplychain.com	2.51	2.21	101	101	32.8	34.4	0.00	0.00
15	peer1-exporter-couchdb	3.49	2.24	58.7	58.7	0.0840	0.00	0.00	4.00
16	peer0.importer.supplychain.com	17.86	7.17	452	452	255	282	0.00	0.00
17	peer0-importer-couchdb	34.90	12.28	84.1	84.1	53.0	31.9	0.00	0.00
18	peer1.importer.supplychain.com	4.00	2.74	93.9	93.9	29.6	26.0	0.00	0.00
19	peer1-importer-couchdb	3.56	2.08	59.2	59.1	0.0840	0.00	0.00	4.00
20	peer0.processor.supplychain.com	19.49	8.06	258	258	298	345	0.00	4.00
21	peer0-processor-couchdb	42.61	14.90	90.4	90.3	53.4	44.9	0.00	4.00
22	peer1.processor.supplychain.com	2.89	1.96	97.8	97.8	30.9	27.7	0.00	0.00
23	peer1-processor-couchdb	2.13	1.63	61.8	61.8	0.0840	0.00	0.00	0.00

Table 5
Summary of performance metrics for QueryBatch.

No.	Number of transactions	Send Rate (TPS)	Max Latency (s)	Min Latency (s)	Avg Latency (s)	Throughput (TPS)
1	50	106.4	1.26	0.11	1.04	34.7
2	100	160.8	2.38	0.28	1.97	35.0
3	150	154.6	4.82	2.75	4.39	27.5
4	200	159.4	5.70	1.93	5.28	30.7
5	250	159.4	5.70	1.93	5.28	30.7

Table 6
Performance analysis for varying number of transactions at 25 TPS for CreateBatch.

No.	Number of transactions	Send Rate (TPS)	Max Latency (s)	Min Latency (s)	Avg Latency (s)	Throughput (TPS)
1	50	26.1	12.57	3.88	8.27	3.5
2	100	25.5	16.88	3.12	9.96	4.9
3	150	24.3	29.84	2.08	16.30	4.4

Table 6 shows the performance for the create transaction through metrics—throughput and latency at varying the send rates. Table 6 is described through the chart in Fig. 12. Table 7 shows the performance for the query transaction through the same metrics. Table 7 is described through the chart in Fig. 13. From both charts, it is observed that as the number of transactions increased, the average latency also increased.

Table 7
Performance analysis for varying number of transactions at 25 TPS for QueryBatch.

No.	Number of transactions	Send Rate (TPS)	Max Latency (s)	Min Latency (s)	Avg Latency (s)	Throughput (TPS)
1	50	106.4	1.26	0.11	1.04	34.7
2	100	160.8	2.38	0.28	1.97	35.0
3	150	154.6	4.82	2.75	4.39	27.5
4	200	159.4	5.70	1.93	5.28	30.7
5	250	196.7	6.29	0.07	5.60	34.1

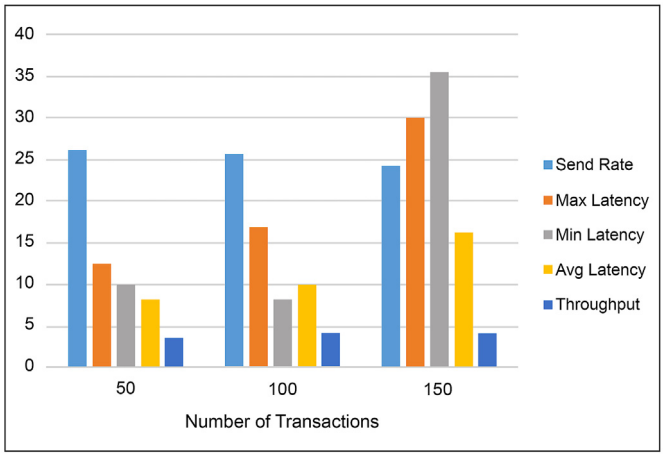

Fig. 12. Calculations with varying numbers of transactions—CreateBatch.

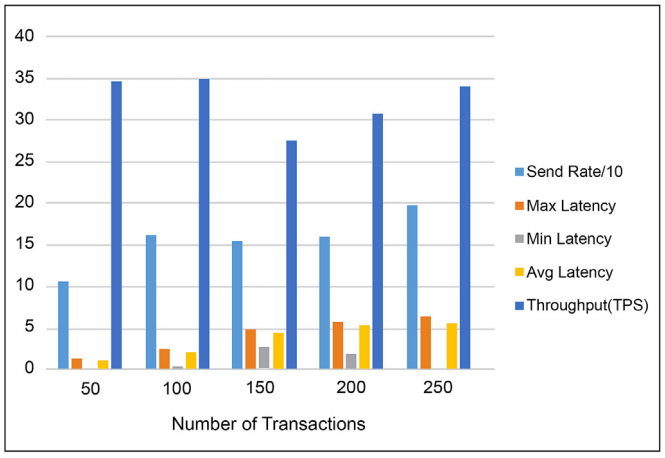

Fig. 13. Calculations with varying number of transactions—QueryBatch.

6.1.1. Evaluation of resource consumption

For resource consumption, during the execution of caliper testing for the proposed network, various parameters like CPU utilization, memory consumption, and traffic I/O are measured for peer0 of each organization. For orderer1, the values are also calculated. Tables 5, 8–10, and 15–17 show the resource utilization details for query transaction with varying numbers of transactions: 50, 100, 150, 200, and 250, respectively, at 25 TPS (see Table 7).

6.2. Measurements with varying TPS

Here, we focus on the impact of the varying transaction rate (TPS) through throughput and latency. The total number of transactions 300 at TPS 100, 200, and 250 are calculated as shown in Table 12. Table 14 shows the performance analysis for varying TPS for 300 transactions.

Table 12 is described through the chart in Fig. 14. From the chart, it is observed that as the number of transactions increased, the average latency also increased.

6.3. Evaluation of resource consumption

For resource consumption, during the execution of caliper testing for the proposed network, various parameters like CPU utilization, memory consumption, and traffic I/O are measured for the peer0 of each organization. For orderer1, the values are also calculated. Tables 11–13 show the resource utilization details for create transactions for varying TPS at 50, 100, and 150 number of transactions, respectively.

7. Conclusions

This paper intends to examine the idea of integrating permissioned blockchain technology into the existing process of the coffee supply chain. Initially, a literature survey is performed to understand the current issues and the latest progress concerning the supply chain industry, such as provenance tracking and transparency. Then, a survey is performed on the challenges posed by public blockchains, thus paving the way for permissioned blockchains and how they help to mitigate the challenges and also introduce new features such as privacy and security. A case is then composed, taking the coffee supply chain industry as an example, for Hyperledger Fabric DLT to solve these prevailing problems. It is also established that the design features, specifically membership and access control, included in Hyperledger Fabric make it ideal for integrating blockchain technology into existing enterprise solutions, especially due to its permission management, fine-grained access control, anonymity, and pluggable consensus algorithm. Based on these features, a comparison is drawn between the proposed solution and the contemporary solutions, proving how Hyperledger Fabric DLT emerges as a clear winner in terms of advantages, features, and practicality, thereby bridging gaps and solving several problems. Then a performance analysis is done over various parameters and benchmarks, illustrating Hyperledger Fabric's high transaction performance. Although the research conducted mainly dealt with the coffee supply chain industry, it can be stressed that this

Table 8
Resource utilization for QueryBatch 50.

No.	Name	CPU %(max)	CPU %(avg)	Memory (avg) [MB]	Traffic In [KB]	Traffic Out [KB]
1	peer0.exporter.supplychain.com	15.86	9.25	96.8	318	330
2	peer0.importer.supplychain.com	15.21	9.53	106	305	303
3	peer0.processor.supplychain.com	12.77	8.78	104	306	319
4	peer0.harvester.supplychain.com	11.66	8.61	110	311	309
5	peer0.farminspector.supplychain.com	12.92	9.53	169	323	676
6	orderer1.supplychain.com	1.56	1.03	34.1	140	140

Table 9
Resource utilization for QueryBatch 100.

No.	Name	CPU %(max)	CPU %(avg)	Memory (avg) [MB]	Traffic In [KB]	Traffic Out [KB]
1	peer0.exporter.supplychain.com	9.24	7.66	96.9	49.4	54.2
2	peer0.importer.supplychain.com	10.44	7.36	106	48.7	55.8
3	peer0.processor.supplychain.com	8.43	7.26	104	48.1	52.0
4	peer0.harvester.supplychain.com	10.46	7.78	110	43.1	42.0
5	peer0.farminspector.supplychain.com	37.28	21.75	169	535	692
6	orderer1.supplychain.com	1.31	0.96	34.1	19.6	19.7

Table 10
Resource utilization for QueryBatch 150.

No.	Name	CPU %(max)	CPU %(avg)	Memory (avg) [MB]	Traffic In [KB]	Traffic Out [KB]
1	peer0.exporter.supplychain.com	13.09	8.47	96.9	55.2	61.4
2	peer0.importer.supplychain.com	17.04	9.00	106	60.0	58.0
3	peer0.processor.supplychain.com	13.25	8.98	104	58.2	59.7
4	peer0.harvester.supplychain.com	13.15	7.47	110	61.0	57.4
5	peer0.farminspector.supplychain.com	111.27	30.79	170	575	655
6	orderer1.supplychain.com	1.00	0.85	34.1	25.3	25.3

Table 11
Resource utilization for CreateBatch 50.

No.	Name	CPU %(max)	CPU %(avg)	Memory (avg) [MB]	Traffic In [KB]	Traffic Out [KB]
1	peer0.exporter.supplychain.com	20.11	9.48	115	629	0.418
2	peer0.importer.supplychain.com	16.27	9.89	117	650	0.651
3	peer0.processor.supplychain.com	18.08	10.34	122	619	0.602
4	peer0.harvster.supplychain.com	18.79	10.10	117	620	0.412
5	peer0.farminspector.supplychain.com	21.59	11.17	450	787	1.00
6	orderer1.supplychain.com	31.88	3.52	76.2	551	0.936

Table 12
Resource utilization for CreateBatch 100.

No.	Name	CPU %(max)	CPU %(avg)	Memory (avg) [MB]	Traffic In [KB]	Traffic Out [KB]
1	peer0.exporter.supplychain.com	18.38	12.46	116	684	0.287
2	peer0.impoter.supplychain.com	21.30	13.98	117	695	0.690
3	peer0.processor.supplychain.com	20.95	11.57	122	700	0.696
4	peer0.harvester.supplychain.com	20.35	11.49	118	683	0.288
5	peer0.farminspector.supplychain.com	34.01	17.44	450	984	0.730
6	orderer1.supplychain.com	15.58	2.95	76.1	576	1.35

Table 13
Resource utilization for CreateBatch 150.

Sr.No.	Name	CPU %(max)	CPU %(avg)	Memory (avg) [MB]	Traffic In [KB]	Traffic Out [KB]
1	peer0.exporter.supplychain.com	23.08	12.61	117	1.06	1.08
2	peer0.importer.supplychain.com	24.30	14.21	117	1.06	0.480
3	peer0.processorr.supplychain.com	20.01	12.48	119	1.05	0.489
4	peer0.havester.supplychain.com	31.95	15.83	122	1.07	1.08
5	peer0.farminspector.supplychain.com	33.62	16.09	450	1.47	1.28
6	orderer1.supplychain.com	23.63	5.07	79.3	1.01	2.19

proposed solution could also be expanded to other supply chains. Finally, the paper asserts that permissioned blockchain platforms, particularly Hyperledger Fabric, could make the supply chain industry more reliable, robust, modular, and trustworthy.

There is immense scope for future work building on top of the proposed solution and architecture in this paper. In Hyperledger Fabric, the default implementation allows for atomic transactions within a channel but not between channels. Introducing atomicity between channels would be really useful and would improve the integrity of transactions and data. The utilization of ZKP is not exploited to its fullest potential in the current implementation of Hyperledger Fabric, so extending this feature would certainly enhance the security and privacy of the blockchain multifold. Finally, there is also an opportunity for integrating a Hardware Secure Module (HSM), in which case the private keys would be

Table 14
Performance analysis for varying tps for 300 transactions.

Sr No	TPS	Send Rate (TPS)	Max Latency (s)	Min Latency (s)	Avg Latency (s)	Throughput (TPS)
1	100	99.9	6.48	5.45	5.93	35.4
2	200	169.9	5.16	0.08	4.58	45.7
3	250	182.1	6.85	0.08	5.76	39.5

Table 15
Resource utilization for QueryBatch 100.

Sr.No.	Name	CPU %(max)	CPU %(avg)	Memory (avg) [MB]	Traffic In [KB]	Traffic Out [KB]
1	peer0.exporter.supplychain.com	13.27	8.85	97.8	0.274	0.276
2	peer0.importer.supplychain.com	15.30	9.23	106	0.271	0.279
3	peer0.processor.supplychain.com	13.88	9.09	104	0.268	0.268
4	peer0.harvester.supplychain.com	15.16	9.31	110	0.267	0.276
5	peer0.farminspector.supplychain.com	67.21	15.85	172	1.26	1.72
6	orderer1.supplychain.com	1.76	1.05	34.1	0.127	0.127

Table 16
Resource utilization for QueryBatch 200.

Sr.No.	Name	CPU %(max)	CPU %(avg)	Memory (avg) [MB]	Traffic In [KB]	Traffic Out [KB]
1	peer0.exporter.supplychain.com	13.87	10.31	97.4	62.8	0.0593
2	peer0.importer.supplychain.com	14.05	9.29	106	56.3	0.0594
3	peer0.processor.supplychain.com	13.38	10.70	104	54.7	0.0520
4	peer0.harvester.supplychain.com	12.32	9.06	110	59.3	0.0575
5	peer0.farminspector.supplychain.com	51.11	25.48	172	983	1.02
6	orderer1.supplychain.com	3.01	1.29	34.1	25.4	0.0248

Table 17
Resource utilization for QueryBatch 250.

Sr.No.	Name	CPU %(max)	CPU %(avg)	Memory (avg) [MB]	Traffic In [KB]	Traffic Out [KB]
1	peer0.exporter.supplychain.com	10.53	8.35	97.2	0.0631	0.0643
2	peer0.importer.supplychain.com	12.19	8.07	106	0.0676	0.0667
3	peer0.processor.supplychain.com	12.22	8.86	104	0.0652	0.0679
4	peer0.harvester.supplychain.com	14.52	9.01	109	0.0694	0.0653
5	peer0.farminspector.supplychain.com	44.05	23.53	172	1.26	1.59
6	orderer1.supplychain.com	1.15	0.95	34.1	0.0321	0.0321

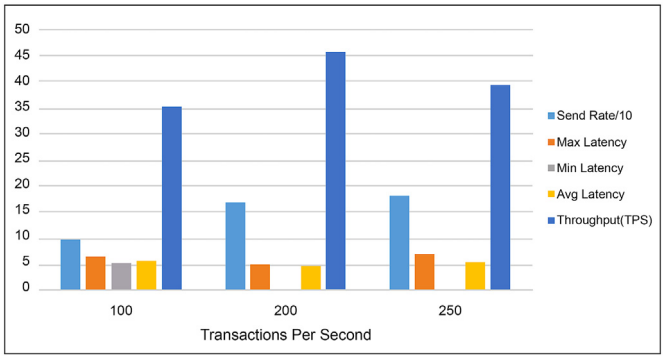

Fig. 14. Calculations with varying TPS (transactions per second).

completely hidden and protected within the HSM, and all the cryptographic operations would take place inside the HSM with no secret information leaving the premises of the hardware, thereby preventing exposure of the keys and secrets to software hacks.

Declaration of competing interest

We wish to confirm that there are no known conflicts of interest associated with this publication and that there has been no significant financial support for this work that could have influenced its outcome.

We confirm that the manuscript has been read and approved by all named authors and that there are no other persons who satisfied the criteria for authorship but are not listed. We further confirm that the order of authors listed in the manuscript has been approved by all of us.

We confirm that we have given due consideration to the protection of intellectual property associated with this work and that there are no impediments to publication, including the timing of publication, with respect to intellectual property. In so doing, we confirm that we have followed the regulations of our institutions concerning intellectual property.

We understand that the corresponding author is the sole contact for the editorial process (including the editorial manager and direct communication with the office). She is responsible for communicating with the other authors about progress, submissions of revisions, and final approval of proofs. We confirm that we have provided a current, correct email address that is accessible by the corresponding author and which has been configured to accept email from brindham@nitt.edu.

References

[1] S. Nakamoto, Bitcoin: a peer-to-peer electronic cash system, 2008. Available online: http://www.bitcoin.org/bitcoin.pdf. (Accessed 1 March 2021).

[2] V. Buterin, A next-generation smart contract and decentralized application platform, 2014. Available online: https://ethereum.org/en/whitepaper/. (Accessed 1 March 2021).

[3] M. Vukolić, Rethinking permissioned blockchains, in: Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts; 2 Apr 2017; Abu Dhabi, UAE. ACM, New York, NY, USA, 2017, pp. 3–7.

[4] E. Androulaki, A. Barger, V. Bortnikov, et al., Hyperledger fabric: a distributed operating system for permissioned blockchains, in: Proceedings of the Thirteenth EuroSys Conference; 23–26 Apr 2018; Porto, Portugal, ACM, New York, NY, USA, 2018, pp. 1–15.

[5] P. Tasatanattakoolm, C. Techapanupreeda, Blockchain: challenges and applications, in: 2018 International Conference on Information Networking (ICOIN); 10–12 Jan 2018; Chiang Mai, Thailand, IEEE, Piscataway, NJ, USA, 2018, pp. 473–475.

[6] H. Juma, K. Shaalan, I. Kamel, A survey on using blockchain in trade supply chain solutions, IEEE Access 7 (2019) 184115–184132.

[7] J.M. Song, J. Sung, T. Park, Applications of blockchain to improve supply chain traceability, Procedia Comput. Sci. 162 (2019) 119–122.

[8] X. Jing, Z. Dongjie, M. Zhongsu, The research on the BP neural network application in food supply chain risk management, Int. Conf. Inf. Manag. Innov. Manag. Ind. Eng. 1 (2009) 545–548.

[9] F. Tian, An agri-food supply chain traceability system for China based on RFID & blockchain technology, in: 13th international conference on service systems and service management (ICSSSM); 24–26 Jun 2016; Kunming, China, IEEE, Piscataway, NJ, USA, 2016, pp. 1–6.

[10] M. Li, Z.-X. Jin, C. Chen, Application of RFID on products tracking and tracing system, Comput. Integr. Manuf. Syst. 16 (1) (2010) 202–208.

[11] H. Zhang, S. Xu, R. Liu, Application of RFID technology in vegetable cold-chain logistics management, Logistics Technol. 33 (4) (2014) 348–353.

[12] J.-Y. Yeh, S.-C. Liao, Y.-T. Wang, et al., Understanding consumer purchase intention in a blockchain technology for food traceability and transparency context, in: 2019 IEEE Social Implications of Technology (SIT) and Information Management (SITIM); 9–10 Nov 2019; Matsuyama, Japan. IEEE, Piscataway, NJ, USA, 2019, pp. 1–6.

[13] K. Salah, N. Nizamuddin, R. Jayaraman, M. Omar, Blockchain-based soybean traceability in agricultural supply chain, IEEE Access 7 (2019) 73295–73305.

[14] R. Kamath, Food traceability on blockchain: Walmart's pork and mango pilots with IBM, J. British Blockchain Assoc. 1 (1) (2018) 1–12.

[15] Q. Lin, H. Wang, X. Pei, J. Wang, Food safety traceability system based on blockchain and EPCIS, IEEE Access 7 (2019) 20698–20707.

[16] Q. Tao, X. Cui, X. Huang, A.M. Leigh, H. Gu, Food safety supervision system based on hierarchical multi-domain blockchain network, IEEE Access 7 (2019) 51817–51826.

[17] B.M.A.L. Basnayake, C. Rajapakse, A blockchain-based decentralized system to ensure the transparency of organic food supply chain, in: 2019 International Research Conference on Smart Computing and Systems Engineering (SCSE); 28 Mar 2019; Colombo, Sri Lanka, IEEE, Piscataway, NJ, USA, 2019, pp. 103–107.

[18] G. Baralla, A. Pinna, G. Corrias, Ensure traceability in European food supply chain by using a blockchain system, in: 2019 IEEE/ACM 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB); 27 May 2019; Montreal, QC, Canada, IEEE, Piscataway, NJ, USA, 2019, pp. 40–47.

[19] Y.P. Tsang, K.L. Choy, C.H. Wu, G.T.S. Ho, H.Y. Lam, Blockchain-driven IOT for food traceability with an integrated consensus mechanism, IEEE Access 7 (2019) 129000–129017.

[20] H. Hayati, I.G.B.B. Nugraha, Blockchain based traceability system in food supply chain, in: 2018 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI); 21–22 Nov 2018; Yogyakarta, Indonesia, IEEE, Piscataway, NJ, USA, 2018, pp. 120–125.

[21] A. Shahid, A. Almogren, N. Javaid, et al., Blockchain-based agri-food supply chain, IEEE Access 8 (2020) 69230–69243.

[22] K. Biswas, V. Muthukkumarasamy, W.L. Tan, Blockchain based wine supply chain traceability system, in: Future Technologies Conference (FTC) 2017; 29–30 Nov 2017; Vancouver, BC, Canada. The Science and Information Organization, West Yorkshire, UK, 2017, pp. 56–62.

[23] S. Wang, D. Li, Y. Zhang, J. Chen, Smart contract-based product traceability system in the supply chain scenario, IEEE Access 7 (2019) 115122–115133.

[24] K. Toyoda, P. Takis Mathiopoulos, I. Sasase, et al., A novel blockchain-based product ownership management system (POMS) for anti-counterfeits in the post supply chain, IEEE Access, 5 (2017) 17465–17477.

[25] S.A. Abeyratne, R. Monfared, Blockchain ready manufacturing supply chain using distributed ledger, Int. J. Renew. Energy Technol. 5 (2016) 1–10.

[26] Marija Krstić, Lazar Krstić, Hyperledger frameworks with a special focus on Hyperledger Fabric, Vojnotehnicki glasnik 68 (3) (2020) 639–663.

[27] A. Iftekhar, X. Cui, Q. Tao, C. Zheng, Hyperledger fabric access control system for internet of things layer in blockchain-based applications, Entropy 23 (8) (2021) 1054.

[28] N. Vadgama, P. Tasca, An analysis of blockchain adoption in supply chains between 2010 and 2020, Frontiers in Blockchain 4 (2020) 61047.

[29] C. Ma, X. Kong, Q. Lan, et al., The privacy protection mechanism of Hyperledger Fabric and its application in supply chain finance, Cybersecurity 2 (5) (2019) 1–9.

[30] N. Hazbiy, F. Ekadiyanto, A.A.P. Ratna, Blockchain based warehouse supply chain management using hyperledger fabric and hyperledger composer, Int. J. Inf. Technol. 9 (2020) 147–151.

[31] M. Antwi, A. Adnane, F. Ahmad, et al., Habib ur Rehman, The case of HyperLedger fabric as a blockchain solution for healthcare applications, Blockchain Res. Appl. 2 (1) (2021), 100012.

[32] S. Jabbar, H. Lloyd, M. Hammoudeh, et al., Blockchain-enabled supply chain: analysis, challenges, and future directions, Multimed. Syst. 27 (2021) 787–806.

[33] W. Li, H. Guo, M. Nejad, et al., Privacy-preserving traffic management: a blockchain and zero-knowledge proof inspired approach, IEEE Access 8 (2020) 181733–181743.

[34] Fabrice Benhamouda, Shai Halevi, Tzipora Halevi, Supporting private data on hyperledger fabric with secure multiparty computation, IBM J. Res. Dev. 63 (2/3) (2019) 3:1–3:8.

[35] S. Brotsis, N. Kolokotronis, K. Limniotis, et al., On the security and privacy of hyperledger fabric: challenges and open issues, in: 2020 IEEE World Congress on Services (SERVICES); 18–23 Oct 2020; Beijing, China, IEEE, Piscataway, NJ, USA, 2020, pp. 197–204.

[36] W. Li, C. Meese, H. Guo, M.M. Nejad, Blockchain-enabled identity verification for safe ridesharing leveraging zero-knowledge proof, in: 2020 3rd International Conference on Hot Information-Centric Networking (HotICN); 12–14 Dec 2020; Hefei, China, IEEE, Piscataway, NJ, USA, 2020, pp. 18–24.

[37] S. Delgado-Segura, C. Pérez-Solà, G. Navarro-Arribas, J. HerreraJoancomartí, Analysis of the Bitcoin UTXO set, in: Financial Cryptography and Data Security, Springer, Heidelberg, Berlin, Germany, 2019, pp. 78–91.

[38] C. Stamatellis, P. Papadopoulos, N. Pitropakis, et al., A privacy-preserving healthcare framework using hyperledger fabric, Sensors 20 (22) (2020), 6587.

[39] A. Marchese, O. Tomarchio, An agri-food supply chain traceability management system based on Hyperledger Fabric blockchain, in: 23rd International Conference on Enterprise Information Systems; 26–28 Apr 2021; Virtual. SciTePress, Setúbal, Portugal, 2021, pp. 648–658.

[40] I. Surjandari, H. Yusuf, E. Laoh, R. Maulida, Designing a permissioned blockchain network for the halal industry using hyperledger fabric with multiple channels and the raft consensus mechanism, J. Big Data 8 (2021) 10.