Blockchain Based Monitoring on Trustless Supply Chain Processes

Ali Shahzad[1] Chen Wenyu[2] Rajesh Kumar[3]
alishahzadtariq@gmail.com cwy@uestc.edu.cn rajakunarlohano@gmail.com
Department of Computer Science and Technology[123].
University of Electronic Science and Technology of China.
Chengdu, China

2021 IEEE 6th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA) | 978-1-6654-2311-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICCCBDA51879.2021.9442512

Abstract—**Current trends in global business and enterprise typically force companies that create, distribute and sell products to employ long and diverse supply chain networks. While delivering value, these networks also have several challenges: tracking orders, receipts, and payments while tracking digital assets. This paper presents a decentralized blockchain-based traceability solution that provides secure and transparent transactions between parties in the supply chain. We design smart contracts for authenticating the agreement between multiple parties and traceability of the supply chain. Furthermore, We design a session generation scheme for authentication of the parties. The proposed design achieved user data integrity, secure key agreement, immutability, transaction traceability, and provide secure auditing. Our results show that the proposed smart contract and the session-based scheme have high scalability and performance in latency time and response time.**

Keywords—supply chain management, sovereign blockchain, smart contracts, business, off-chain processes, business continuity

I. Introduction

TRADITIONAL supply chain functions in relation to payment processing and order fulfillment suffer considerably from several challenges. Lack of transparency and trust in the supply chain are two major problems. Buyers and sellers have to have reliable systems for verifying and validating the different transactions. Currently supply chains potentially involve hundreds of stages, actors and different geographic locations. When an actor in the supply chain conducts illegal activity, investigations become a difficult task and it is often hard to identify the offender [1], [2], [3]. The payment for orders are usually handled by banks in the time frame agreed to by the transacting parties. This arrangement consequently leaves the customer access to only such information as when the ordered product will be delivered or when the proposed service will be rendered.

Modern supply chain methods continue to seek more cost-effective means of operation [4] (which pertains to in-house supply chain process), greater transparency and efficiency in the enterprise value-chain [5], [6](by ensuring fairness in a trustless business environment enabling parties to trust visibility achieved by blockchain monitoring on business processes). While large, centralized systems have been created to manage the flow of goods and data generated as a result, a single challenge remains to be addressed [7], [8], [15]. The data generated as a result of transactions can be changed from its original form [9], causing some parties to perceive the supply chain as not being fully transparent with suppliers', manufacturers' and logistics' processes[10], [11], [12].

One solution to this problem would be to develop a system that helps the customer gain access to more information on the origin of a product. It includes specific supplier information, payment mode, and shipment, ensuring significant transparency between customers and suppliers. The proposed architecture consists of three components i) users, ii) Smart Contact, and iii) blockchain network that can enhance the transparency of activities and processes between customers and suppliers. The main contributions of our framework as follows:

- We design a session generation scheme in section III. The session-based scheme for authenticating the transactions from and multiple parties or users.
- We design the agreement smart contact and transaction tractability smart contract for multiple parties in the blockchain network.
- Our results demonstrate high transaction latency, and the session-based scheme provides the authentication.

II. System Architecture

For the system being proposed we couple entity nodes with a blockchain system composed of blockchain (processing) nodes and a smart contracts center. This would permit agreed-to terms and conditions (encoded in the appropriate format) to be forwarded to the blockchain system for onward checking for consistency and enforcement.

A. Architecture

The system has three main components that interact with one another to achieve system goals. These are the Users, Smart Contract Network and the Blockchain Network. The Users consist of nodes of participating manufacturers, suppliers and others who request goods and/or services. These users each run nodes that participate in the network and take part in the creation, transmission, reception or forwarding of services/goods requests. The process of providing a service begins when any one party contacts the other with a specific service request. Typically this group comprises retailers, manufacturers, distributors and suppliers. We also include logistics services providers, consultancy firms, financial institutions,

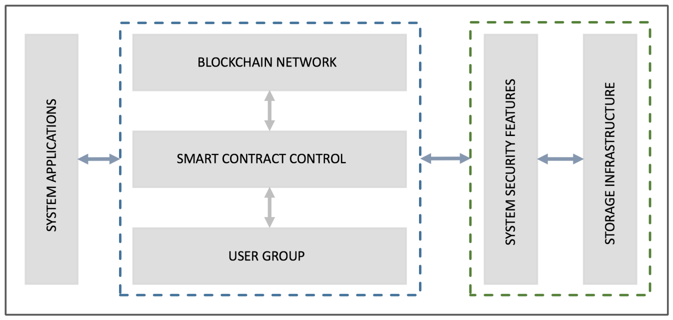

Fig. 1. System architecture.

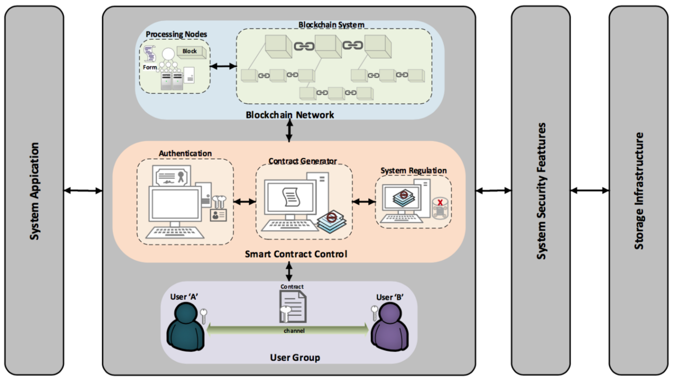

Fig. 2. Complete system architecture.

government and regulatory agencies, machine and equipment providers, and indirect materials suppliers. These, often in pairs, initiate and perpetrate actions in the system.

The blockchain system consists of a smart contracts center and a blockchain network. They work together to generate smart contracts after transacting parties have agreed on service provision requests. They also ensure quantities and timelines are transparent to all parties that may be required to observe the transaction. The smart contracts center checks received requests (contracts) for attributes necessary to permit the service request to proceed. It inspects the signatures on the requests for correctness and checks quantities and delivery conditions along with penalties for non-compliance. There may be other checks as transacting parties and system demands may deem appropriate. When complete the service request is timestamped and a hash of it is also generated and forwarded to the blockchain network. Once here, all active nodes receive a copy of the transaction. We allow for two transactions between any pair of users at a time: an ”opening” transaction and a ”closing” one. We allow for multiple transactions and messages between parties but to keep the network traffic light and performance optimal we require only opening and closing transactions. The intervening messages may pass freely back and forth between the parties through dedicated off-blockchain channels setup for the purpose. Since the system is architected on the blockchain and has consequences (legal sanctions and financial penalties) for all involved it should be the case that opening and closing transactions reconcile easily.

B. Entities of the System

The system has three main entity types. These are the Users (transacting parties), the Smart Contract Network and the Blockchain Network. Each Supply Chain party runs nodes that after setup participate in the system protocol, checking submitted contracts and flagging infractions to them as required.

1) Users: Users consist of individuals and/or organizations that request services or goods so as to initiate a two-way flow of information, materials and payments. On the basis of the products or services in question, the scope of requirements and particular demand and supply attributes, the users or participants in the supply chain network may be geographically dispersed. For our purposes we consider retailers, distributors, manufacturers, and suppliers of raw materials.
2) Smart contract network: A Smart Contract, broadly speaking, is an autonomous program that encodes the terms of a binding agreement between transacting parties and is capable of facilitating, verifying and enforcing the terms of the agreement without the involvement of third parties. The document (service or supply of materials request) is ”smart” because of the clever use of self-executing computer programs for the enforcement of agreements and its a ”contract” in the true sense of the word in that it specifies the legal relationship of obligations between the transacting parties. For the system we propose we use smart contracts to enforce checking of contract conditions necessary for carrying out specified actions. For example the Smart Contracts Center checks that the document is signed by both transacting parties. This is trivially done as the center has the public keys of all active entities on the network. It would then check for conflicting conditions in the contract document. When this passes successfully it sends the hash and timestamp of the document to the blockchain network.
3) Blockchain network: The blockchain is a distributed public ledger technology that permits the recording, verification and validation of transactions in a timestamped tamper-proof manner. Its design permits all active nodes in the network access to all the data available in the network. Protocols for validating and ordering transactions permit all nodes to arrive at one version of transaction history that is consistent with other active nodes across the network. For the supply chain system under construction the blockchain receives the hash of the signed, timestamped document for keeping. The transacting parties are free at anytime revisit the stored document (signed and timestamped) as proof of service or product request.

C. System Overview

The system we propose is intended to facilitate transactions between mutually distrustful parties by ensuring transparency of operations, correctness of quantities requested and procedures for detecting and penalizing attacks on the system without the need for trusted third parties. To begin the process it is necessary to have two parties, A and B. Party A requests a service or goods and party B steps in provide the said goods or services.

Party A sends a request (containing the quantity, location, time and date of delivery, etc. of goods) to party B in a contract. Upon reception, party B reviews the request along with its terms and conditions and sends an acknowledgement to party B if there is agreement on the content of the request. Party B is free to amend portions of the contract or its terms and request party B to review and accept before proceeding or reject the intended transaction altogether. Messages of this nature at the initial stage may go back and forth between the transacting parties in a special message channel between them till an agreement is reached. Both parties agree to take part in the transaction by signing the contract with their private keys. The last party to sign the contract sends it to the Smart Contract Network for checking of terms. At the Smart Contracts Center the submitted contract document is run through a set of rules. The center then generates a hash of the document and a timestamp and then forwards the two to the blockchain network.

Once on the blockchain network, all connected nodes receive copies of the contract hash and timestamp. The hash points to the fact of the transaction between the parties. The only valid hash to append to the preceding one (hash of the contract in the blockchain) is the hash of the ”closing” transaction which would contain facts of the fulfillment of the order with given conditions.

D. System Security Features

1) **Anonymity**: The identity of transacting parties in the system must not be revealed in the authentication process to entities aside the authenticator. Entities other than the authenticator would be identified as adversaries. An adversary on knowing the identity of a transacting party may try to get information from the system (by logging in as the identified user) and authenticating transactional messages and other relevant information and processes pertaining to the user.
2) **User untraceability**: The allocated transaction identity of a user must be anonymous and changeable in every session. This is due to the fact that an adversary should not be able to relate the several login and authentication of every session before a transaction commences.
3) **Transaction traceability**: A transactional data achieved from the completion of an agreed transaction between two transacting parties should be traceable to its source till destination as well as its start till completion. This is to ensure that an unidentified user identity can still be managed by the network and append penalties to defaulting scenarios in a transaction.
4) **Login detection**: Detection of inaccurate login credentials is necessary to avoid communication overheads(delay). Login and password changes are hampered without the appropriate implementation of this feature as a wrong input from a transacting party triggers login messages and send this to the server. The server in turn computes the messages and reject the legitimate user on the basis that the wrong parameters (credentials) has been computed.
5) **Mutual authentication**: It is a requirement on all authentication systems to verify itself and registered transaction parties in communication. This is done to ensure that transacting parties are not communicating with an adversary pretending to be the remote authentication server. It is a requirement that the authentication server ensure that communication is completed with a legitimate transacting party. This prevents masquerading and man-in-the-middle attacks.
6) **Data integrity**: Data integrity refers to the fact that the system should be resistant to login and authentication message tampering. In instances where login and authentication massages have been tampered with, the authentication protocol should be able to terminate the authentication process of such malicious entities.
7) **Secure key agreement**: After authentication, transacting parties should agree on a one-time session key and channel to complete business transactions. Session keys would be used to transact on such instances and would be used to achieve confidential transactions as well as authenticating transactional messages in the session. The session must be terminated on successful transactions and data generated on such transactions securely stored in relevant storage infrastructures.
8) **Immutability**: Logs generated on transactions should be appended on relevant blocks and indexed accordingly. These logs should be immutable to prevent malicious entities from tampering with the system.
9) **Secure auditing and provenance**: Timestamping and secure hashing algorithms used in the blockchain system can be used to ensure provenance and secure audit. Transaction timestamps help build a precise irrefutable order of events that make cause and effect deductions convenient for auditing processes and applications even with multi-event transactions. The combined effect adds to the traceability and hence the transparency of the supply chain process.

III. Session Generation

To establish a secured communication channel between two transacting parties in the business environment, there is the need to generate session keys between the transacting parties. The blockchain system aids in authenticating various transacting parties and eventually creates the channel.

For our illustration, we denote transacting parties *A* and *B* by Tx_A and Tx_B respectively. For a business transaction between the two transacting parties, party *A* chooses a random number Rn_{k_A} concatenates this with his identity ID_A in addition to transacting party *B's* identity ID_B and sends $Enc(ID_A||Rn_{k_A}||ID_B)$ to the authenticator. Transacting party *B* correspondingly chooses a random number Rn_{k_B} concatenates this with his identity ID_B in addition to transacting party *A's* identity ID_A and sends $Enc(ID_B||Rn_{k_B}||ID_A)$

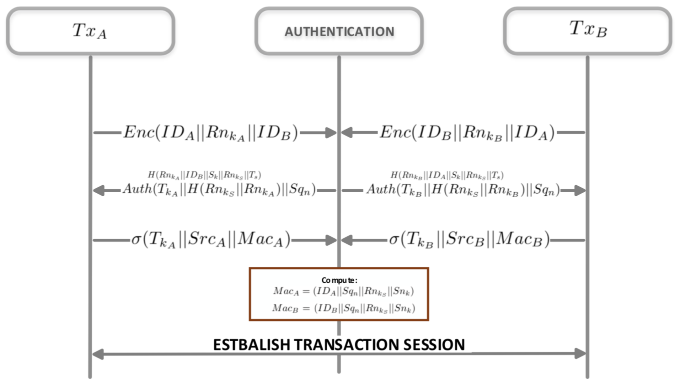
Tx_A
AUTHENTICATION
Tx_B
$Enc(ID_A||Rn_{k_A}||ID_B)$
$Enc(ID_B||Rn_{k_B}||ID_A)$
$H(Rn_{k_A}||ID_B||S_k||Rn_{k_S}||T_s)$
$Auth(T_{k_A}||H(Rn_{k_S}||Rn_{k_A})||Sq_n)$
$H(Rn_{k_B}||ID_A||S_k||Rn_{k_S}||T_s)$
$Auth(T_{k_B}||H(Rn_{k_S}||Rn_{k_B})||Sq_n)$
$\sigma(T_{k_A}||Src_A||Mac_A)$
$\sigma(T_{k_B}||Src_B||Mac_B)$
Compute:
$Mac_A = (ID_A||Sq_n||Rn_{k_S}||Sn_k)$
$Mac_B = (ID_B||Sq_n||Rn_{k_S}||Sn_k)$
ESTBALISH TRANSACTION SESSION

Fig. 3. Establishing session between two transacting parties.

to the authenticator. This is done to identify the various transacting parties and whom business relations for a particular transaction is being formed. The random number is generated to verify the authenticity of the authenticator.

The authenticator replies to both parties, A and B with a verified message $Auth(T_{k_A}||H(Rn_{k_S}||Rn_{k_A})||Sq_n)$ and $Auth(T_{k_B}||H(Rn_{k_S}||Rn_{k_B})||Sq_n)$ respectively where Sq_n is used to identify the sequence number of the message. T_{k_A} and T_{k_B} are token allocated to A and B which will be used to establish the session between the parties. The token is made up of $H(Rn_{k_A}||ID_B||S_k||Rn_{k_S}||T_s)$ and $H(Rn_{k_B}||ID_A||S_k||Rn_{k_S}||T_s)$ for A and B respectively. S_k denotes the session key to be used for the business process. Parties A and B compute and sign $\sigma(T_{k_A}||Src_A||Mac_A)$ and $\sigma(T_{k_B}||Src_B||Mac_B)$ respectively which is sent to the authenticating unit. Src_A and Src_B are used to directly connect the parties together for the business transaction processes. Mac_A and Mac_B denotes the message authentication code used to authenticate the messages sent from the transacting parties to the authenticator to specify the message came from the stated parties and has not been changed. The authenticator verifies Mac_A and Mac_B which denotes $Mac_A = (ID_A||Sq_n||Rn_{k_S}||Sn_k)$ and $Mac_B = (ID_B||Sq_n||Rn_{k_S}||Sn_k)$ and sends the Mac to respective communicating parties A and B which contains the session keys used to encrypt and decrypt transactional messages.At this point, a channel is established and parties will use the session key to hold communication on a business transaction. The session is closed when both parties agree to end a transaction and the result of the business transaction has been broadcast into the blockchain network.

IV. Smart Contract Design

A Smart Contract is an autonomous software (a self-executing contractual state) that resides on the blockchain, controlled collectively and has its actions easily verifiable. In terms of a supply chain, the smart contract permits convenient carrying out of payments once contractual obligations have been met. They are particularly relevant to supply chains because of the sheer size a chain can assume in order to facilitate the provision of requested services. Ideally transactions are initiated and carried out between pairs of requesters and suppliers but because of trust issues each party in the transaction calls in their respective banks and legal representatives. These are trusted entities whose fees usually lead to bloated service charges. The smart contract permits the removal of these in a manner that reduces costs while maintaining the security of the transaction process for all the interested parties. Smart contracts can be written to include all conditions, legal clauses, penalties and sanctions of a contract and still be enforced in a manner that the transacting parties respect. The blockchain smart contract in this paper is initialized into two categories called the **agreement contract** and **processing monitoring contract**. The first sets out the terms of the requested services while the last, as implied in the name, systematically checks the status of the requested service till it is concluded.

Algorithm 1 Agreement Contract

```
Require: Contract parameters
  Initialise: Consumer.ID, ServiceProvider.ID, SpecifyBy, Specify.Status, Confirmby.Status
  Retrieve: Contract.Formulation
  for $Specify.By$ == $Consumer.ID$ and $Specify.Status$ == $Active$ do
    Consumer.ID $\rightarrow$ Contract.Term
    Retrieve.PublicKey $\rightarrow$ Customer.ID
    Customer.ID $\rightarrow$ Contract.Sign
    Send.Contract $\rightarrow$ ServiceProvider.ID
    Timestamp $\rightarrow$ BlockchainNode
    Log $\rightarrow$ BlockchainNode
    Terminate process
  end for
  for $ConfirmBy$ == $ServiceProvider.ID$ and $Confirm.Status$ == $Active$ do
    Confirm.ConsumerSig $\leftarrow$ ServiceProvider.ID
    Retrieve.PublicKey $\rightarrow$ ServiceProvider.ID
    ServiceProvider.ID $\rightarrow$ Contract.Sign
    Timestamp $\rightarrow$ BlockchainNode
    Log $\rightarrow$ BlockchainNode
    Terminate process
  end for
  Initiate: Smart Contract Protocol
```

V. System Performance Analysis

Blockchain systems are generally heavy with all data nodes generated going straight to the blockchain network. For our system we make use of opening and closing transactions which leave intervening messages to reside only on the information systems of directly transacting parties. In this section we analyses the performance of our system against already-proposed works by implementation. We transform existing business processes as implemented in supply chain environments through a translator implemented as scripts through smart contracts in solidity.

For our blockchain-based monitoring system for supply chain processes, we design an implementation as a proof-of-concept on Ethereum in order to evaluate the system. Ethereum is a programmable blockchain that utilize solidity. We initiate

Algorithm 2 Process Monitoring Contract (A)

```
Initialize: Agreement.Contract, Session.ID, Timestamp.Start, Timestart.End, Transaction.Status
commit.Customer: Customer.ID $\rightarrow$ Network
commit.Customer = Amt.Payable + Amt.Penalty
commit.ServiceProvider: ServiceProvider.ID $\rightarrow$ Network
commit.ServiceProvider = Amt.Penalty
if Transaction.Status == Active and End.Time == Elapsed then
    Terminate procedure
    Retrieve.Contract $\rightarrow$ Network
    Allocate.Amt.Payable $\rightarrow$ Consumer.ID
    for violation == Service.Default do
        Allocate.Amt.Penalty: ServiceProvider.ID $\rightarrow$ Consumer.ID
        Retrieve.PublicKey $\rightarrow$ ServiceProvider.ID
        ServiceProvider.ID $\rightarrow$ Contract.Violation
    end for
    Retrieve.PublicKey $\rightarrow$ ServiceProvider.ID
    ServiceProvider.ID $\rightarrow$ Contract.Violation
    Retrieve.PublicKey $\rightarrow$ Customer.ID
    Customer.ID $\rightarrow$ Contract.Sign
    Timestamp $\rightarrow$ BlockchainNode
    Log $\rightarrow$ BlockchainNode
else if Transaction.Status == InActive then
    Retrieve.Contract $\rightarrow$ Network
    Check: Service.Contract
    for Service.Contract == Reached do
        Allocate.Amt.Payable $\rightarrow$ ServiceProvider.ID
        Retrieve.PublicKey $\rightarrow$ ServiceProvider.ID
        Retrieve.PublicKey $\rightarrow$ Customer.ID
        Customer.ID $\rightarrow$ Contract.Sign
    end for
    for Service.Contract == Target.Unachieved do
        Allocate.Amt.Payable $\rightarrow$ Customer.ID
        Retrieve.PublicKey $\rightarrow$ Customer.ID
        Retrieve.PublicKey $\rightarrow$ ServiceProvider.ID
        ServiceProvider.ID $\rightarrow$ Contract.Sign
    end for
    Timestamp $\rightarrow$ BlockchainNode
    Log $\rightarrow$ BlockchainNode
else {Transaction.Status == Active and End.Time == Elapsed and Time.Elapsed == ignore}
    Continue business process
end if
```

smart contracts in our system by the use of solidity which is a state-based scripting language with no restrictions on data size stored in the blockchain network. Transacting participants in the system are accounted as external which define actions on each user based on smart contracts. Smart contracts are therefore activated by initiation of a business process and agreement between transacting parties. Enforcing these contracts are achieved using Truffles which is used to initialise contract processes involving contract management, deployment and contract migration with several testing frameworks enabled through truffle boxes. These serve as the substructure fit setting up environments for testing owing to pre-configured web-development environments. Mining in our work is achieved by processing nodes through pre-processing transactions and validating business process to ensure correctness before blocks are created and broadcast into the blockchain network. This is backed by smart contract data.

Interfaces connecting transacting parties to the blockchain system use a web3.js library enabled through RPC calls. In addition, javascript libraries connect all entities in the blockchain structure by use of COAP to allow interactions among the processing nodes, blockchain system and contract generator. Partcipants in the system interact with each other in a business process through javascript web-client libraries implemented. We in-turn generate private and public key pairs to uniquely identify participants registered to the system.

We finally evaluate the integration of entity configuration for all component connections to the processing nodes. The testing of the parameters of the system is achieved by use of vertigo/ethereum image based on ethereum/go on an Ubuntu 16.04.1 operating system with the aid of docker. These refer to the initiation of transactions, validation of transacting parties, management and processing of transactions. We initialize the number of active requests from 50 transactions to 2500 transactions as active users to varied periods of time for between 10 minutes to allow for generation, processing and broadcasting of blocks.

In our system, benchmarking is achieved through jMeter which is an apache benchmarking tool. The main metrics to establish the efficiency of our system is latency, query time and response time. Latency is based on the delay before the processing of blocks are completed based on the initialization of a transaction completed by participants in the system. This includes all steps required to process a request by all participants in the supply chain system. Response time is evaluated on the length of time taken by the entire system to achieve concurrency in blocks generated per participant which directly affect the efficiency of the entire system. In considering response time, we account for query time which is based on the rate at which system requests are completed in initiating a business process in addition to generating smart contracts pertaining to business transactions. Given variations to the number of active transactional requests, we analyse the time it takes for each system to execute the total number of requests. The number of requests in the system is therefore represented as active transactions.

As represented in ***table 1*** and Fig 4, a close observation shows a considerable increase in latency as business transactions per group of participants increase in the system. This however is controlled by optimization on the methods of processing transactions by the processing nodes. This enables us to achieve an acceptable and efficient response time for business processes. Query time proves to be controlled since all transactions are completed within seconds of being created. The increase in latency is due to the trade-off of achieving security on business processes.

In comparing our proposed design to other systems as showing in *table 2* we have generated sets of metrics on the basis of these factors that our system is easy to scale due to

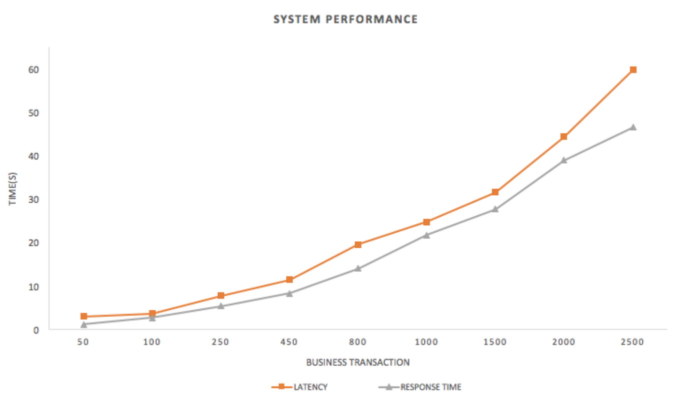

Fig. 4. System performance based on latency and response time.

TABLE I
PERFORMANCE ANALYSIS FOR BLOCKCHAIN BASED MONITORING ON TRUSTLESS SUPPLY CHAIN PROCESSES

Active Transactions	Latency(sec)	Response Time(sec)
50	3.018	1.25
100	3.71	2.73
250	7.89	5.42
450	11.46	8.35
800	19.65	14.09
1000	24.83	21.81
1500	31.61	27.73
2000	44.39	39.01
2500	59.83	46.54

off-line blockchain storage. Processing nodes have sufficient time to process the huge data volumes such a system may be required to handle. The data stored in our system is immutable due to its blockchain foundations. The high security features of our system ensure collective control of data instead of perpetrating the individual data silo by single system actors. The data tractability and data auditing built into our system also make it an efficient fit for the current supply chain environment.

TABLE II
COMPARISON BETWEEN PROPOSED SYSTEM AND OTHER RELATED SYSTEMS

Metric	[14]	[16]	[17]	Our System
Scalability	N	Y	N	Y
Off-Chain Support	N	N	N	Y
On-Chain based	Y	Y	Y	Y
Data Traceability	Y	Y	Y	Y
Data Auditing and Provenance	Y	Y	Y	Y
Immutability	Y	Y	Y	Y
Identity Management	N	N	N	Y

VI. CONCLUSION

The global nature of large supply chains results in difficult management and costly executions. Errors in requests do not just result in product delays and costly financial penalties. They can have a domino-effect on several other industries. It is therefore very important that erroneous information in supply chain systems be eliminated altogether or drastically minimized. The blockchain with its emphasis on immutability of transactions and chronological ordering of events presents a way to ensure entities in a supply chain can place requests for services whose fulfillment and payment terms can be supervised by smart contracts. By so doing we eliminate inaccurate information from participating nodes and ensure a more efficient service process. Since the system removes the necessity of third parties such as banks the service process also becomes significantly less expensive.

REFERENCES

[1] A. Brinkhoff, . zer, and G. Sargut, All you need is trust? An examination of inter-organizational supply chain projects, Prod. Oper. Manag., vol. 24, no. 2, pp. 181200, 2015.
[2] A. Singh and J. T. C. Teng, Enhancing supply chain outcomes through Information Technology and Trust, Comput. Human Behav., vol. 54, pp. 290300, 2016.
[3] J. Peng, J. Quan, G. Zhang, and A. J. Dubinsky, Mediation effect of business process and supply chain management capabilities on the impact of IT on firm performance: Evidence from Chinese firms, Int. J. Inf. Manage., vol. 36, no. 1, pp. 8996, 2016.
[4] Y. Abuidris , K. Rajesh, Y. Ting , J. Onginjo. Secure largescale Evoting system based on blockchain contract using a hybrid consensus model combined with sharding. ETRI Journal. 2020 Nov 30.
[5] E. Crisan, I. Parpucea, and L. Ilies, Models for Supply Chain Governance, Proc. 7th Eur. Conf. Manag. Leadersh. Gov., pp. 535537, 2011.
[6] A. I. Pettersson and A. Segerstedt, Measuring supply chain cost, in International Journal of Production Economics, 2013, vol. 143, no. 2, pp. 357363.
[7] S. Qrunfleh and M. Tarafdar, Supply chain information systems strategy: Impacts on supply chain performance and firm performance, Int. J. Prod. Econ., vol. 147, no. PART B, pp. 340350, 2014.
[8] J. Bastian and J. Zentes, Supply chain transparency as a key prerequisite for sustainable agri-food supply chain management, Int. Rev. Retail. Distrib. Consum. Res., vol. 23, no. 5, pp. 553570, 2013.
[9] N. u-Babi, D. Rebolj, M. Nekrep-Perc, and P. Podbreznik, Supply-chain transparency within industrialized construction projects, Comput. Ind., vol. 65, no. 2, pp. 345353, 2014.
[10] Y Abuidris., K Rajesh , and W. Wenyong. ”A survey of blockchain based on e-voting systems.” Proceedings of the 2019 2nd International Conference on Blockchain Technology and Applications. 2019.
[11] K.Rajesh, W. WenYong , K. Jay, Y, Ting, A. Waqar, and S.Abubackar . ”IoTMalware: Android IoT Malware Detection based on Deep Neural Network and Blockchain Technology.” arXiv preprint arXiv:2102.13376 (2021).
[12] K.Rajesh., Khan, A. A., Zhang, S., Wang, W., Abuidris, Y., Amin, W., and Kumar, J. (2020). Blockchain-federated-learning and deep learning models for covid-19 detection using ct imaging. arXiv preprint arXiv:2007.06537.
[13] D. Tapscott and A. Tapscott, The Impact of the Blockchain Goes Beyond Financial Services, 2016.
[14] S. A. Abeyratne and R. P. Monfared, Blockchain Ready Manufacturing Supply Chain Using Distributed Ledger, Int. J. Res. Eng. Technol., vol. 5, no. 9, pp. 16, 2016.
[15] Kumar, R., Wang, W., Kumar, J., Yang, T., Khan, A., Ali, W., and Ali, I. (2021). An Integration of Blockchain and AI for Secure Data Sharing and Detection of CT images for the Hospitals. Computerized Medical Imaging and Graphics, 87, 101812.
[16] F. Tian, An Agri-food Supply Chain Traceability System for China Based on RFID and Blockchain Technology, 2016 13th Int. Conf. Serv. Syst. Serv. Manag., pp. 16, 2016.
[17] I. Weber, X. Xu, R. Riveret, G. Governatori, A. Ponomarev, and J. Mendling, Untrusted business process monitoring and execution using blockchain, in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2016, vol. 9850 LNCS, pp. 329347.